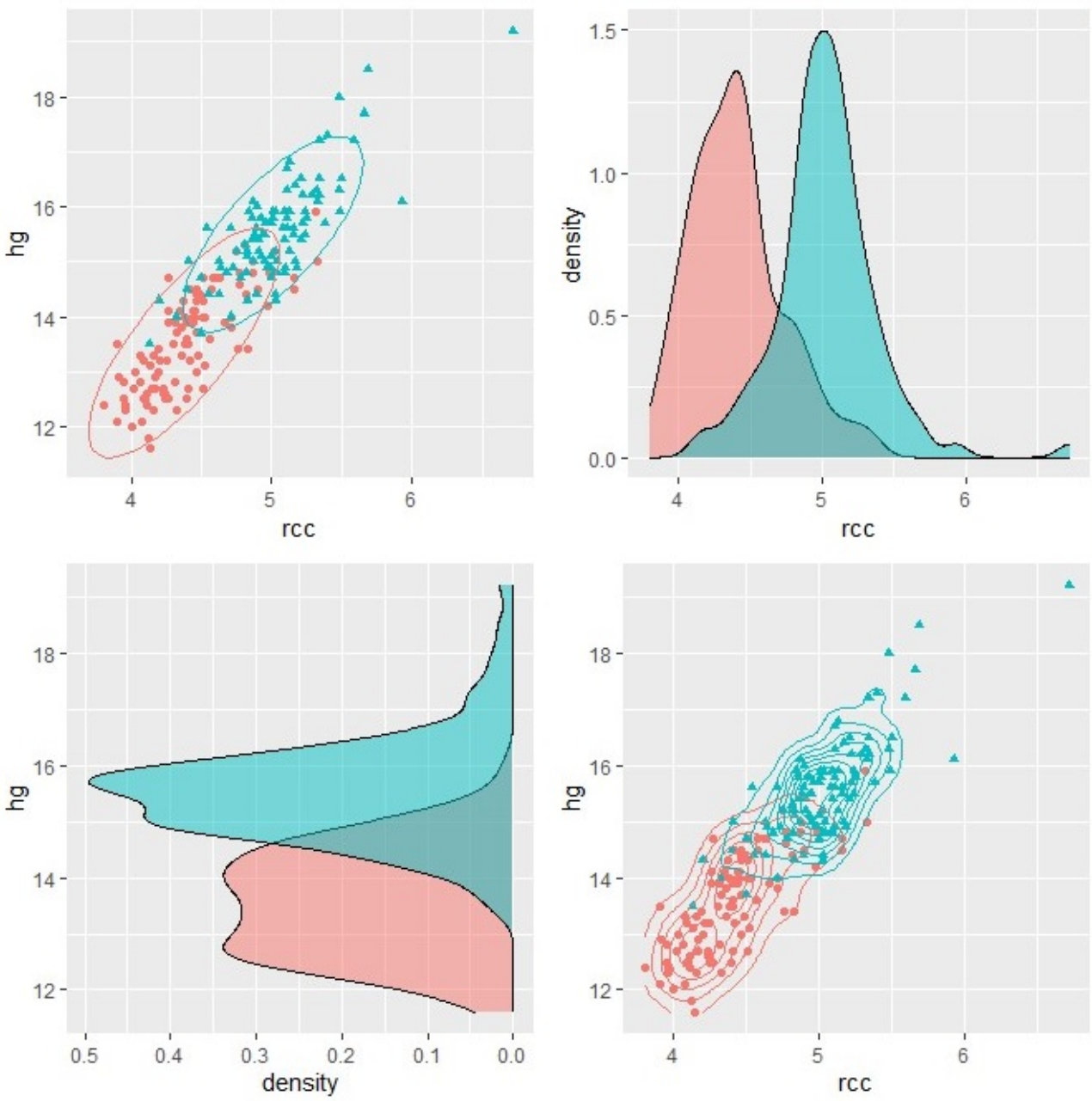
Marco Besozzi

ImparaRfacendo



Raccolta cronologica dei post da:

https://impararfacendo.blogspot.com/

NOTA BENE

Questo testo contiene la raccolta completa, alla data di aggiornamento riportata, dei post del sito https://impararfacendo.blogspot.com/
per una semplice consultazione offline in formato “simil-cartaceo” del materiale pubblicato.

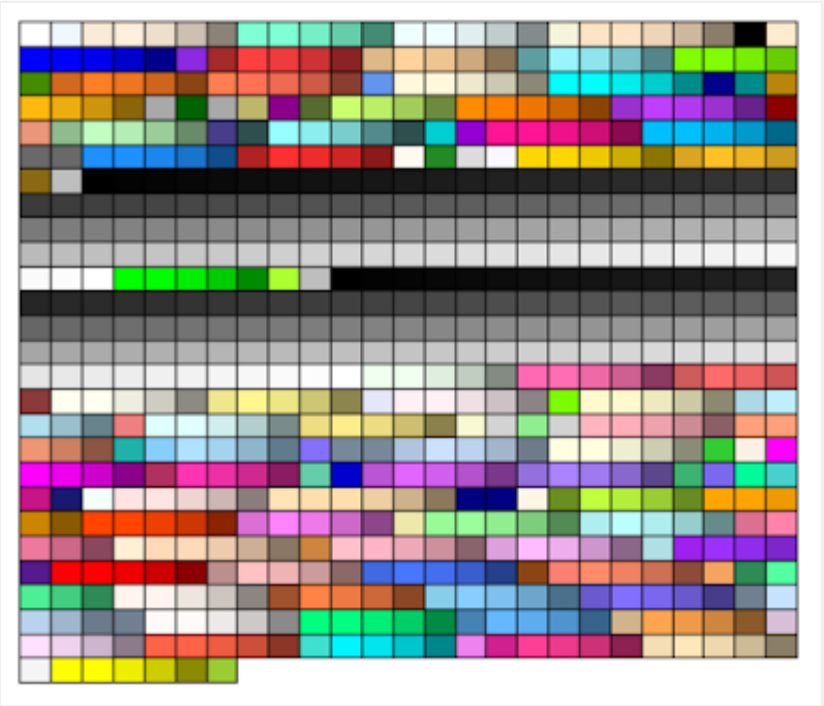
Per i file contenenti i dati impiegati negli esempi, per l'indice degli argomenti trattati e per gli script contenenti il codice che può essere direttamente eseguito con **R** si rimanda alle pagine del sito.

~~Versione 1.0 (aggiornamento aprile 2024)~~
~~Versione 1.1 (aggiornamento giugno 2024)~~
Versione 1.2 (aggiornamento agosto 2024)

mercoledì 14 novembre 2018

Nomi e codici dei colori di R

In **R** sono disponibili i 657 colori base qui rappresentati



i cui nomi vengono visualizzati nella `Console di R` digitando:

colors()

Tuttavia per visualizzare di ogni colore contemporaneamente:
→ il nome da impiegare per configurare il colore nelle funzioni che lo prevedono
→ un tassello colorato che riporta il colore corrispondente
→ il codice esadecimale che può essere impiegato in alternativa al nome
è necessario disporre di una tabella di corrispondenza che deve essere generata con uno script (non banale) come quello qui riportato, che è stato adattato da [1].

Innanzitutto se non l'avete già fatto create la cartella `C:\Rdati`

Lo script salva **la tabella dei colori di R** nel file `C:\Rdati\Rcolors.pdf` - tenete il file a portata di mano, vi servirà quando vorrete impiegare i colori di **R**.

Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`.

```
# LA TABELLA DEI COLORI DI R
# Tratto e adattato da: Earl F Glynn. Color Chart in R.
# GitHub, https://bit.ly/2WJrJrR
#
# Lo script salva la tabella dei colori di R nel file C:\Rdati\Rcolors.pdf
#
pdf("C:/Rdati/Rcolors.pdf", width=6, height=10)
oldparameters <- par(mar=c(1,1,2,1), mfrow=c(2,1))
#
stopifnot(length(colors()) == 657)
SetTextContrastColor <- function(color)
```

```
{ ifelse( mean(col2rgb(color)) > 127, "black", "white")}
TextContrastColor <- unlist( lapply(colors(), SetTextContrastColor) )
#
# tabella dei colori di R ordinati per numero
#
colCount <- 25 # number per row
rowCount <- 27
plot( c(1,colCount), c(0,rowCount), type="n", ylab="", xlab="", axes=FALSE,
ylim=c(rowCount,0))
title("Tabella dei colori di R")
for (j in 0:(rowCount-1))
{base <- j*colCount
remaining <- length(colors()) - base
RowSize <- ifelse(remaining < colCount, remaining, colCount)
rect((1:RowSize)-0.5,j-0.5, (1:RowSize)+0.5,j+0.5, border="black", col=colors()[base +
(1:RowSize)])
text((1:RowSize), j, paste(base + (1:RowSize)), cex=0.7, col=TextContrastColor[base +
(1:RowSize)])
}
#
# colori di R ordinati per tonalità e saturazione
#
RGBColors <- col2rgb(colors()[1:length(colors())])
HSVColors <- rgb2hsv( RGBColors[1,], RGBColors[2,], RGBColors[3,],
maxColorValue=255)
HueOrder <- order( HSVColors[1,], HSVColors[2,], HSVColors[3,] )
plot(0, type="n", ylab="", xlab="", axes=FALSE, ylim=c(rowCount,0),
xlim=c(1,colCount))
title("Colori di R ordinati per tonalità e saturazione")
for (j in 0:(rowCount-1))
{for (i in 1:colCount)
{k <- j*colCount + i
if (k <= length(colors()))
{rect(i-0.5,j-0.5, i+0.5,j+0.5, border="black", col=colors()[ HueOrder[k] ])
text(i,j, paste(HueOrder[k]), cex=0.7, col=TextContrastColor[ HueOrder[k] ])}}}
#
# colori di R con nomi e codice esadecimale del colore
#
GetColorHexAndDecimal <- function(color)
{c <- col2rgb(color)
sprintf("#%02X%02X%02X %3d %3d %3d", c[1],c[2],c[3], c[1], c[2], c[3])}
par(oldparameters)
oldparameters <- par(mar=c(1,1,1,1))
index <- paste(1:length(colors()))
HexAndDec <- unlist( lapply(colors(), GetColorHexAndDecimal) )
PerColumn <- 50
PerPage <- 2*PerColumn
for (page in 1: (trunc( (length(colors()) + (PerPage-1)) / PerPage) ) )
{plot(0, type="n", ylab="", xlab="",
axes=FALSE, ylim=c(PerColumn,0), xlim=c(0,1))
title("Colori di R con nome e codice esadecimale")
mtext(paste("pag.", page, "/ 7"), SOUTH<-1, adj=1, line=-1)
base <- PerPage*(page-1)
remaining <- length(colors()) - base
ColumnSize <- ifelse(remaining < PerColumn, remaining, PerColumn)
```

```
rect(0.00, 0:(ColumnSize-1), 0.49, 1:ColumnSize, border="black", col=colors()[(base+1):
(base+ColumnSize)])
text(0.045, 0.45+(0:(ColumnSize-1)), adj=1, index[(base+1):(base+ColumnSize)],
cex=0.6, col=TextContrastColor[(base+1):(base+ColumnSize)])
text(0.06, 0.45+(0:(ColumnSize-1)), adj=0, colors()[(base+1):(base+ColumnSize)],
cex=0.6, col=TextContrastColor[(base+1):(base+ColumnSize)])
save <- par(family="mono") # use mono-spaced font with number columns
text(0.25, 0.45+(0:(ColumnSize-1)), adj=0, HexAndDec[(base+1):(base+ColumnSize)],
cex=0.6, col=TextContrastColor[(base+1):(base+ColumnSize)])
par(save)
if (remaining > PerColumn)
{remaining <- remaining - PerColumn
ColumnSize <- ifelse(remaining < PerColumn, remaining, PerColumn)
rect(0.51, 0:(ColumnSize-1), 1.00, 1:ColumnSize, border="black", col=colors()
[(base+PerColumn+1):(base+PerColumn+ColumnSize)])
text(0.545, 0.45+(0:(ColumnSize-1)), adj=1, index[(base+PerColumn+1):
(base+PerColumn+ColumnSize)], cex=0.6, col=TextContrastColor[(base+PerColumn+1):
(base+PerColumn+ColumnSize)])
text(0.56, 0.45+(0:(ColumnSize-1)), adj=0, colors()[(base+PerColumn+1):
(base+PerColumn+ColumnSize)], cex=0.6, col=TextContrastColor[(base+PerColumn+1):
(base+PerColumn+ColumnSize)])
save <- par(family="mono")
text(0.75, 0.45+(0:(ColumnSize-1)), adj=0, HexAndDec[(base+PerColumn+1):
(base+PerColumn+ColumnSize)], cex=0.6, col=TextContrastColor[(base+PerColumn+1):
(base+PerColumn+ColumnSize)])
par(save)}}
par(oldparameters)
dev.off()
#
```

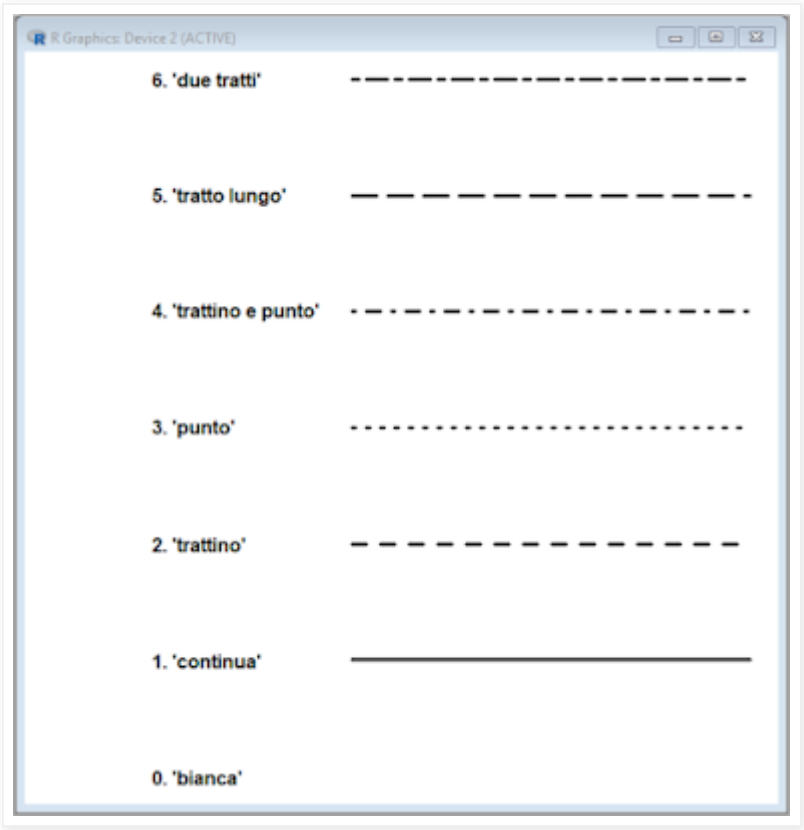
---

[1] Earl F. Glynn. *Color Chart in R*. Pubblicato su GitHub, URL consultato il 05/01/2023: https://bit.ly/2WJrJrR

domenica 18 novembre 2018

Cambiare gli stili delle linee di R

Se in un grafico volete riportare più rette, spezzate o curve, potete - con l'argomento **lty=***num* dove *num* è un numero compreso tra **0** e **6** - impiegare diversi stili della linea, con i tratteggi riportati in questa immagine (lo stile di default è la linea continua **1**):



Questo script, tratto e adattato da [1], crea la funzione **StiliDelleLinee()** che genera l'immagine con **gli stili delle linee di R**.

Per eseguire lo script copiatelo, incollatelo nella `Console di R` e premete `↵ Invio`.

```
# GLI STILI DELLE LINEE DI R
#
StiliDelleLinee <-function() # crea la funzione StiliDelleLinee()
{
oldPar <-par()
par(font=2, mar=c(0, 0, 0, 0))
plot(1, pch="", ylim=c(0, 6), xlim=c(0, 0.7), axes=FALSE, xlab="", ylab="")
for(i in 0:6) lines(c(0.3, 0.7), c(i, i), lty=i, lwd=3)
text(rep(0.1, 6), 0:6, adj = 0, labels=c("0. 'bianca'", "1. 'continua'", "2. 'trattino'", "3. 'punto'", "4. 'trattino e punto'", "5. 'tratto lungo'", "6. 'due tratti'"))
par(mar=oldPar$mar, font=oldPar$font )
}
#
StiliDelleLinee() # esegue la funzione che mostra gli stili delle linee di R
#
```

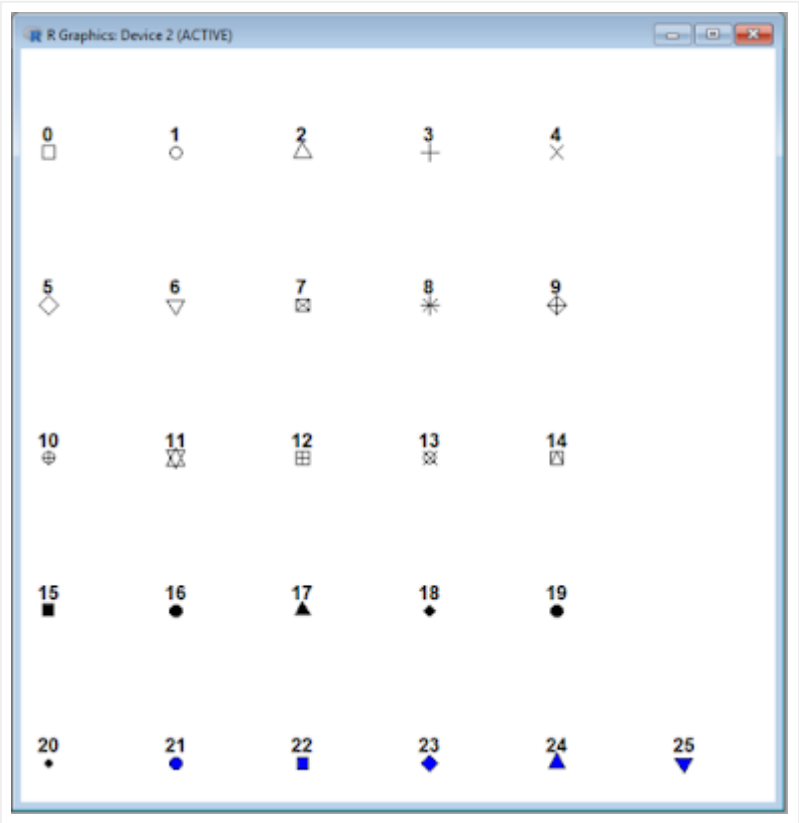
---

[1] STDHA. *Statistical tools for high-throughput data analysis*. URL consultato il 05/01/2023: https://goo.gl/Ztvcih

domenica 18 novembre 2018

Cambiare i simboli dei punti di R

Nei grafici potete - con l'argomento **pch=***num* ove *num* è un numero compreso tra **0** e **25** - riportare i punti con uno dei simboli riportati in questa immagine (il simbolo di default è il cerchio vuoto **1**).



Questo script, tratto e adattato da [1], crea la funzione **SimboliDeiPunti()** che genera l'immagine con **i simboli dei punti di R**.

Per eseguire lo script copiatelo, incollatelo nella `Console di R` e premete `↵ Invio`.

```
# I SIMBOLI DEI PUNTI DI R
#
SimboliDeiPunti <-function() # crea la funzione SimboliDeiPunti()
{
oldPar<-par()
par(font=2, mar=c(0.5,0,0,0))
y=rev(c(rep(1,6), rep(2,5), rep(3,5), rep(4,5), rep(5,5)))
x=c(rep(1:5, 5), 6)
plot(x, y, pch=0:25, cex=1.5, ylim=c(1, 5.5), xlim=c(1, 6.5), axes=FALSE, xlab="", ylab="", bg="blue")
text(x, y, labels=0:25, pos=3)
par(mar=oldPar$mar, font=oldPar$font )
}
#
SimboliDeiPunti() # esegue la funzione che mostra i simboli dei punti di R
#
```

---

[1] STDHA. *Statistical tools for high-throughput data analysis*. URL consultato il 05/01/2023: https://goo.gl/AkBdK3

lunedì 19 novembre 2018

Eseguire uno script

In **R** uno **script** è costituito semplicemente da una serie di righe di testo: scritte in linguaggio **R** queste righe di testo, riportate nella `Console di R`, sono eseguite in sequenza dall'`Interprete di comandi di R` allo scopo di elaborare dei dati o effettuare delle rappresentazioni grafiche.

Nei post di questo blog il testo degli script è riportato con queste convenzioni:

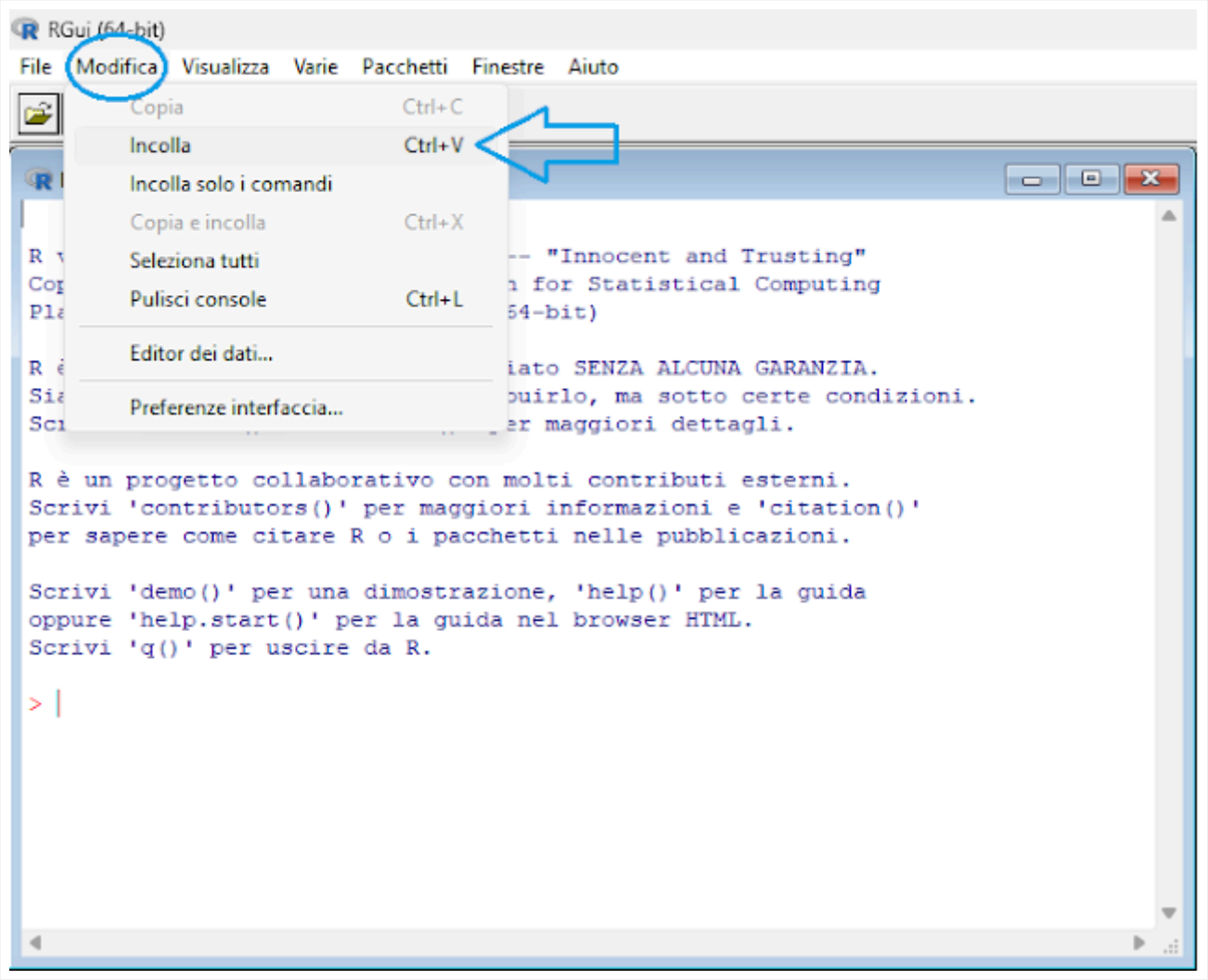
→ in colore nero preceduti dal simbolo # sono riportati i promemoria e i commenti che non sono eseguiti da **R**;
→ **in colore rosso** e in **grassetto** è riportato il codice **R** che viene eseguito e commentato;
→ **in colore nero** e in **grassetto** è riportato il codice **R** che viene eseguito ma che è già stato illustrato o riveste scarso interesse nell'esempio specifico;

mentre i risultati che compaiono nella `Console di R` in seguito all'esecuzione del codice sono riportati con queste convenzioni;

→ in `colore rosso` sono riportati sia il codice **R** inserito sia gli eventuali promemoria e commenti;
→ in `colore blu` sono riportati i **risultati** dell'elaborazione da parte di **R**.

Nel caso di uno **script riportato in un post** per eseguire lo script è sufficiente copiarlo quindi incollarlo nella `Console di R` in uno di questi modi.

Primo modo (tradizionale):
→ selezionate il testo dello script dal primo all'ultimo `#` (inclusi);
→ con il tasto destro del mouse aprite il menù contestuale e con il tasto sinistro fate `click` su `Copia` per copiare;
→ nella `RGui` fate `click` con il tasto sinistro del mouse sul menù `Modifica` quindi selezionate `Incolla` e se necessario premete ↵ `Invio`.



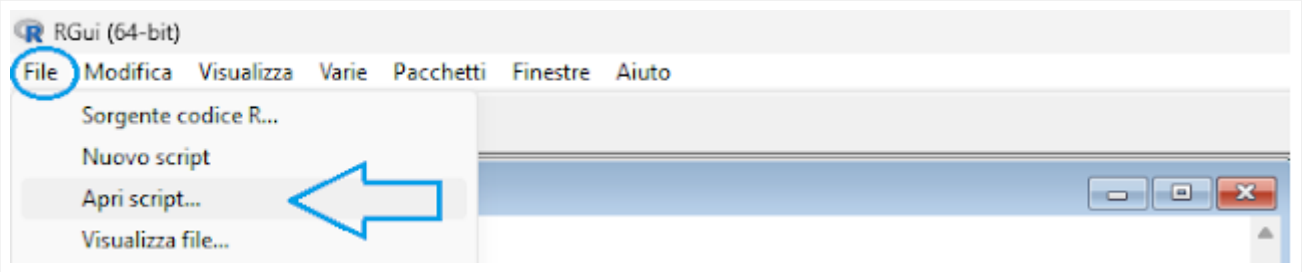
Secondo modo (alternativo):
→ selezionate il testo dello script dal primo all'ultimo `#` (inclusi);
→ con il tasto destro del mouse aprite il menù contestuale e con il tasto sinistro fate `click` su `Copia` per copiare;
→ nella `Console di R` fate `click` con il tasto destro del mouse per aprire il menù contestuale e con il tasto sinistro fate `click` su `Incolla` e se necessario premete `↵ Invio`.

Terzo modo (rapido e consigliato):
→ selezionate il testo dello script dal primo all'ultimo `#` (inclusi) e fate `ctrl-C` (tenendo premuto il tasto `ctrl` premete il tasto con la lettera `C`) per copiare;
→ fate `click` con il tasto sinistro del mouse sulla finestra della `Console di R` quindi fate `ctrl-V` (tenendo premuto il tasto `ctrl` premete il tasto con la lettera `V`) per incollare e se necessario premete `↵ Invio`.

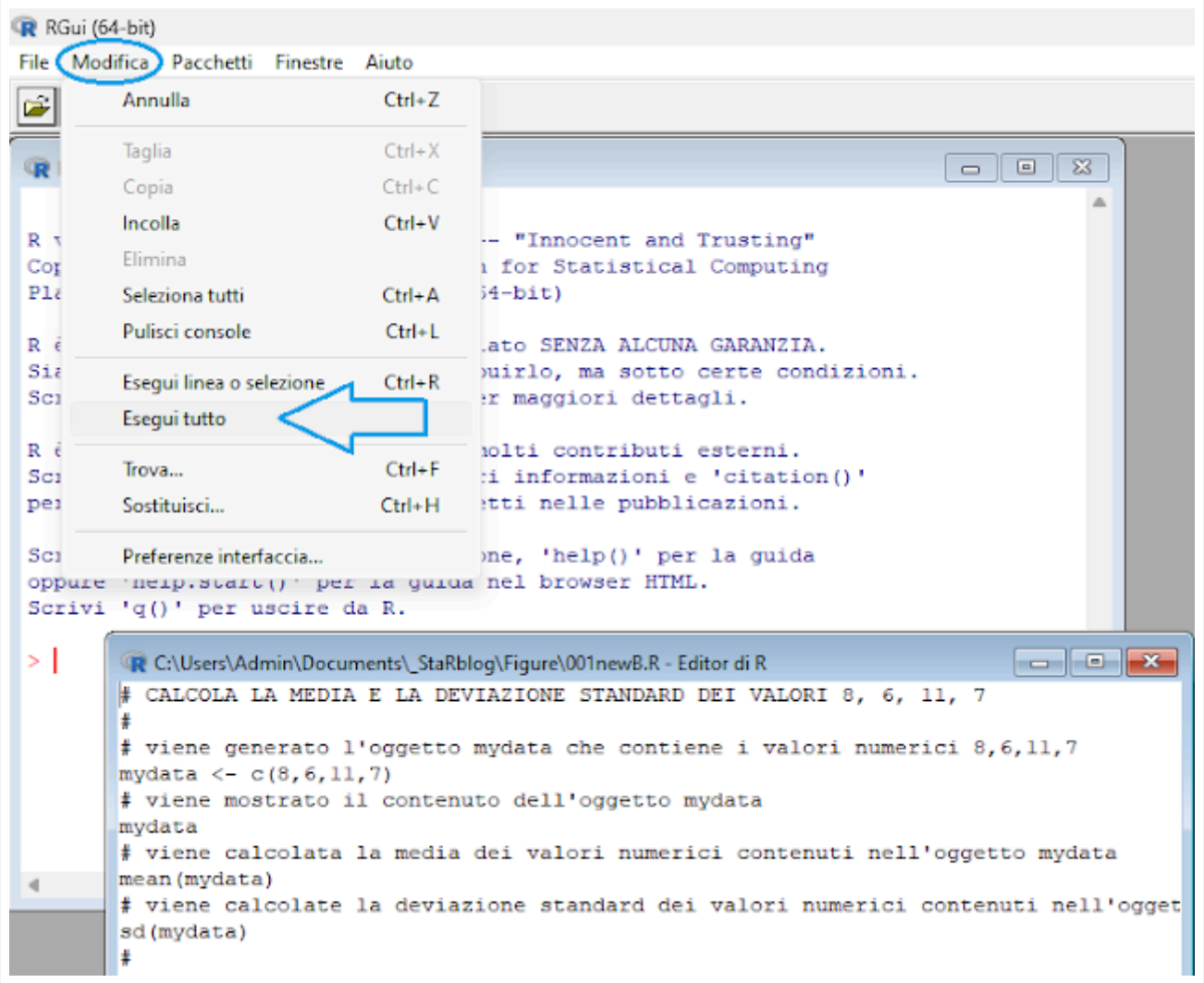
Nota: se lo script è molto lungo per selezionare e copiare il testo può essere comodo:
→ fare `click` con il tasto sinistro del mouse a sinistra del primo carattere della prima riga dello script (il più delle volte un `#`) quindi rilasciare il tasto;
→ ruotando la rotella del mouse portarsi alla fine dello script;
→ tenendo premuto il tasto `↑ Shift` (`↑ Maiuscolo`) fare `click` con il tasto sinistro del mouse a destra dell'ultimo carattere dell'ultima riga dello script;
→ con il tasto destro del mouse aprire il menù contestuale e con il tasto sinistro fare `click` su `Copia` per copiare.

Nel caso di uno **script salvato sul PC** [1]:

→ se l'avete salvato in un file di testo `.txt` dovete aprire il file con il vostro editor di testo preferito, copiare lo script con gli strumenti che questo vi fornisce, quindi incollarlo con una delle modalità riportate sopra nella `Console di R` e se necessario premete `↵ Invio`;
→ se l'avete salvato con **R** in un file `.R` dovete dal menù `File` della `RGui` selezionare `Apri script...` per aprirlo nell'`Editor di R`



quindi dal menù `Modifica` selezionare `Esegui tutto` per eseguire lo script.



Ora copiate e incollate questo script nella `Console di R` seguendo una delle indicazioni riportate e se necessario premete `↵ Invio`:

```
# CALCOLA LA MEDIA E LA DEVIAZIONE STANDARD DEI VALORI 8, 6, 11, 7
#
# viene generato l'oggetto mydata che contiene i valori numerici 8,6,11,7
mydata <- c(8,6,11,7)
# viene mostrato il contenuto dell'oggetto mydata
mydata
```

```
# viene calcolata la media dei valori numerici contenuti nell'oggetto mydata
mean(mydata)
# viene calcolate la deviazione standard dei valori numerici contenuti nell'oggetto mydata
sd(mydata)
#
```

Le righe di codice vengono eseguite una ad una in sequenza e questo è quanto compare nella `Console di R`:

```
> # CALCOLA LA MEDIA E LA DEVIAZIONE STANDARD DEI VALORI 8, 6, 11, 7
> #
> # viene generato l'oggetto mydata che contiene i valori numerici 8,6,11,7
> mydata <- c(8,6,11,7)
> # viene mostrato il contenuto dell'oggetto mydata
> mydata
[1] 8 6 11 7
> # calcola la media dei valori numerici contenuti nell'oggetto mydata
> mean(mydata)
[1] 8
> # calcola la deviazione standard dei valori numerici contenuti nell'oggetto mydata
> sd(mydata)
[1] 2.160247
> #
```

Dal punto di vista della statistica niente di più banale. Ma lo script ci offre l'opportunità per alcuni chiarimenti importanti sugli elementi del linguaggio **R**, utili a chi affronta per la prima volta l'argomento:
→ **mydata** è un **oggetto**;
→ **<-** è l'**operatore di assegnamento**;
→ **c()**, **mean()** e **sd()** sono **funzioni**;
→ l'operatore di assegnamento **<-** assegna all'oggetto **mydata** il contenuto dell'**array** (o **vettore**) generato dalla funzione **c()**;
→ l'oggetto **mydata** è l'**argomento** delle funzioni **mean()** e **sd()**;
→ la funzione **mean()** utilizza come argomento l'oggetto **mydata** che contiene i valori **8,6,11,7** per calcolarne la media, che risulta uguale a `8`;
→ la funzione **sd()** utilizza come argomento l'oggetto **mydata** che contiene i valori **8,6,11,7** per calcolarne la deviazione standard, che risulta uguale a `2.160247`.

Il nome **mydata** attribuito all'**oggetto** che contiene i numeri **8,6,11,7** è arbitrario, possiamo chiamarlo in qualsiasi altro modo purché sia un identificativo univoco dell'oggetto. Da notare che **R** riconosce le lettere maiuscole per cui **mydata** è un oggetto diverso da **Mydata**.

Come vedete nello script, digitando nella `Console di R` il nome di un oggetto come ad esempio

mydata

ne viene mostrato il contenuto

```
> mydata
[1] 8 6 11 7
```

mentre con la funzione **str()** come ad esempio con

str(mydata)

ne viene mostrato la struttura

```
> str(mydata)
 num [1:4] 8 6 11 7
```

In questo caso la struttura di **mydata** è semplice, si tratta di un oggetto che contiene un **array** (**vettore**) numerico (`num`) che include quattro valori (`[1:4]`) successivamente riportati (`8 6 11 7`), ma la funzione **str()** diventa particolarmente utile quando si salvano i molti risultati forniti da una funzione complessa in un oggetto dal quale si vuole estrarre qualche risultato specifico per una successiva elaborazione.

La funzione **c()** combina gli elementi numerici (**8,6,11,7**) inseriti come argomento in una sequenza di celle ciascuna della quali contiene un dato. In informatica si parla di **array** o **vettore** nel caso di dati monodimensionali disposti su una sola riga

8	6	11	7

si parla di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne [2]

8	9	15	14
6	7	18	12
11	8	17	13
7	4	19	17

e si parla di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

Se nella `Console di R` digitate **help(c)** si apre la finestra di aiuto con la documentazione della funzione **c()**.

All'interno della parentesi graffa `{}` compare il nome del pacchetto nel quale è inclusa la funzione, nel caso specifico la funzione **c()** è inclusa nel pacchetto `base` di funzioni che viene installato con il programma **R**.

```
c {base} R Documentation
Combine Values into a Vector or List
Description
This is a generic function which combines its arguments.
The default method combines its arguments to form a vector ...
```

Più in generale per aprire la finestra di aiuto con la documentazione di una qualsiasi funzione è sufficiente digitare **help(*nomedellafunzione*)** nella `Console di R` e pertanto:
→ per la documentazione della funzione **mean()** digitate **help(mean)** nella `Console di R`;
→ per la documentazione della funzione **sd()** digitate **help(sd)** nella `Console di R`.

Nota bene: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata non solo negli script ma anche nei dati e nei testi dei post.

---

[1] Vedere il post Salvare uno script.

[2] I vettori sono matrici aventi una sola riga o una sola colonna. Una matrice con una sola riga e più colonne è detta matrice riga o vettore riga, mentre una matrice con una sola colonna e più righe è detta matrice colonna o vettore colonna.

martedì 20 novembre 2018

Codifica dei caratteri ASCII, ANSI, Unicode (UTF)

ASCII è l'acronimo di American Standard Code for Information Interchange ed è il primo standard introdotto nella codifica dei caratteri di scrittura nel campo dell'IT (Information Techology). Il **codice ASCII** è definito nella norma *ISO/IEC 646 - Information technology - ISO 7-bit coded character set for information interchange* [1].

Nel codice ASCII i caratteri sono codificati con `7 bit` cosa che permette di codificare $2^7 = 128$ caratteri numerati da `0` a `127`. I concetti alla base dello standard sono semplici e pratici:
→ codificare i caratteri di controllo ovvero i comandi necessari per poter gestire da remoto una stampante (originariamente una telescrivente);
→ codificare i caratteri stampabili - lettere, numeri, segni di interpunzione e simboli - di impiego più frequente nella scrittura di un testo.

I caratteri da `0` a `31` sono caratteri di controllo e non sono stampabili. Il carattere `32` corrisponde a uno spazio. I caratteri da `33` a `126` sono caratteri stampabili. Il carattere `127` corrisponde a `Delete` (ed è equivalente a `Backspace`).

Questo è il set di caratteri ASCII standard altrimenti noto come **set di caratteri ASCII di base**:

Codice	Simbolo	Codice	Carattere	Codice	Carattere	Codice	Carattere
0	NUL	32		64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	TAB	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	

Questa è la versione sintetica della tabella precedente, con i soli caratteri ASCII di base stampabili:

```
!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

Di fatto il set di caratteri ASCII di base riflette il modo in cui i caratteri sono scritti su (e possono essere riletti da) un foglio di carta scritto con una macchina da scrivere, impiegando un set di caratteri/codice universalmente riconosciuto. Non solo: della macchina da scrivere il codice ASCII conserva, ad esempio, il comando `LF` (Line Feed) che fa avanzare il foglio alla riga successiva, e il comando `CR` (Carriage Return) che riporta il carrello della macchina da scrivere (virtuale) all'inizio della riga. Anche se la macchina da scrivere è stato oramai da decenni sostituita da PC, word processor e stampante, è da questa che sono derivati i concetti e questo *standard minimo di base*.

Gli altri set di caratteri che vediamo ora hanno progressivamente esteso il set di caratteri ASCII di base con set di caratteri aggiuntivi, con l'obiettivo di includere tutti i caratteri di tutte le lingue (cosa inimmaginabile via hardware, con un dispositivo meccanico come la macchina da scrivere, ma senza sostanziali problemi via software).

Il primo nato allo scopo di estendere il set di caratteri ASCII di base è stato il **set di caratteri ANSI** (acronimo di American National Standard Institute) nel quale i caratteri sono codificati con `8 bit` cosa che permette di codificare $2^8 = 256$ caratteri numerati da `0` a `255`. Il set di caratteri ANSI include il set di caratteri ASCII di base (caratteri numerati da `0` a `127`) più il set di caratteri denominato ASCII esteso (caratteri numerati da `128` a `255`). Purtroppo per i caratteri da `128` a `255` non si è stabilito subito un accordo e questo ha determinato la presenza di differenze tra il set di caratteri a 8 bit *Windows 1252* (che viene denominato impropriamente *ANSI* e che è ampiamente utilizzato) e il set di caratteri *ISO-8859-1* anch'esso a 8 bit (i due infatti codificano in modo differente i caratteri da 128 a 159).

Dato il numero elevato di caratteri da codificare (si pensi ad esempio al cirillico, al greco, all'aramaico, all'ebraico e al numero di ideogrammi necessari per codificare i testi in Cinese, Giapponese e Coreano) e data la vitale importanza di una codifica univoca dei caratteri, è nato quindi **Unicode**, un consorzio no-profit privato al quale partecipano tutte le principali aziende di informatica, con lo scopo di assegnare ad ogni carattere impiegato per la scrittura un numero univoco indipendente dalla lingua, dalla piattaforma informatica e dal software impiegati [2].

Gli standard Unicode sono identificati dalla sigla **UTF** (Unicode Transformation Format) seguita da un numero progressivo di versione. Dal 1991 il Consorzio Unicode ha sviluppato gli standard in parallelo con la norma *ISO/IEC 10646* che definisce l'Universal Coded Character Set (**UCS**). Entrambi hanno lo stesso repertorio di caratteri, con gli stessi numeri: fortunatamente, e saggiamente, Unicode e ISO hanno deciso di normare congiuntamente in modo identico questo aspetto di base e così delicato della comunicazione.

Di fatto Unicode è oggi il riferimento ufficiale - per definizione in linea con la normativa ISO - per la codifica standard dei caratteri, che include gli storici set di caratteri
→ codice ASCII di base - *Basic Latin* [3]
→ codice ASCII esteso - *Latin-1 Supplement* [4]
accanto agli altri numerosissimi set di caratteri che nel frattempo sono stati sviluppati, e continuano ad essere aggiornati, per coprire tutte le lingue del mondo [5] .

Attualmente il più estesamente impiegato è lo **standard UTF-8**: alla data di pubblicazione di questo post la maggior parte delle pagine web sono realizzate impiegando la codifica dei caratteri prevista da questo standard.

---

[1] *ISO/IEC 646*. URL consultato il 29/09/2023: https://goo.gl/hAuERi

[2] *The Unicode Consortium*. URL consultato il 29/09/2023: https://home.unicode.org/

[3] *C0 Controls and Basic Latin*. URL consultato il 29/09/2023: https://goo.gl/GJHqfW

[4] *C1 Controls and Latin-1 Supplement*. URL consultato il 29/09/2023: https://goo.gl/cfhgLd

[5] *Unicode Character Code Charts*. URL consultato il 29/09/2023: https://goo.gl/rsFsgv

martedì 20 novembre 2018

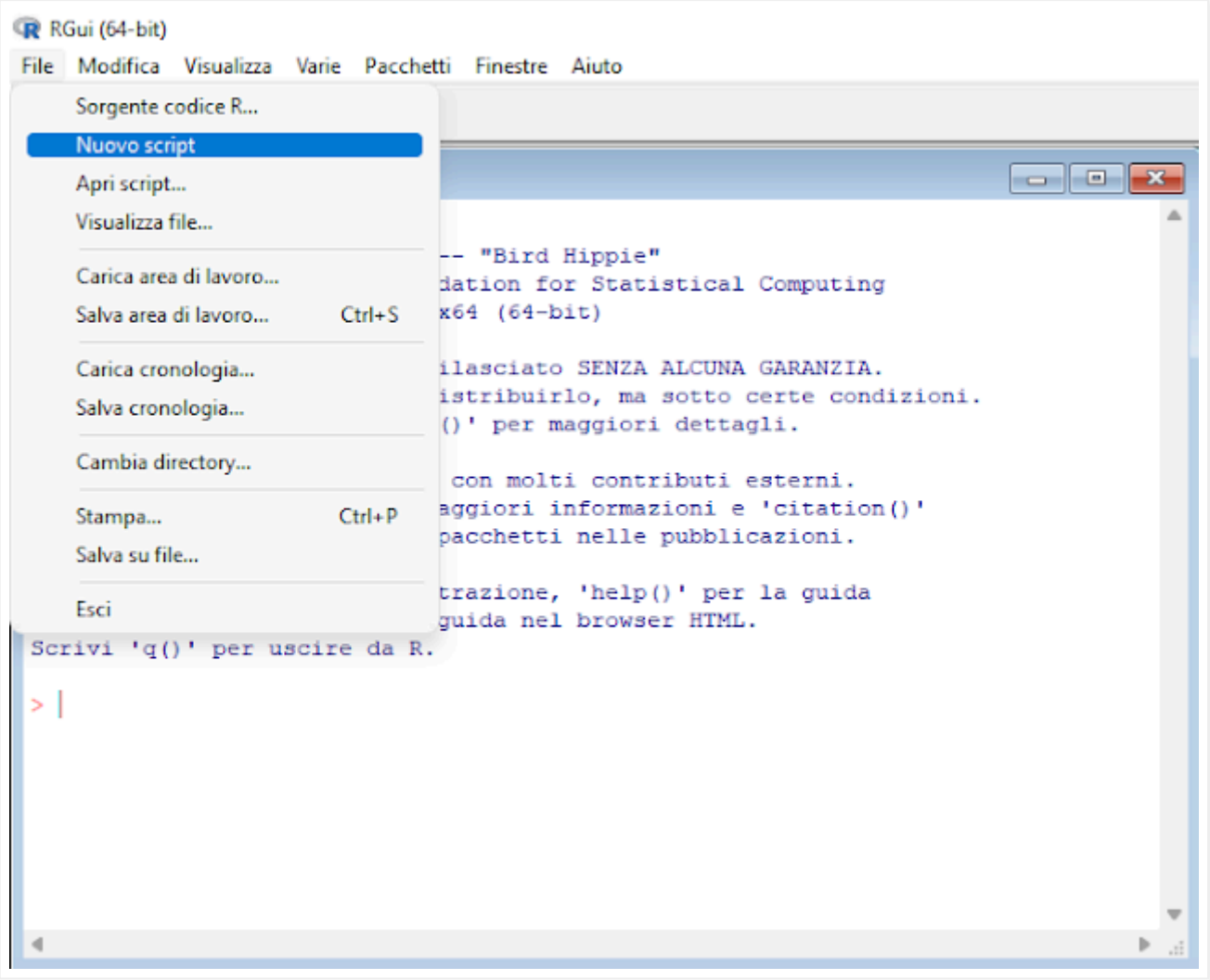
Salvare uno script

Riprendiamo lo script per il calcolo della media e della deviazione standard che abbiamo impiegato per illustrare come eseguire uno script [1], questa volta con l'obiettivo di salvarlo sotto forma di file per iniziare a realizzare una raccolta di script da riutilizzare al bisogno.

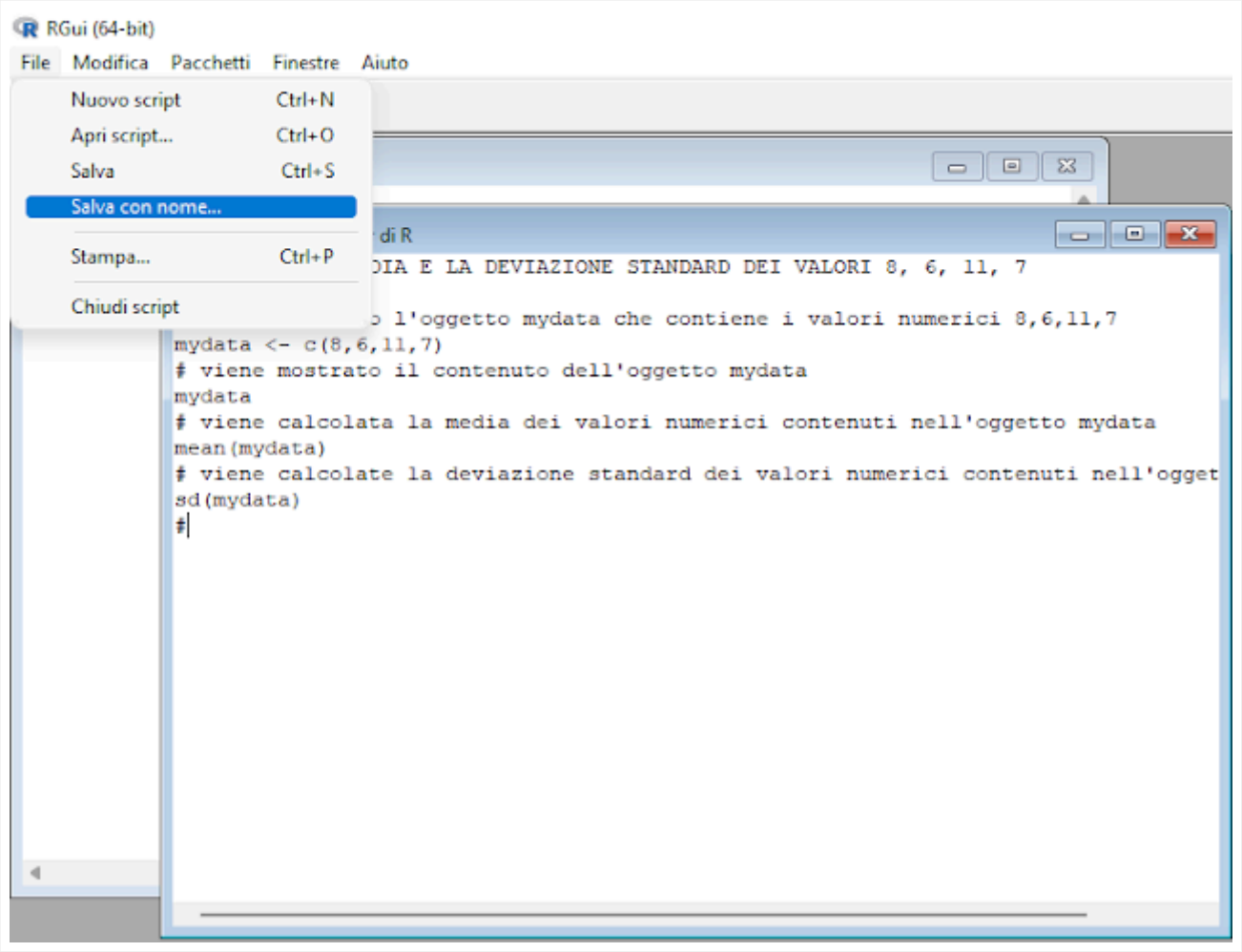
```
# CALCOLA LA MEDIA E LA DEVIAZIONE STANDARD DEI VALORI 8, 6, 11, 7
#
# viene generato l'oggetto mydata che contiene i valori numerici 8,6,11,7
mydata <- c(8,6,11,7)
# viene mostrato il contenuto dell'oggetto mydata
mydata
# viene calcolata la media dei valori numerici contenuti nell'oggetto mydata
mean(mydata)
# viene calcolate la deviazione standard dei valori numerici contenuti nell'oggetto mydata
sd(mydata)
#
```

Ci sono vari modi in cui potete farlo. Ma tutti prevedono di salvare il codice **R** in un **file in formato testo** [2] impiegando una codifica universalmente riconosciuta [3].

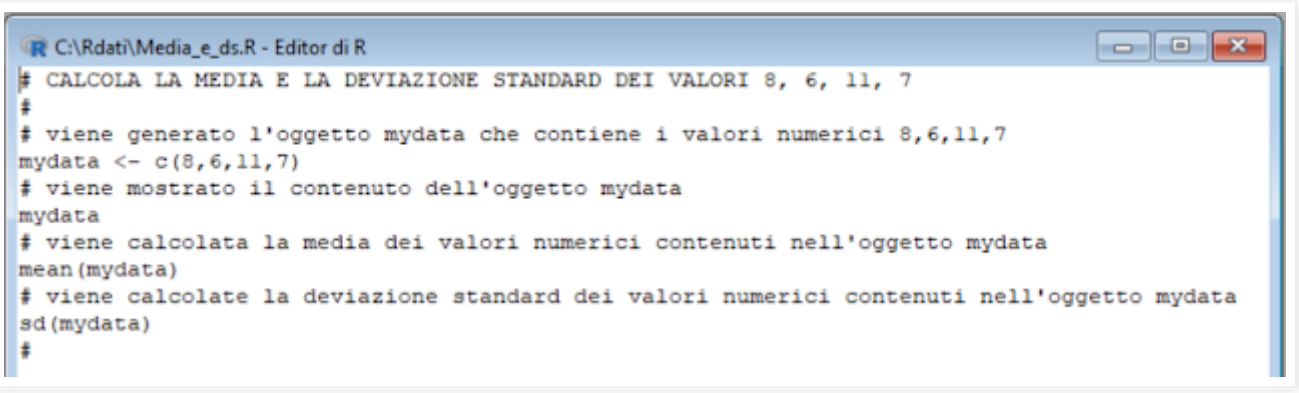
Un primo modo prevede di utilizzare l'`Editor di R` che si apre dalla `RGui` selezionando `File >> Nuovo script`



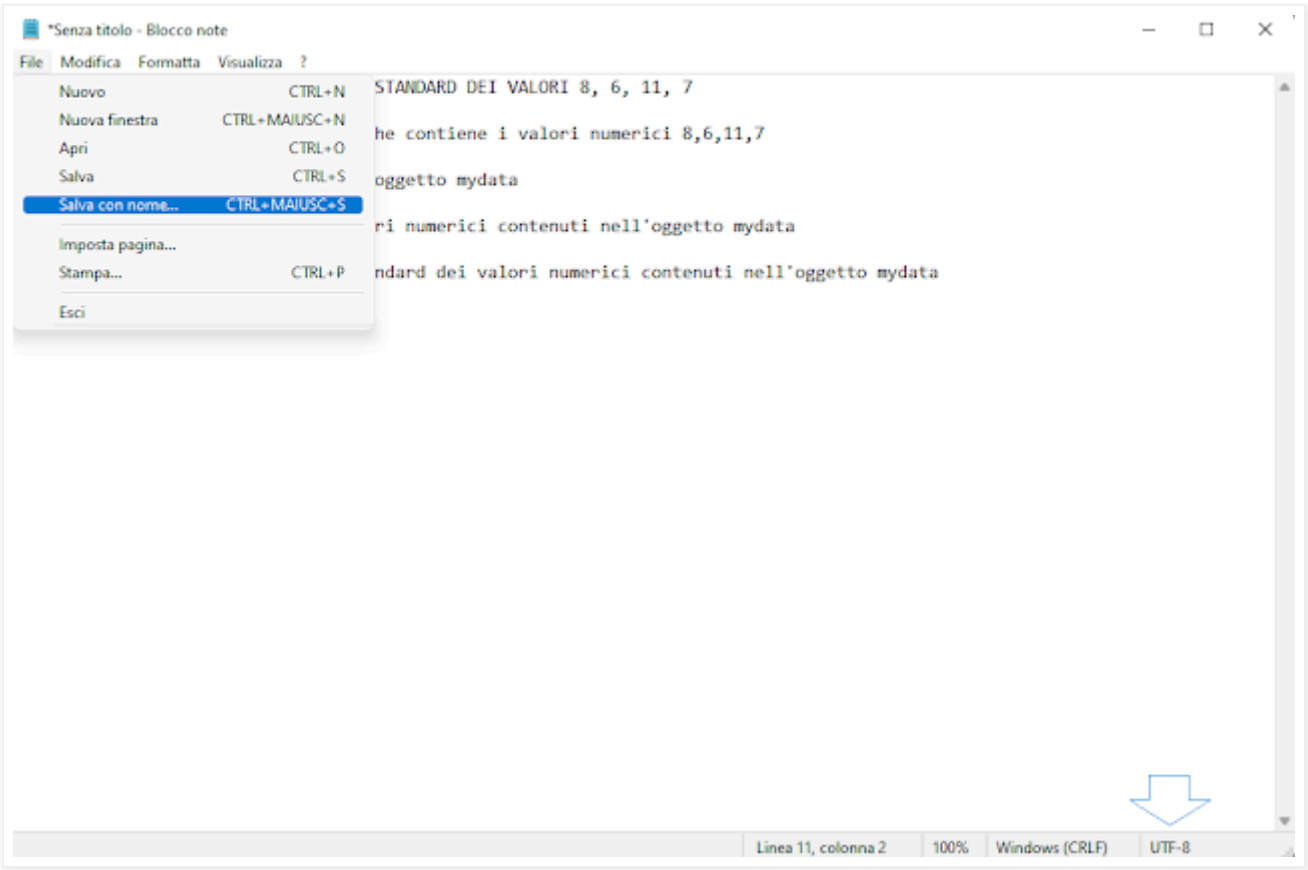
Copiate il codice dello script, incollatelo nell'`Editor di R` quindi selezionate `File >> Salva con nome`



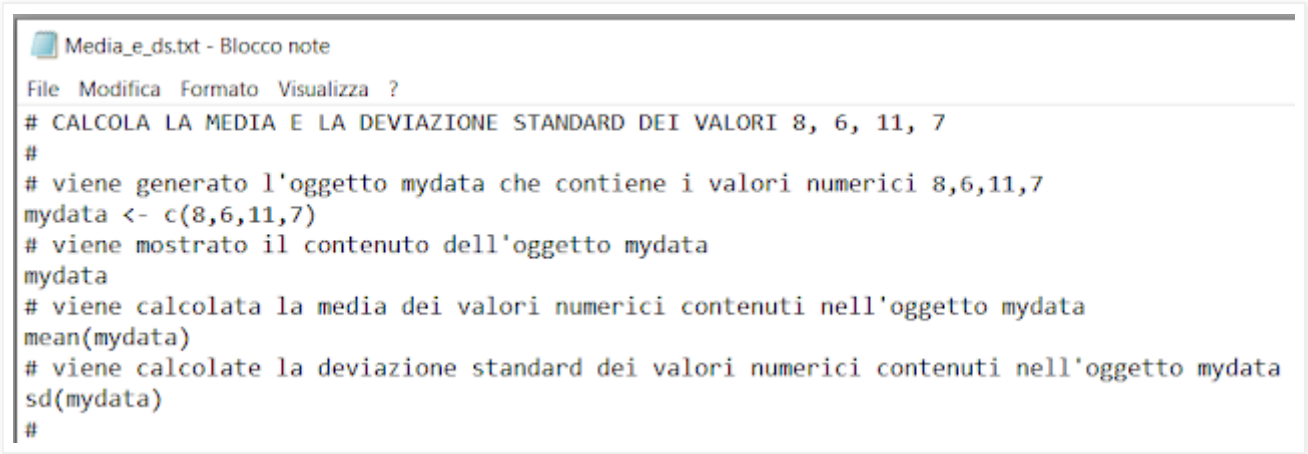
e salvate il file per esempio con il nome `Media_e_ds.R` (l'estensione `.R` viene aggiunta automaticamente dall'`Editor di R`)



Un modo alternativo prevede di utilizzare uno degli strumenti forniti con il sistema operativo, come ad esempio il `Blocco note` di Windows. Dopo avere incollato il codice nel `Blocco note` selezionate `File >> Salva con nome`



e salvate il file per esempio con il nome `Media_e_ds.txt` (l'estensione `.txt` viene aggiunta automaticamente)



Nel `Blocco note` quando selezionate `File >> Salva con nome` dovete verificare che vicino al tasto `Salva` sia selezionato `Codifica: UTF-8` cioè la codifica dei caratteri impiegata di default da **R** [3]. Nella penultima figura si vede in basso a destra nel `Blocco note` di Windows la scritta UTF-8 che conferma che il salvataggio avviene con questa codifica dei caratteri.

In conclusione l'`Editor di R` aggiunge automaticamente al nome del file l'estensione `.R` mentre il `Blocco note` di Windows aggiunge automaticamente al nome del file l'estensione `.txt`: ma si tratta di file di testo perfettamente identici tanto che è possibile aprire i file `.R` con il `Blocco note` di Windows e aprire i file `.txt` con l'`Editor di R` (provare per credere).

Potete ancora impiegare non solo un qualsiasi altro editor di testo ma anche un qualsiasi `wordprocessor`, salvando i file `in formato testo` (tutti i wordprocessor hanno questa opzione per il salvataggio dei file) con l'accortezza di verificare che siano salvati con la codifica `UTF-`

`8`, cosa che attualmente dovreste trovare prevista di default con i wordprocessor di tutte le piattaforme (Windows, Mac, Linux).

Quale di questi metodi usare allora? Personalmente uso il `Blocco note` di Windows ma in realtà non fa nessuna differenza: è possibile salvare gli script con il programma che si trova più comodo e più pratico utilizzare. La sola cosa che conta è salvarli in un **file in formato testo** perché questo è il solo formato con cui lo script può essere riconosciuto ed essere eseguito in **R**.

Per il seguito si rimanda nuovamente a [1].

Nota bene: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata non solo negli script ma anche nei dati e nei testi dei post.

---

[1] Vedere il post Eseguire uno script.

[2] Cioè in un file nel quale i caratteri possono essere scritti (e successivamente essere riletti) in chiaro nello stesso modo in cui sono scritti su (e possono essere riletti da) un foglio di carta scritto con una macchina da scrivere, impiegando un set di caratteri/codice universalmente riconosciuto. La macchina da scrivere è stata oramai da decenni sostituita da PC, wordprocessor e stampante, ma è da lei che è derivato il concetto.

[3] Vedere il post Codifica dei caratteri ASCII, ANSI, Unicode (UTF).

domenica 25 novembre 2018

Inserimento manuale dei dati [1]

Inserire a mano i dati in **R** non accade di frequente, ma è utile quando i dati da inserire sono pochi, come ad esempio quando si vuole effettuare un test chi-quadrato [1], impiegare il teorema di Bayes [2] o realizzare dei grafici a torta [3].

Per questo ho predisposto due esempi che illustrano la sintassi da utilizzare per inserire direttamente da tastiera array (vettori) e combinarli in matrici assegnando i nomi alle variabili e ai casi [4].

Il primo esempio genera un **vettore** (array), lo trasforma in una **matrice**, assegna un nuovo nome alla variabile/colonna e infine assegna un nuovo descrittore a ciascuno dei casi/righe.

Per eseguire lo script copiatelo quindi incollatelo nella `Console di R` e premete `↵ Invio`.

```
# GENERA UN ARRAY E LO TRASFORMA IN UNA MATRICE
#
x <- c(4.2, 6.8, 2.5, 8.3, 5.4, 7.9, 5.3, 6.7, 2.2, 3.1) # genera l'array x
x # mostra l'array x
mean(x) # calcola la media
#
mymatrix <- data.frame(x) # trasforma l'array x in una matrice
mymatrix # mostra la matrice con i casi/righe identificati automaticamente da R
mean (mymatrix$x) # calcola la media
#
names(mymatrix) <- c("Variabile_1") # assegna un nuovo nome alla variabile/colonna
mymatrix # mostra la matrice con il nuovo nome della variabile/colonna
mean(mymatrix$Variabile_1) # calcola la media richiamando il nome della variabile/colonna
#
row.names(mymatrix) <- c("Riga_uno", "Riga_due", "Riga_tre", "Riga_quattro",
"Riga_cinque", "Riga_sei", "Riga_sette", "Riga_otto", "Riga_nove", "Riga_dieci") #
sostituisce gli identificativi numerici di riga di R con nuovi descrittori univoci dei casi/righe
mymatrix # mostra la matrice con i nuovi descrittori dei casi/righe
mean(mymatrix[,1]) # calcola la media richiamando il numero della variabile/colonna
#
```

Utilizzate i tasti `Pag-su` e `Pag-giù` per scorrere nella finestra della `Console di R` quanto è accaduto, che viene illustrato dai commenti inseriti in ciascuna riga.

Da notare come, quando l'array **x** viene trasformato nella matrice **mymatrix** mediante la funzione **data.frame()** [5], ai casi/righe viene assegnato di default un identificativo numerico univoco.

A questo punto viene impiegata la funzione **names()** per assegnare un nuovo nome alla variabile. In questo modo gli identificativi numerici dei casi/righe assegnati di default sono sostituiti con dei nuovi descrittori univoci (“Riga_uno”, “Riga_due”, eccetera) impiegando la funzione **row.names()**.

Questo è quindi il contenuto dell'oggetto **mymatrix** definitivo riportato alla penultima riga di codice:

```
> mymatrix # mostra mymatrix con il nome della variabile/colonna e i descrittori dei
casi
           Variabile_1
Riga_uno           4.2
Riga_due           6.8
```

```
Riga_tre            2.5
Riga_quattro        8.3
Riga_cinque         5.4
Riga_sei            7.9
Riga_sette          5.3
Riga_otto           6.7
Riga_nove           2.2
Riga_dieci          3.1
```

Il calcolo della media è stato introdotto ogni volta per illustrare la sintassi da impiegare. Interessante l'ultimo caso, all'ultima riga di codice: qui la media è stata calcolata richiamando il numero della colonna sulla quale va calcolata, un modo interessante e utile per richiamare una variabile di una matrice o di una tabella.

```
> mean(mymatrix[,1]) # calcola la media richiamando il numero della colonna
[1] 5.24
```

In questo secondo esempio di inserimento manuale dei dati sono generati due vettori (array), che sono combinati in una **matrice** di 2 colonne per 10 righe, quindi alla matrice viene aggiunta una terza colonna/variabile. Sono poi assegnati nuovi nomi alle variabili/colonne e infine viene assegnato un nuovo descrittore univoco a ciascuno dei casi/righe.

Copiate lo script quindi incollatelo nella `Console di R` e premete `↵ Invio`.

```
# GENERA DUE ARRAY, LI COMBINA IN UNA MATRICE E AGGIUNGE UNA COLONNA
#
x <- c(4.2, 6.8, 2.5, 8.3, 5.4, 7.9, 5.3, 6.7, 2.2, 3.1) # genera l'array x
y <- c(3.1, 2.2, 6.7, 5.3, 7.9, 5.4, 8.3, 2.5, 6.8, 4.2) # genera l'array y
#
mymatrix <- data.frame(x, y) # combina gli array x e y in una matrice
mymatrix # mostra la matrice con i casi/righe identificati automaticamente da R
sapply(mymatrix, mean) # calcola la media
#
mymatrix$z <- mymatrix$x + mymatrix$y # aggiunge una nuova colonna
mymatrix # mostra la matrice con i casi/righe identificati automaticamente da R
sapply(mymatrix, mean) # calcola la media
#
names(mymatrix) <- c("Var_1", "Var_2", "Var_3") # assegna un nuovo nome alle
variabili/colonne
mymatrix # mostra la matrice con i nuovi nomi delle variabili/colonne
mean(mymatrix$Var_3) # calcola la media richiamando il nome della variabile/colonna
#
row.names(mymatrix) <- c("Caso_uno", "Caso_due", "Caso_tre", "Caso_quattro",
"Caso_cinque", "Caso_sei", "Caso_sette", "Caso_otto", "Caso_nove", "Caso_dieci") #
sostituisce gli identificativi numerici di riga di R con nuovi descrittori univoci dei casi/righe
mymatrix # mostra la matrice con i nuovi descrittori dei casi/righe
mean(mymatrix[,3]) # calcola la media richiamando il numero della variabile/colonna
#
```

Dopo avere eseguito lo script utilizzate i tasti `Pag-su` e `Pag-giù` per scorrere nella finestra della `Console di R` quanto è accaduto, che viene di nuovo illustrato dai commenti inseriti in ciascuna riga.

Da notare nuovamente come una volta combinati i due array **x** e **y** nella matrice **mymatrix** mediante la funzione **data.frame()** ai casi/righe viene assegnato di default un identificativo numerico univoco. Quindi con la funzione **sapply()** viene calcolata la media su tutte le colonne/variabili della

matrice. Questo accade anche quando viene aggiunta alla matrice una nuova colonna/variabile **z** contenente la somma della variabile **x** e della variabile **y**.

A questo punto viene impiegata la funzione **names()** per assegnare i nuovi nomi alle variabili delle due colonne mentre con la funzione mean() è possibile calcolare separatamente la media della colonna/variabile **Var_3**.

Infine gli identificativi numerici delle righe/casi assegnati da **R** sono sostituiti con dei nuovi descrittori univoci ("Riga uno", "Riga due", eccetera) impiegando la funzione **row.names()**.

Questo è quindi il contenuto dell'oggetto **mymatrix** definitivo riportato alla penultima riga di codice:

```
> mymatrix # mostra la matrice con i nuovi descrittori dei casi/righe
             Var_1 Var_2 Var_3
Caso_uno       4.2   3.1   7.3
Caso_due       6.8   2.2   9.0
Caso_tre       2.5   6.7   9.2
Caso_quattro   8.3   5.3  13.6
Caso_cinque    5.4   7.9  13.3
Caso_sei       7.9   5.4  13.3
Caso_sette     5.3   8.3  13.6
Caso_otto      6.7   2.5   9.2
Caso_nove      2.2   6.8   9.0
Caso_dieci     3.1   4.2   7.3
```

Infine l'ultima riga di codice ci ricorda la possibilità di impiegare il numero della colonna per specificare i dati sui quali effettuare il calcolo della media:

```
> mean(mymatrix[,3]) # calcola la media richiamando il numero della colonna
[1] 10.48
```

Se siete interessati al tema potrebbero esservi utili anche gli esempi riportati in:
→ Inserimento manuale dei dati [2]
→ Inserimento manuale dei dati [3]

---

[1] Vedere la sezione *Analisi di dati qualitativi* alla pagina Indice.

[2] Vedere il post Sensibilità specificità e valore predittivo.

[3] Vedere il post Grafici a torta.

[4] Parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14
6	7	18	12
11	8	17	13

7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

Di fatto i vettori sono matrici aventi una sola riga o una sola colonna. Una matrice con una sola riga e più colonne è detta matrice riga o vettore riga, mentre una matrice con una sola colonna e più righe è detta matrice colonna o vettore colonna.

[5] Digitate **help(***nomedellafunzione***)** nella `Console di R` per la documentazione di questa e delle altre funzioni qui impiegate.

lunedì 26 novembre 2018

Inserimento manuale dei dati [2]

Sapere inserire a mano i dati in **R** può essere utile. Per questo ho predisposto due esempi che illustrano la sintassi da utilizzare per inserire direttamente da tastiera array (vettori) e combinarli in matrici o tabelle [1] assegnando i nomi alle variabili e ai casi.

Questo script genera una **matrice** di due righe per due colonne. Si tratta della struttura dati tipica necessaria per effettuare il test chi-quadrato, per effettuare il test di Fisher, il test di McNemar o impiegare il teorema di Bayes [2].

Copiate lo script, incollatelo nella `Console di R` e premete `↲ Invio`.

```
# DA QUATTRO VALORI GENERA UNA MATRICE 2x2
#
cells <- c(1, 26, 24, 68) # genera l'array cells con i valori numerici contenuti nelle celle
cells # mostra l'array cells
#
rnames <- c("Riga 1", "Riga 2") # genera l'array rnames con i nomi delle righe
rnames # mostra l'array rnames
#
cnames <- c("Colonna 1", "Colonna 2") # genera l'array cnames con i nomi delle colonne
cnames # mostra l'array cnames
#
mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames,
cnames)) # costruisce la matrice a partire dagli array cells, rnames, cnames
mymatrix # mostra la matrice
#
```

Dopo avere eseguito lo script utilizzate i tasti `Pag-su` e `Pag-giù` per scorrere nella finestra della `Console di R` quanto è accaduto.

Mediante la funzione **c()** [3] sono generati gli array che mediante l'operatore di assegnamento **<-** vengono denominati **cells**, **rnames**, **cnames** e contengono rispettivamente i dati (**cells**), i nomi delle righe (**rnames**) e i nomi delle colonne (**cnames**).

Mediante la funzione **matrix()** [4] gli array sono combinati nella matrice **mymatrix** di 2 righe (**nrow=2**) per due colonne (**ncol=2**) inserendo i valori per riga (**byrow=TRUE**) e quindi da sinistra a destra e dall'alto in basso ottenendo quindi la matrice desiderata alla quale con l'ultimo argomento (**dimnames=list(rnames, cnames)**) sono assegnati i nomi alle righe e alle colonne.

Questo è il contenuto dell'oggetto **mymatrix** definitivo riportato da **R** al termine dell'elaborazione dopo l'ultimo comando **mymatrix**:

```
> mymatrix
       Colonna 1 Colonna 2
Riga 1         1        26
Riga 2        24        68
```

Quest'altro script genera una **tabella** (*dataframe*) che contiene valori numerici, alfanumerici e logici, e assegna i nomi alle variabili (colonne).

Copiatelo per intero quindi incollatelo nella `Console di R` e premete `↲ Invio`:

```
# GENERA UNA TABELLA (DATAFRAME) DI 3 COLONNE PER 4 RIGHE
#
d <- c(9901, 9902, 9903, 9904) # genera un array con quattro valori numerici
d # mostra l'array d
#
e <- c("rosso", "bianco", "blu", NA) # genera un array con quattro valori alfanumerici, NA=Not Available
e # mostra l'array e
#
f <- c(TRUE,TRUE,TRUE,FALSE) # genera un array con quattro valori logici
f # mostra l'array f
#
mytable <- data.frame(d, e, f) # genera una tabella (dataframe) a partire dagli array d, e, f
mytable # mostra il contenuto della tabella
#
names(mytable) <- c("Codice", "Colore", "Dato valido") # assegna i nomi alle variabili/colonne
row.names(mytable) <- c("A", "B", "C", "D") # sostituisce gli identificativi numerici di riga di R con nuovi descrittori univoci dei casi/righe
mytable # mostra il contenuto della tabella
#
```

Dopo avere eseguito lo script utilizzate casi i tasti `Pag-su` e `Pag-giù` per scorrere nella finestra della `Console di R` quanto è accaduto.

Qui di nuovo con la funzione **c()** sono generati gli array che mediante l'operatore di assegnamento **<-** vengono denominati **d**, **e**, **f** e che subito dopo sono combinati mediante la funzione **data.frame()** nella tabella **mytable**.

Successivamente con la funzione **names()** sono generati i nomi che vengono assegnati alle colonne/variabili della tabella e con la funzione **row.names()** sono sostituiti gli identificativi numerici assegnati di default ai casi/righe.

Questo è il contenuto dell'oggetto definitivo riportato al termine dell'elaborazione dopo l'ultimo comando **mytable**:

```
> mytable # mostra il contenuto della tabella
  Codice Colore Dato valido
A   9901  rosso        TRUE
B   9902 bianco        TRUE
C   9903    blu        TRUE
D   9904   <NA>       FALSE
```

Anche se il significato dei vari passaggi è stato illustrato con i commenti inclusi negli script, si ricordano alcune regole generali:
→ i nomi assegnati agli oggetti che vengono generati (**cells**, **rnames**, **d**, **f**, **mytable** e quant'altro) possono essere stabiliti liberamente;
→ potete stabilire liberamente anche i nomi da assegnare alle colonne/variabili e ai casi/righe;
→ per vedere il contenuto di un oggetto è sufficiente digitare nella `Console di R` il nome dell'oggetto;
→ una tabella per definizione può contenere sia valori numerici, sia valori alfanumerici (stringhe di testo), sia valori logici;
→ se un dato non è disponibile, al posto del dato va inserita la sigla **NA** (**N**ot **A**vailable).

Se siete interessati al tema potrebbero esservi utili anche gli esempi riportati in:
→ Inserimento manuale dei dati [1]
→ Inserimento manuale dei dati [3]

---

[1] Parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14
6	7	18	12
11	8	17	13
7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

[2] Vedere le rispettive voci alla pagina Indice.

[3] Digitate **help(c)** nella `Console di R` per la documentazione della funzione **c()**.

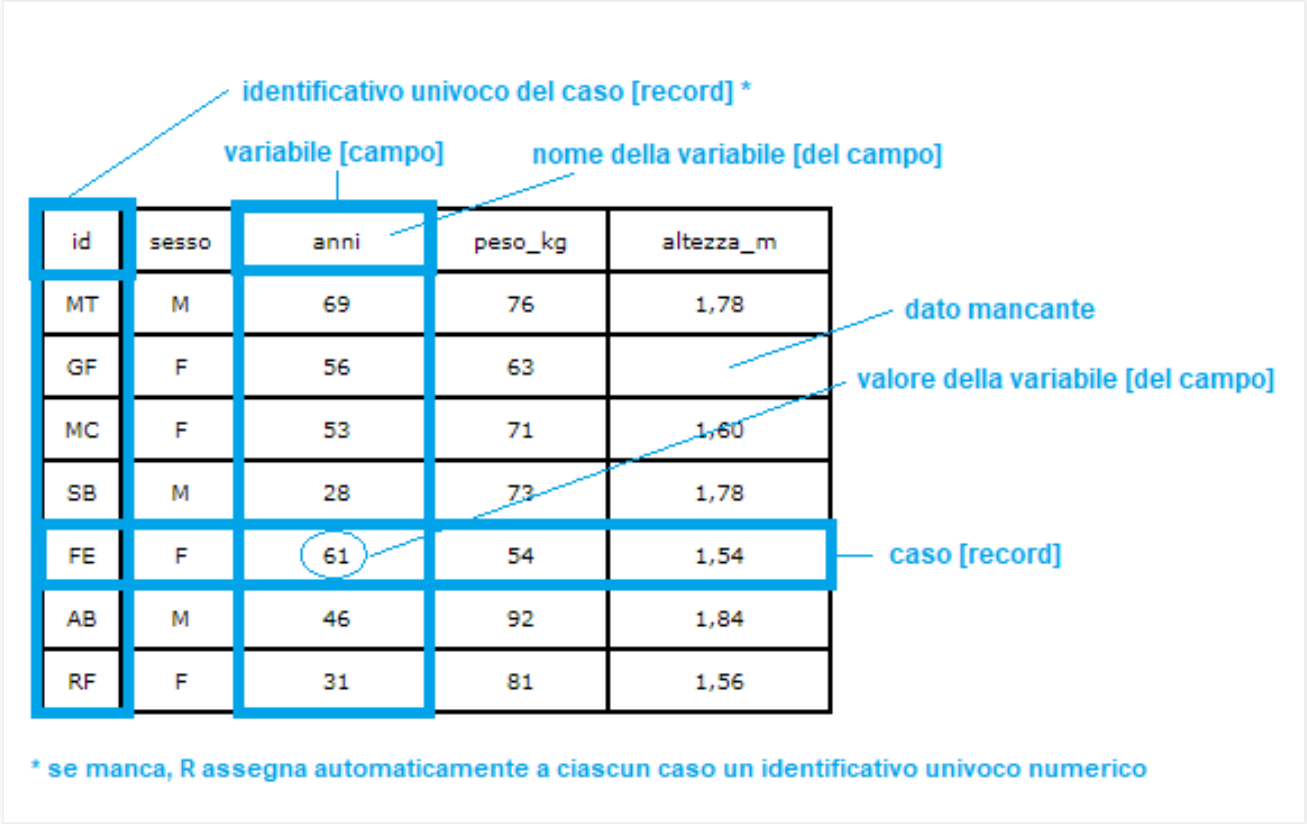
[4] Digitate **help(matrix)** nella `Console di R` per la documentazione della funzione **matrix**.

martedì 27 novembre 2018

Importazione dei dati da un file .csv

R è un programma per la **analisi statistica e grafica dei dati** e la cosa che accade con maggior frequenza è avere dei dati gestiti esternamente a **R**, in un database o in un foglio elettronico, e volerli importare in **R** per poterli analizzare.

Una struttura dati tipica, ed estesamente impiegata anche in **R** [1], è riportata in questa **tabella** (che è di fatto un minuscolo **database**)



id	sesso	anni	peso_kg	altezza_m
MT	M	69	76	1,78
GF	F	56	63	
MC	F	53	71	1,60
SB	M	28	73	1,78
FE	F	61	54	1,54
AB	M	46	92	1,84
RF	F	31	81	1,56

nella quale le cose da notare in relazione a **R** sono abbastanza semplici:
→ le **righe** corrispondono ai **casi** della statistica (e ai **record** del database);
→ le **colonne** corrispondono alle **variabili** della statistica (e ai **campi** del database);
→ i **nomi delle variabili** sono riportati nella prima riga (in **R** sono facoltativi ma fortemente raccomandati per mantenere ordine e chiarezza nel proprio lavoro);
→ le **variabili** possono essere sia numeriche, sia qualitative (per esempio qui `Sesso` è una variabile qualitativa);
→ un **identificativo univoco** di ciascuno dei **casi** può essere riportato nella prima colonna (è facoltativo). Se l'identificativo non è presente nei dati originali **R** numera automaticamente i **casi** in ordine crescente per identificare ciascun caso in modo univoco;
→ dato che come separatore delle cifre decimali **R** al proprio interno impiega esclusivamente il punto (.) se nei dati da importare viene impiegata la virgola (,) questa verrà convertita da **R** in un punto (.);
→ è possibile che si verifichi la **mancanza di dati**, per esempio qui manca il valore di `Altezza` nel caso `GF`, e il campo è quindi vuoto. **R** al momento di importare i dati riconoscerà automaticamente questi casi riportando nel campo del dato mancante la sigla `NA` ovvero `Not Available`;

Nota bene: in **R** le variabili qualitative, non numeriche, sono denominate **fattori** e sono importanti in quanto consentono di raggruppare i dati di una database in sottoinsiemi. Così i dati della nostra tabella potranno essere elaborati tutti insieme o suddivisi in due gruppi in base al valore assunto dalla variabile `Sesso`. Ovviamente è indispensabile che la variabile in base alla quale i dati possono essere

raggruppati sia codificata in modo rigoroso, così il sesso maschile, poiché **R** riconosce lettere maiuscole e lettere minuscole, deve essere espresso sempre con `M` (o con `m`) e non si possono usare `M` o `m` indifferentemente.

Il tema che si pone con **R** è ora questo: in quale formato salvare (e leggere) i dati?

Abbiamo visto altrove che il modo migliore per salvare uno script [2] è farlo in un file di testo (in genere con estensione `.txt` ma in **R** anche con estensione `.R`) nel quale i **caratteri** possono essere scritti (e successivamente essere riletti) in chiaro nello stesso modo in cui sono scritti su (e possono essere riletti da) un foglio di carta scritto con una macchina da scrivere [3], impiegando un set di caratteri limitato, ma universalmente riconosciuto, tipicamente il codice ASCII di base [4].

La stessa identica soluzione - salvarli in un file di testo nel quale sono scritti in chiaro - è raccomandata per i **dati** e impiega il formato `.csv` (l'acronimo `csv` sta per `comma separated values` cioè per valori separati dalla virgola (,) tuttavia come vedremo tra poco il separatore di campo può essere anche un altro carattere). Le modalità per salvare in formato `.csv` i dati dipendono dal programma sul quale risiedono e dal quale li volete esportare, ora invece vediamo come li possiamo leggere e importare in **R**.

Per continuare è necessario:
→ effettuare il download del file `importa_csv.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per questo e gli altri file di dati impiegati nei post trovate link e modalità di download alla pagina Dati.

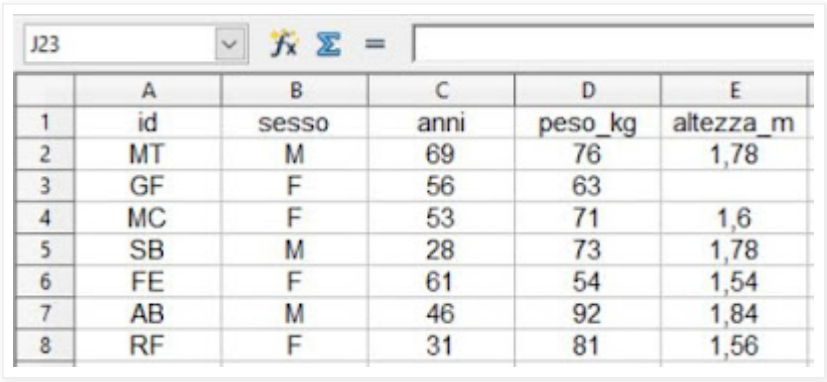
Il file, aperto con un editor di testo come ad esempio il `Blocco note di Windows`, contiene i dati della tabella/database:

```
id;sesso;anni;peso_kg;altezza_m
MT;M;69;76;1,78
GF;F;56;63;
MC;F;53;71;1,60
SB;M;28;73;1,78
FE;F;61;54;1,54
AB;M;46;92;1,84
RF;F;31;81;1,56
```

Come vedete un file `.csv` è, analogamente a un file `.txt`, un file di testo scritto **in chiaro** con caratteri standard e in cui i **dati** sono organizzati con regole molto semplici e **immediatamente riconoscibili** semplicemente aprendo il file con un qualsiasi editor di testo (ed è esattamente questo il punto di forza del formato `.csv`).

In un tipico file `.csv` come questo:
→ nella prima riga sono riportati i nomi delle **variabili** ovvero dei **campi** che compongono i **record**;
→ nelle righe successive alla prima sono riportati, uno per riga, i casi ovvero i **record** che compongono il **database**;
→ il punto e virgola (;) è il **separatore di campo** che indica la fine di un campo e il passaggio al campo successivo;
→ come **separatore delle cifre decimali** è impiegata la virgola (**,**);
→ il fatto che la prima colonna/variabile debba essere interpretata come una variabile qualsiasi piuttosto che come un **identificativo univoco** di ciascun record/caso viene specificato nel momento in cui sono importati i dati (vedere qui sotto);
→ nel caso `GF` abbiamo un **dato mancante**, quello dell'altezza.

L'impiego del punto e virgola (;) come separatore di campo e della virgola (,) come separatore delle cifre decimali non è casuale. Il file `importa_csv.csv` è stato generato salvando in formato `.csv` i dati contenuti in un foglio elettronico



	A	B	C	D	E
1	id	sesso	anni	peso_kg	altezza_m
2	MT	M	69	76	1,78
3	GF	F	56	63	
4	MC	F	53	71	1,6
5	SB	M	28	73	1,78
6	FE	F	61	54	1,54
7	AB	M	46	92	1,84
8	RF	F	31	81	1,56

che è lo strumento più frequentemente impiegato per gestire i propri dati [53]. Il punto e virgola (;) come separatore di campo e la virgola (,) come separatore delle cifre decimali sono impostati di default in Windows nei paesi come Spagna, Italia e Francia, e il foglio elettronico nel nostro caso ha salvato il file `.csv` impiegando la configurazione di Windows [nei paesi anglosassoni in Windows sono impostati di default la virgola (,) come separatore di campo e il punto (.) come separatore delle cifre decimali].

Ora copiate e incollate nella `Console di R` questo script e premete ↵ `Invio`:

```
# IMPORTA I DATI DI UN FILE CSV
# notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",")
#
```

All'oggetto **mydata** viene assegnato (**<-**) il contenuto importato dalla funzione **read.table()** [6]. Gli argomenti della funzione, racchiusi nella parentesi, specificano che:
→ il file dal quale importare i dati è **C:/R/importa_csv.csv**;
→ la prima riga nel file è una riga di intestazione con i nomi delle variabili (**header=TRUE**);
→ nel file è impiegato come separatore di campo il punto e virgola (**sep=";"**);
→ nel file è impiegato come separatore delle cifre decimali la virgola (**dec=","**).

Se ora nella `Console di R` digitate

```
mydata
```

vedete comparire i dati che avete appena importato:

```
> mydata
  id sesso anni peso_kg altezza_m
1 MT     M   69      76      1.78
2 GF     F   56      63        NA
3 MC     F   53      71      1.60
4 SB     M   28      73      1.78
5 FE     F   61      54      1.54
6 AB     M   46      92      1.84
7 RF     F   31      81      1.56
```

Notate che:
→ dalla tabella/database `importa_csv.csv` sono stati importati 7 record/casi ciascuno contenente cinque campi/variabili (`id`, `sesso`, `anni`, `peso_kg`, `altezza_m`);

→ **R** ha aggiunto a ciascuno dei casi un identificativo univoco sotto forma di un numero progressivo (`1` per la prima riga/caso, `2` per la seconda riga/caso, e così via);
→ la virgola (,) presente come separatore delle cifre decimali nei dati originali è stata automaticamente trasformata da **R** in un punto (.).

Se avessimo deciso che nel campo "`id`" è contenuto l'**identificativo univoco** di ciascun caso avremmo dovuto eseguire quest'altro script:

```
# IMPORTA I DATI DI UN FILE CSV CON IDENTIFICATIVO UNIVOCO DEI CASI
# notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",",
row.names="id")
#
```

Rispetto al precedente script, alla funzione **read.table()** è stato aggiunto l'argomento **row.names="id"** mediante il quale viene specificato che come identificativo di ciascuna riga/caso deve essere impiegato il campo `id`. In questo modo **R**:
→ dalla tabella/database `importa_csv.csv` importerà 7 record ciascuno contenente quattro campi/variabili (`sesso`, `anni`, `peso_kg`, `altezza_m`);
→ assocerà a ciascuno dei casi l'identificativo univoco contenuto nel campo `id`.

Se ora nella `Console di R` digitate

```
mydata
```

vedete che **R** non ha più assegnato ai casi l'identificativo numerico (`1`, `2`, `...`) previsto di default, ma che ciascuna riga/caso è stato identificato in modo univoco mediante il valore contenuto nel campo `id` (`MT` per il primo caso/record, `GF` per il secondo caso/record, e così via):

```
> mydata
   sesso anni peso_kg altezza_m
MT     M   69      76      1.78
GF     F   56      63        NA
MC     F   53      71      1.60
SB     M   28      73      1.78
FE     F   61      54      1.54
AB     M   46      92      1.84
RF     F   31      81      1.56
```

A questo punto ecco la domanda che sarà venuta in mente a tutti: cosa accade se due casi/record per errore hanno lo stesso identificativo, quindi se il campo specificato in realtà non contiene un identificativo univoco?

Supponiamo che tabella/database `importa_csv.csv` contenga questi dati (il primo e il terzo caso hanno ora lo stesso identificativo `MT`):

```
id;sesso;anni;peso_kg;altezza_m
MT;M;69;76;1,78
GF;F;56;63;
MT;F;53;71;1,60
SB;M;28;73;1,78
FE;F;61;54;1,54
AB;M;46;92;1,84
RF;F;31;81;1,56
```

In questo caso questa ecco la risposta che comparirebbe nella `Console di R`:

```
Error in read.table("C:/Rdati/importa_csv.csv", header = TRUE, sep = ";", dec = ",",
:
  duplicate 'row.names' are not allowed
```

Quindi **R** non importa i dati e ci viene in aiuto in modo che non possiamo sbagliare su un aspetto così critico.

Dato che l'importazione dei dati è un punto di svolta nell'apprendimento di **R**, si consiglia di familiarizzare adeguatamente con questo aspetto adattando gli script qui riportati a file `.csv` contenenti propri dati.

Chi desidera importare i dati direttamente da file `.xls` o `.xlsx` può consultare il post Importazione dei dati da un file .xls o .xlsx.

---

[1] In alcuni pacchetti di **R** possono essere richieste strutture dati differenti, che sono peraltro specificate e illustrate nei manuali di riferimento dei pacchetti che le richiedono.

[2] Vedere il post Salvare uno script.

[3] Macchina da scrivere oramai da tempo soppiantata, ma è da qui che è originato il concetto che, passando per la telescrivente, è stato applicato a PC e stampante.

[4] Vedere il post Codifica dei caratteri ASCII, ANSI, Unicode (UTF).

[3] Non mi dilungo ulteriormente su questo punto in quanto qualsiasi foglio elettronico e qualsiasi database consente con una voce del menù tipo `Esporta...` o tipo `Salva con nome...` di esportare i dati in formato `.csv`

[4] Digitate **help(read.table)** nella `Console di R` per la documentazione della funzione **read.table()**.

mercoledì 28 novembre 2018

Importazione dei dati da un file .xls o .xlsx

Il formato dati raccomandato di **R** è il formato `.csv` e consente prima dell'importazione una ispezione preliminare dei dati, che sono accessibili in chiaro, semplicemente aprendo il file `.csv` con un editor di testo.

I formati `.xls` e `.xlsx` sono invece formati binari, nei quali i dati non sono accessibili in chiaro, e non sono formati standardizzati per cui la struttura dei file potrebbe cambiare senza preavviso nelle nuove versioni dei programmi che salvano in questi formati, causando errori imprevedibili e pertanto non gestibili nell'importazione dei dati in **R** [1]. Nonostante questo, data la larga diffusione dei dati salvati in questi formati, in **R** si trovano pacchetti che consentono di importare i dati direttamente da file `.xls` e `.xlsx` come ad esempio il pacchetto xlsx che si può scaricate dal **CRAN** (**C**omprehensive **R** **A**rchive **N**etwork) selezionando nel menù `Pacchetti` di **R** l'opzione `Installa pacchetti…` e quindi selezionando `xlsx` dall'elenco dei pacchetti disponibili.

Per la documentazione completa del pacchetto **xlsx** vedere il suo manuale di riferimento [2]. Se nel menù `Aiuto` di **R** selezionate `Guida Html` nella sezione `Reference` alla voce **Packages** trovate la documentazione dei pacchetti che avete installato sul vostro PC o notebook e che include, però in una versione meno completa, la documentazione del pacchetto **xlsx**.

Potete anche digitare **help(read.xlsx)** nella `Console di R` per una documentazione concisa ed essenziale della funzione **read.xlsx()**.

Per proseguire ora è necessario:
→ effettuare il download del file `importa_xls.xls`
→ effettuare il download del file `importa_xlsx.xlsx`
→ salvare i file nella cartella `C:\Rdati\`

Per questo e gli altri file di dati impiegati nei post trovate link e modalità di download alla pagina Dati.

Copiate e incollate nella `Console di R` questo script e premete ↵ `Invio`:

```
# IMPORTA I DATI DI UN FILE XLS
#
# carica il pacchetto xlsx
require(xlsx)
# carica nell'oggetto mydata i dati del file e del foglio specificati, notare / invece di \ su windows
mydata <- read.xlsx("C:/Rdati/importa_xls.xls", sheetName="peso_altezza")
#
```

La funzione **require()** carica in **R** il pacchetto **xlsx** contenente la funzione **read.xlsx()** che permette di importare i dati.

Gli unici argomenti richiesti dalla funzione **read.xlsx()** sono il nome del file con il percorso completo (**"C:/Rdati/importa_xls.xls"**) e il nome del foglio che contiene i dati (**sheetName="peso_altezza"**) all'interno del file. Quest'ultimo argomento è cruciale in quanto permette di gestire i molteplici fogli che possono essere presenti all'interno di un unico file `.xls` o `xlsx`.

Se ora digitate **mydata** e premete ↵ Invio potete scorrere nella `Console di R` i dati che sono stati importati:

```
> mydata
  id sesso anni peso_kg altezza_m
1 MT     M   69      76      1.78
2 GF     F   56      63        NA
3 MC     F   53      71      1.60
4 SB     M   28      73      1.78
5 FE     F   61      54      1.54
6 AB     M   46      92      1.84
7 RF     F   31      81      1.56
```

Per avere la conferma del fatto che i dati sono stati importati correttamente confrontateli con l'originale aprendo il file con *Excel* o in alternativa con un programma appartenente alla categoria del software libero [3] come *OpenOffice calc* o *LibreOffice calc*. La sola differenza che riscontrerete risiederà nel separatore dei decimali, che nel foglio elettronico vedrete essere (nella configurazione italiana di Windows) la virgola (,) mentre nei dati importati trovate il punto (.) che è il separatore delle cifre decimali impiegato da **R**.

Notare che **R** assegna automaticamente un **identificativo univoco** numerico (**1**, **2**, eccetera) ai casi/righe.

Per importare un file `.xlsx` copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

```
# IMPORTA I DATI DI UN FILE XLSX
#
# carica il pacchetto xlsx
require(xlsx)
# carica nell'oggetto mydata i dati del file e del foglio specificati, notare / invece di \ su windows
mydata <- read.xlsx("C:/Rdati/importa_xlsx.xlsx", sheetName="peso_altezza",
row.names="id")
#
```

Questa volta l'argomento **row.names="id"** specifica che gli identificativi univoci dei casi sono contenuti nel campo id del file. Se ora digitate **mydata** e premete `↵ Invio` potete scorrere nella `Console di R` i dati che sono stati importati

```
> mydata
   sesso anni peso_kg altezza_m
MT     M   69      76      1.78
GF     F   56      63        NA
MC     F   53      71      1.60
SB     M   28      73      1.78
FE     F   61      54      1.54
AB     M   46      92      1.84
RF     F   31      81      1.56
```

nei quali l'identificativo univoco numerico di **R** non compare più, essendo stato sostituito da quello già presente nei dati importati.

Importare in **R** file `.xls` e `.xlsx` salvati con un foglio elettronico è quindi semplice. Tuttavia si ricorda che qualsiasi foglio elettronico è in grado di salvare i dati anche in formato `.csv` e che è questo il formato raccomandato da **R** (vedere il post Importazione dei dati da un file .csv).

---

[1] Il manuale ufficiale *R Data Import/Export* a pag. 29 così recita: "*The most common R data import/export question seems to be 'how do I read an Excel spreadsheet' ... The first piece of advice is to avoid doing so if possible!*". URL consultato il 17/10/2018: https://goo.gl/8mzH4L

[2] *xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files*. URL consultato il 28/11/2018: https://goo.gl/DJCHTx

[3] Per il significato di *Software libero*, di *Software open source* e di *Software di dominio pubblico* vedere: *The Free Software Foundation. GNU Operating System. Categories of free and nonfree software*. URL consultato il 28/11/2018: https://goo.gl/MqwM2Q

giovedì 29 novembre 2018

Il set di dati ais

Nel set di dati **ais** che viene distribuito con il pacchetto **DAAG** sono contenuti i dati rilevati su 202 atleti australiani così espressi [1]:
→ [`campo senza nome`] = un numero progressivo da 1 a 202 identificativo del caso [2]
→ `rcc` = eritrociti (globuli rossi) espressi in 10^{12}/L o in 10^{6}/μL (le due espressioni sono numericamente equivalenti)
→ `wcc` = leucociti (globuli bianchi) espressi in 10^{9}/L o in 10^{3}/μL (le due espressioni sono numericamente equivalenti)
→ `hc` = ematòcrito espresso in %
→ `hg` = emoglobina espressa in g/dL
→ `ferr` = ferritina espressa in μg/L
→ `bmi` = indice di massa corporea espresso in kg/m^2
→ `ssf` = somma dello spessore delle pliche cutanee di tricipite e bicipite (nell'avambraccio), sottoscapolare e sovrailiaca espressa in mm
→ `pcBfat` = percentuale di grasso corporeo espresso in %
→ `lbm` = massa magra espressa in kg
→ `ht` = altezza espressa in cm
→ `wt` = peso espresso in kg
→ `sex` = sesso (m,f)
→ `sport` = sport praticato (B_Ball, Field, Gym, Netball, Row, Swim, T_400m, T_Sprnt, Tennis, W_Polo)

Lo scopo è mettere in relazione sesso (`sex`) e sport praticato (`sport`) con alcuni dati ematologici (`rcc`, `wbc`, `hc`, `hg`, `ferr`) e con alcuni dati biometrici (`bmi`, `ssf`, `pcBfat`, `lbm`, `ht`, `wt`) dell'atleta.

Se non l'avete già fatto, dovete scaricare e installare il pacchetto **DAAG**. Ricordatevi che esso prevede il pacchetto **latticeExtra** che in realtà dovrebbe essere scaricato automaticamente con il **DAAG** (eventualmente scaricatelo voi).

Per utilizzare il set di dati **ais** dovete semplicemente caricare il pacchetto digitando nella `Console di R` la riga di codice

library(DAAG)

A questo punto trovate i dati nell'oggetto **ais** del quale potete visualizzare la struttura digitando

str(ais)

Nella `Console di R` vi compariranno queste righe:

```
> str(ais)
'data.frame':   202 obs. of  13 variables:
 $ rcc   : num  3.96 4.41 4.14 4.11 4.45 4.1 4.31 4.42 4.3 4.51 ...
 $ wcc   : num  7.5 8.3 5 5.3 6.8 4.4 5.3 5.7 8.9 4.4 ...
 $ hc    : num  37.5 38.2 36.4 37.3 41.5 37.4 39.6 39.9 41.1 41.6 ...
 $ hg    : num  12.3 12.7 11.6 12.6 14 12.5 12.8 13.2 13.5 12.7 ...
 $ ferr  : num  60 68 21 69 29 42 73 44 41 44 ...
 $ bmi   : num  20.6 20.7 21.9 21.9 19 ...
 $ ssf   : num  109.1 102.8 104.6 126.4 80.3 ...
```

```
$ pcBfat: num  19.8 21.3 19.9 23.7 17.6 ...
$ lbm   : num  63.3 58.5 55.4 57.2 53.2 ...
$ ht    : num  196 190 178 185 185 ...
$ wt    : num  78.9 74.4 69.1 74.9 64.6 63.7 75.2 62.3 66.5 62.9 ...
$ sex   : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
$ sport : Factor w/ 10 levels "B_Ball","Field",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Come vedete l'oggetto **ais** (tecnicamente in **R** si tratta di un `data frame`) contiene 10 variabili numeriche indicate con `num` più due variabili indicate come `Factor`. Le variabili fattore, o più semplicemente i `fattori`, in **R** sono molto importanti: sono le variabili qualitative che consentono di raggruppare i dati numerici suddividendoli in sottoinsiemi e consentendo una analisi dei dati di maggior dettaglio. Nel caso del set di dati **ais** quindi, grazie ai fattori, le singole variabili numeriche potranno essere analizzate statisticamente e/o essere rappresentate graficamente sia nella loro totalità sia suddividendo al bisogno i dati per sesso e/o per sport praticato.

Se volete impiegare il set di dati **ais** indipendentemente dal pacchetto **DAAG** che lo contiene potete impiegare un file nel quale ho provveduto a esportare l'intero set di dati. In questo caso dovete fare il download del file `ais.csv` come indicato alla pagina Dati, salvare il file `ais.csv` nella cartella `C:\Rdati\` quindi negli script sostituire la riga

library(DAAG)

con la riga

ais <- read.table("c:/Rdati/ais.csv", header=TRUE, sep=";", dec=",")

per caricare il set di dati. Da notare che viene impiegata la virgola come separatore delle cifre decimali (**dec=","**) per garantire la compatibilità con l'installazione standard in lingua italiana di Excel, OpenOffice Calc, eccetera.

Infine potete anche copiare le 203 righe riportate qui sotto aggiungendo un ↵ Invio al termine dell'ultima riga, e salvare il tutto in `C:\Rdati\` nel file di testo denominato `ais.csv` (attenzione all'estensione `.csv` al momento del salvataggio del file), quindi di nuovo negli script sostituire **library(DAAG)** con la riga riportata sopra che importa i dati dal file.

```
rcc;wcc;hc;hg;ferr;bmi;ssf;pcBfat;lbm;ht;wt;sex;sport
3,96;7,5;37,5;12,3;60;20,56;109,1;19,75;63,32;195,9;78,9;f;B_Ball
4,41;8,3;38,2;12,7;68;20,67;102,8;21,3;58,55;189,7;74,4;f;B_Ball
4,14;5;36,4;11,6;21;21,86;104,6;19,88;55,36;177,8;69,1;f;B_Ball
4,11;5,3;37,3;12,6;69;21,88;126,4;23,66;57,18;185;74,9;f;B_Ball
4,45;6,8;41,5;14;29;18,96;80,3;17,64;53,2;184,6;64,6;f;B_Ball
4,1;4,4;37,4;12,5;42;21,04;75,2;15,58;53,77;174;63,7;f;B_Ball
4,31;5,3;39,6;12,8;73;21,69;87,2;19,99;60,17;186,2;75,2;f;B_Ball
4,42;5,7;39,9;13,2;44;20,62;97,9;22,43;48,33;173,8;62,3;f;B_Ball
4,3;8,9;41,1;13,5;41;22,64;75,1;17,95;54,57;171,4;66,5;f;B_Ball
4,51;4,4;41,6;12,7;44;19,44;65,1;15,07;53,42;179,9;62,9;f;B_Ball
4,71;5,3;41,4;14;38;25,75;171,1;28,83;68,53;193,4;96,3;f;B_Ball
4,62;7,3;43,8;14,7;26;21,2;76,8;18,08;61,85;188,7;75,5;f;B_Ball
4,35;7,8;41,4;14,1;30;22,03;117,8;23,3;48,32;169,1;63;f;B_Ball
4,26;6,2;41;13,9;48;25,44;90,2;17,71;66,24;177,9;80,5;f;Row
4,63;6;43,7;14,7;30;22,63;97,2;18,77;57,92;177,5;71,3;f;Row
4,36;5,8;40,3;13,3;29;21,86;99,9;19,83;56,52;179,6;70,5;f;Row
3,91;7,3;37,6;12,9;43;22,27;125,9;25,16;54,78;181,3;73,2;f;Row
4,51;8,3;43,7;14,7;34;21,27;69,9;18,04;56,31;179,7;68,7;f;Row
4,37;8,1;41,8;14,3;53;23,47;98;21,79;62,96;185,2;80,5;f;Row
```

4,9;6,9;44;14,5;59;23,19;96,8;22,25;56,68;177,3;72,9;f;Row
4,46;5,7;39,2;13;43;23,17;80,3;16,25;62,39;179,3;74,5;f;Row
3,95;3,3;36,9;12,5;40;24,54;74,9;16,38;63,05;175,3;75,4;f;Row
4,46;9,5;41,5;14,5;92;22,96;83;19,35;56,05;174;69,5;f;Row
5,02;6,4;44,8;15,2;48;19,76;91;19,2;53,65;183,3;66,4;f;Row
4,26;5,8;41,2;14,1;77;23,36;76,2;17,89;65,45;184,7;79,7;f;Row
4,46;5,6;41,1;14,3;71;22,67;52,6;12,2;64,62;180,2;73,6;f;Row
4,16;5,8;39,8;13,3;37;24,24;111,1;23,7;60,05;180,2;78,7;f;Row
4,49;7,6;41,8;14,4;71;24,21;110,7;24,69;56,48;176;75;f;Row
4,21;7,5;38,4;13,2;73;20,46;74,7;16,58;41,54;156;49,8;f;Row
4,57;6,6;42,8;14,5;85;20,81;113,5;21,47;52,78;179,7;67,2;f;Row
4,87;6,4;44,8;15;64;20,17;99,8;20,12;52,72;180,9;66;f;Row
4,44;10,1;42,7;14;19;23,06;80,3;17,51;61,29;179,5;74,3;f;Row
4,45;6,6;42,6;14,1;39;24,4;109,5;23,7;59,59;178,9;78,1;f;Row
4,41;5,9;41,1;13,5;41;23,97;123,6;22,39;61,7;182,1;79,5;f;Row
4,87;7,3;44,1;14,8;13;22,62;91,2;20,43;62,46;186,3;78,5;f;Row
4,56;13,3;42,2;13,6;20;19,16;49;11,29;53,14;176,8;59,9;f;Netball
4,15;6;38;12,7;59;21,15;110,2;25,26;47,09;172,6;63;f;Netball
4,16;7,6;37,5;12,3;22;21,4;89;19,39;53,44;176;66,3;f;Netball
4,32;6,4;37,7;12,3;30;21,03;98,3;19,63;48,78;169,9;60,7;f;Netball
4,06;5,8;38,7;12,8;78;21,77;122,1;23,11;56,05;183;72,9;f;Netball
4,12;6,1;36,6;11,8;21;21,38;90,4;16,86;56,45;178,2;67,9;f;Netball
4,17;5;37,4;12,7;109;21,47;106,9;21,32;53,11;177,3;67,5;f;Netball
3,8;6,6;36,5;12,4;102;24,45;156,6;26,57;54,41;174,1;74,1;f;Netball
3,96;5,5;36,3;12,4;71;22,63;101,1;17,93;55,97;173,6;68,2;f;Netball
4,44;9,7;41,4;14,1;64;22,8;126,4;24,97;51,62;173,7;68,8;f;Netball
4,27;10,6;37,7;12,5;68;23,58;114;22,62;58,27;178,7;75,3;f;Netball
3,9;6,3;35,9;12,1;78;20,06;70;15,01;57,28;183,3;67,4;f;Netball
4,02;9,1;37,7;12,7;107;23,01;77;18,14;57,3;174,4;70;f;Netball
4,39;9,6;38,3;12,5;39;24,64;148,9;26,78;54,18;173,3;74;f;Netball
4,52;5,1;38,8;13,1;58;18,26;80,1;17,22;42,96;168,6;51,9;f;Netball
4,25;10,7;39,5;13,2;127;24,47;156,6;26,5;54,46;174;74,1;f;Netball
4,46;10,9;39,7;13,7;102;23,99;115,9;23,01;57,2;176;74,3;f;Netball
4,4;9,3;40,4;13,6;86;26,24;181,7;30,1;54,38;172,2;77,8;f;Netball
4,83;8,4;41,8;13,4;40;20,04;71,6;13,93;57,58;182,7;66,9;f;Netball
4,23;6,9;38,3;12,6;50;25,72;143,5;26,65;61,46;180,5;83,8;f;Netball
4,24;8,4;37,6;12,5;58;25,64;200,8;35,52;53,46;179,8;82,9;f;Netball
3,95;6,6;38,4;12,8;33;19,87;68,9;15,59;54,11;179,6;64,1;f;Netball
4,03;8,5;37,7;13;51;23,35;103,6;19,61;55,35;171,7;68,8;f;Netball
4,36;5,5;41,4;13,8;82;22,42;71,3;14,52;55,39;170;64,8;f;Swim
4,07;5,9;39,5;13,3;25;20,42;54,6;11,47;52,23;170;59;f;Swim
4,17;4,9;38,9;12,9;86;22,13;88,2;17,71;59,33;180,5;72,1;f;Swim
4,23;8,1;38,2;12,7;22;25,17;95,4;18,48;61,63;173,3;75,6;f;Swim
4,46;8,3;42,2;14,4;30;23,72;47,5;11,22;63,39;173,5;71,4;f;Swim
4,38;5,8;42;14;27;21,28;55,6;13,61;60,22;181;69,7;f;Swim
4,31;5,3;41,1;13,9;60;20,87;62,9;12,78;55,73;175;63,9;f;Swim
4,51;5,1;40,9;14;115;19;52,5;11,85;48,57;170,3;55,1;f;Swim
4,13;7;39,7;13,1;124;22,04;62,6;13,35;51,99;165;60;f;Swim
4,48;9,5;36,5;13,3;54;20,12;49,9;11,77;51,17;169,8;58;f;Field
5,31;9,5;47,1;15,9;29;21,35;57,9;11,07;57,54;174,1;64,7;f;T_400m
4,58;5,8;42,1;14,7;164;28,57;109,6;21,3;68,86;175;87,5;f;Field
4,81;6,8;42,7;15,3;50;26,95;98,5;20,1;63,04;171,1;78,9;f;Field
4,51;9;39,7;14,3;36;28,13;136,3;24,88;63,03;172,7;83,9;f;Field
4,77;7,1;40,6;14,6;40;26,85;103,6;19,26;66,85;175,6;82,8;f;Field
5,33;9,3;47;15;62;25,27;102,8;19,51;59,89;171,6;74,4;f;Field
4,75;7,5;43,8;15,2;90;31,93;131,9;23,01;72,98;172,3;94,8;f;Field

4,11;7,3;38,7;12,4;12;16,75;33,8;8,07;45,23;171,4;49,2;f;T_400m
4,76;7,6;42,9;13,4;36;19,54;43,5;11,05;55,06;178;61,9;f;T_Sprnt
4,27;6,9;44,1;14,7;45;20,42;46,2;12,39;46,96;162;53,6;f;T_400m
4,44;6,1;42,6;13,9;43;22,76;73,9;15,95;53,54;167,3;63,7;f;T_400m
4,2;6,5;39,1;13;51;20,12;36,8;9,91;47,57;162;52,8;f;T_400m
4,71;6,9;43,5;13,8;22;22,35;67;16,2;54,63;170,8;65,2;f;T_400m
4,09;6,4;40,1;13,2;44;19,16;41,1;9,02;46,31;163;50,9;f;T_400m
4,24;6,6;38,2;12,6;26;20,77;59,4;14,26;49,13;166,1;57,3;f;T_400m
3,9;6;38,9;13,5;16;19,37;48,4;10,48;53,71;176;60;f;T_400m
4,82;7,6;43,2;14,4;58;22,37;50;11,64;53,11;163,9;60,1;f;T_Sprnt
4,32;6,8;40,6;13,7;46;17,54;54,6;12,16;46,12;173;52,5;f;T_400m
4,77;7,2;43,3;14,8;43;19,06;42,3;10,53;53,41;177;59,7;f;T_400m
5,16;8,2;45,3;14,7;34;20,3;46,1;10,15;51,48;168;57,3;f;T_Sprnt
4,97;7,8;44,7;14,2;41;20,15;46,3;10,74;53,2;172;59,6;f;T_Sprnt
4;4,2;36,6;12;57;25,36;109;20,86;56,58;167,9;71,5;f;Tennis
4,4;4;40,8;13,9;73;22,12;98,1;19,64;56,01;177,5;69,7;f;Tennis
4,38;7,9;39,8;13,5;88;21,25;80,6;17,07;46,52;162,5;56,1;f;Tennis
4,08;6,6;37,8;12,1;182;20,53;68,3;15,31;51,75;172,5;61,1;f;Tennis
4,98;6,4;44,8;14,8;80;17,06;47,6;11,07;42,15;166,7;47,4;f;Tennis
5,16;7,2;44,3;14,5;88;18,29;61,9;12,92;48,76;175;56;f;Tennis
4,66;6,4;40,9;13,9;109;18,37;38,2;8,45;41,93;157,9;45,8;f;Tennis
4,19;9;39;13,4;69;18,93;43,5;10,16;42,95;158,9;47,8;f;Gym
4,53;5;40,7;14;41;17,79;56,8;12,55;38,3;156,9;43,8;f;Gym
4,09;4,9;36;12,5;66;17,05;41,6;9,1;34,36;148,9;37,8;f;Gym
4,42;6,4;42,8;14,5;63;20,31;58,9;13,46;39,03;149;45,1;f;Gym
5,13;7,1;46,8;15,9;34;22,46;44,5;8,47;61;172,7;67;m;Swim
4,83;7,6;45,2;15,2;97;23,88;41,8;7,68;69;176,5;74,4;m;Swim
5,09;4,7;46,6;15,9;55;23,68;33,7;6,16;74;183;79,3;m;Swim
5,17;4,1;44,9;15;76;23,15;50,9;8,56;80;194,4;87,5;m;Swim
5,11;6,7;46,1;15,6;93;22,32;40,5;6,86;78;193,4;83,5;m;Swim
5,03;7,1;45,1;15,2;46;24,02;51,2;9,4;71;180,2;78;m;Swim
5,32;6;47,5;16,3;155;23,29;54,4;9,17;71;183;78;m;Swim
4,75;8,6;45,5;15,2;99;25,11;52,3;8,54;78;184;85;m;Swim
5,34;6,6;48,6;16,5;35;22,81;57;9,2;77;192,7;84,7;m;Swim
4,87;4,8;44,9;15,4;124;26,25;65,3;11,72;81;187,2;92;m;Swim
5,33;5,2;47,8;16,1;176;21,38;52;8,44;66;183,9;72,3;m;Swim
4,81;6,2;45,2;15,3;107;22,52;42,7;7,19;77;192;83;m;Swim
4,32;4,3;41,6;14;177;26,73;35,2;6,46;91;190,4;96,9;m;Swim
4,87;8,2;43,8;15;130;23,57;49,2;9;78;190,7;85,7;m;Row
5,04;7,1;44;14,8;64;25,84;61,8;12,61;75;181,8;85,4;m;Row
4,4;5,3;42,5;14,5;109;24,06;46,5;9,03;78;188,3;85,3;m;Row
4,95;5,9;45,4;15,5;125;23,85;34,8;6,96;87;198;93,5;m;Row
4,78;9,3;43;14,7;150;25,09;60,2;10,05;78;186;86,8;m;Row
5,21;6,8;44,5;15,4;115;23,84;48,1;9,56;79;192;87,9;m;Row
5,22;8,4;47,5;16,2;89;25,31;44,5;9,36;79;185,6;87,2;m;Row
5,18;6,5;45,4;14,9;93;19,69;54;10,81;48;165,3;53,8;m;Row
5,4;6,8;49,5;17,3;183;26,07;44,7;8,61;82;185,6;89,8;m;Row
4,92;5,4;46,2;15,8;84;25,5;64,9;9,53;82;189;91,1;m;Row
5,24;7,5;46,5;15,5;70;23,69;43,8;7,42;82;193,4;88,6;m;Row
5,09;10,1;44,9;14,8;118;26,79;58,3;9,79;83;185,6;92,3;m;Row
4,83;5;43,8;15,1;61;25,61;52,8;8,97;88;194,6;97;m;Row
5,22;6;46,6;15,7;72;25,06;43,1;7,49;83;189;89,5;m;Row
4,71;8;45,5;15,6;91;24,93;78;11,95;78;188,1;88,2;m;Row
5,24;7,2;46,6;15,9;58;22,96;40,8;7,35;85;200,4;92,2;m;B_Ball
4,54;5,9;44,4;15,6;97;20,69;41,5;7,16;73;195,3;78,9;m;B_Ball
5,13;5,8;46,1;15,9;110;23,97;50,9;8,77;82;194,1;90,3;m;B_Ball

5;6,7;45,3;15,7;72;24,64;49,6;9,56;79;187,9;87;m;B_Ball
5,17;8;47,9;16,4;36;25,93;88,9;14,53;97;209,4;113,7;m;B_Ball
4,89;7,5;41,6;14,4;53;23,69;48,3;8,51;90;203,4;98;m;B_Ball
4,5;9,2;40,7;13,7;72;25,38;61,8;10,64;90;198,7;100,2;m;B_Ball
4,84;8,3;46,3;15,9;39;22,68;43;7,06;74;187,1;79,4;m;B_Ball
4,13;8,9;40,3;13,5;61;23,36;61,1;8,87;82;196,6;90,3;m;B_Ball
4,87;7,4;43,5;15;49;22,44;43,8;7,88;72;186,1;77,7;m;B_Ball
4,82;6,4;44,3;14,8;35;22,57;54,2;9,2;76;192,8;83,9;m;B_Ball
4,73;6,7;42,8;14,9;8;19,81;41,8;7,19;70;195,2;75,5;m;B_Ball
4,55;5,6;42,6;14,4;106;21,19;34,1;6,06;57;169,1;60,6;m;T_400m
4,71;7,2;43,6;14;32;20,39;30,5;5,63;67;186,6;71;m;T_400m
4,93;7,3;46,2;15,1;41;21,12;34;6,59;67;184,4;71,8;m;T_400m
5,21;7,5;47,5;16,5;20;21,89;46,7;9,5;70;187,3;76,8;m;T_400m
5,09;8,9;46,3;15,4;44;29,97;71,1;13,97;88;185,1;102,7;m;Field
5,11;9,6;48,2;16,7;103;27,39;65,9;11,66;83;185,5;94,2;m;Field
4,94;6,3;45,7;15,5;50;23,11;34,3;6,43;74;184,9;79;m;Field
4,87;6,3;45,8;16,1;41;21,75;34,6;6,99;62;175;66,6;m;T_400m
4,41;4,5;44,2;15;101;20,89;31,8;6;67;185,4;71,8;m;T_400m
4,86;3,9;44,9;15,4;73;22,83;34,5;6,56;70;181;74,8;m;T_400m
4,91;9;46,3;15,4;56;22,02;31;6,03;64;176;68,2;m;T_400m
4,93;7,3;45,2;15,8;74;20,07;32,6;6,33;58;176,2;62,3;m;T_400m
4,2;4,5;41,2;14,3;58;20,15;31,5;6,82;57;174;61;m;T_400m
5,1;6,1;45,3;14,9;87;21,24;32,6;6,2;73;191;77,5;m;T_400m
4,5;6,1;42,2;14,7;139;19,63;31;5,93;54;171;57,4;m;T_400m
4,89;5,8;45,5;15,6;82;23,58;28;5,8;67;174;71,4;m;T_Sprnt
5,13;4;44,1;15,2;87;21,65;33,7;6,56;66;180,2;70,3;m;T_Sprnt
4,88;4,3;45,6;15,5;80;25,17;30,3;6,76;75;178,5;80,2;m;T_Sprnt
5;8,2;46,8;14,7;67;23,25;38;7,22;78;190,3;84,2;m;Field
5,48;4,6;49,4;18;132;32,52;55,7;8,51;102;185;111,3;m;Field
5,93;6,4;49,1;16,1;43;22,59;37,5;7,72;74;189;80,7;m;Field
5,01;8,9;46;15,9;212;30,18;112,5;19,94;78;180,1;97,9;m;Field
5,48;6,2;48,2;16,3;94;34,42;82,7;13,91;106;189,2;123,2;m;Field
5,16;8,4;44,4;15,5;213;21,86;29,7;6,1;68;182,6;72,9;m;T_Sprnt
4,64;9;42,9;14,9;122;23,99;38,9;7,52;77;186;83;m;T_Sprnt
6,72;7,1;59,7;19,2;76;24,81;44,8;9,56;69;174,9;75,9;m;T_Sprnt
4,83;6,6;43,8;14,3;53;21,68;30,9;6,06;66;180,6;70,7;m;T_400m
5,34;7,6;48,3;16,2;91;21,04;44;7,35;62;178,6;67,1;m;T_400m
5,13;4,6;45,3;16,8;36;23,12;37,5;6;65;173;69,2;m;T_400m
4,68;4,8;43;14,8;101;20,76;37,6;6,92;62;179,7;67,1;m;T_400m
5;5,2;45,1;15,1;184;23,13;31,7;6,33;66;174,6;70,5;m;T_Sprnt
4,99;7,2;41,4;14,9;44;22,35;36,6;5,9;67;178;70,8;m;T_Sprnt
5,49;5,9;47,7;15,9;66;22,28;48;8,84;65;178,5;71;m;T_400m
5,59;7,9;49,7;17,2;220;23,55;41,9;8,94;63;171,3;69,1;m;T_Sprnt
5,03;6,6;44,7;15,9;191;19,85;30,9;6,53;59;178;62,9;m;T_400m
5,5;6,4;48,1;16,5;40;26,51;52,8;9,4;86;189,1;94,8;m;T_Sprnt
5,11;9,3;45,4;15,8;189;24,78;43,2;8,18;87;195,4;94,6;m;Field
4,96;8,3;45,3;15,7;141;33,73;113,5;17,41;89;179,1;108,2;m;Field
5,01;8,9;46;15,9;212;30,18;96,9;18,08;80;180,1;97,9;m;Field
5,11;8,7;46,5;16,3;97;23,31;49,3;9,86;68;179,6;75,2;m;Field
5,69;10,8;50,5;18,5;53;24,51;42,3;7,29;69;174,7;74,8;m;T_Sprnt
4,63;9,1;42,1;14,4;126;25,37;96,3;18,72;77;192,7;94,2;m;W_Polo
4,91;10,2;45;15,2;234;23,67;56,5;10,12;68;179,3;76,1;m;W_Polo
4,95;7,5;44,5;15;50;24,28;105,7;19,17;77;197,5;94,7;m;W_Polo
5,34;10;46,8;16,2;94;25,82;100,7;17,24;71;182,7;86,2;m;W_Polo
5,16;12,9;47,6;15,6;156;21,93;56,8;9,89;72;190,5;79,6;m;W_Polo
5,29;12,7;48;16,2;124;23,38;75,9;13,06;74;191;85,3;m;W_Polo

5,02;6,1;43,6;14,8;87;23,07;52,8;8,84;68;179,6;74,4;m;W_Polo
5,01;9,8;46,5;15,8;97;25,21;47,8;8,87;85;192,6;93,5;m;W_Polo
5,03;7,5;43,6;14,4;102;23,25;76;14,69;75;194,1;87,6;m;W_Polo
5,25;7,4;47,3;15,8;55;22,93;61,2;8,64;78;193;85,4;m;W_Polo
5,08;8,5;46,3;15,6;117;26,86;75,6;14,98;86;193,9;101;m;W_Polo
5,04;6;45,9;15;52;21,26;43,3;7,82;69;187,7;74,9;m;W_Polo
4,63;14,3;44,8;15;133;25,43;49,5;8,97;79;185,3;87,3;m;W_Polo
5,11;7;47,7;15,8;214;24,54;70;11,63;80;191,5;90;m;W_Polo
5,34;6,2;49,8;17,2;143;27,79;75,7;13,49;82;184,6;94,7;m;W_Polo
4,86;8,9;46,9;15,8;65;23,58;57,7;10,25;68;179,9;76,3;m;W_Polo
4,9;7,6;45,6;16;90;27,56;67,2;11,79;82;183,9;93,2;m;W_Polo
5,66;8,3;50,2;17,7;38;23,76;56,5;10,05;72;183,5;80;m;Tennis
5,03;6,4;42,7;14,3;122;22,01;47,6;8,51;68;183,1;73,8;m;Tennis
4,97;8,8;43;14,9;233;22,34;60,4;11,5;63;178,4;71,1;m;Tennis
5,38;6,3;46;15,7;32;21,07;34,9;6,26;72;190,8;76,7;m;Tennis

Questo script infine carica il pacchetto **DAAG** ed esporta nel file `C:\Rdati\ais.csv` il set di dati **ais** [3] impiegando il punto e virgola come separatore di campo (**sep=";"**) e la virgola come separatore delle cifre decimali (**dec=","**).

```
# ESPORTA DA R IN UN FILE .csv IL SET DI DATI ais DEL PACCHETTO DAAG
# impiega come separatore di campo il punto e virgola (;)
# impiega come separatore delle cifre decimali la virgola (,)
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
write.table(ais, file="C:/Rdati/ais.csv", quote=FALSE, sep=";", dec=",", na="",
col.names=TRUE, row.names=FALSE) # esporta i dati, notare / invece di \ su windows
#
```

---

[1] La documentazione ufficiale del set di dati **ais** contiene alcuni errori nelle unità di misura, ad esempio per la concentrazione dell'emoglobina viene riportato "*hemaglobin concentration, in g per decaliter*", dove oltre a un primo refuso ("*hemaglobin*" in luogo di "*hemoglobin*" o "*haemoglobin*") compare "*decaliter*" cioè decalitro (dieci litri) invece di "*deciliter*" cioè decilitro (un decimo di litro).

[2] In assenza di un identificativo univoco di ciascun caso **R** lo crea automaticamente assegnando un numero progressivo identificativo univoco che viene riportato in un campo senza nome. L'identificativo univoco rende possibile effettuare al bisogno con **R** la tabulazione incrociata (cross-tabulation).

[3] Se non esiste, la cartella `C:\Rdati\` va creata prima di eseguire lo script.

giovedì 29 novembre 2018

Esportazione dei dati in un file .csv

Anche se normalmente quello che si fa è importare i dati in **R** [1], talora potrebbe essere necessario esportarli da **R** sotto forma di file. La soluzione migliore è quella di salvare i dati in un file `.csv` che potrà poi essere facilmente importato in qualsiasi software esterno.

La tecnica viene illustrata impiegando la funzione **write.table()** nella quinta e ultima riga di codice di questo script. Se non esiste già, la cartella `C:\Rdati\` va creata prima di eseguire lo script.

Copiate e incollate lo script nella `Console di R` e premete `↵ Invio`:

```
# ESPORTA I DATI DI UNA MATRICE
#
x <- c(4.2, 6.8, 2.5, 8.3, 5.4, 7.9, 5.3, 6.7, 2.2, 3.1) # genera l'array x
y <- c(3.1, 2.2, 6.7, 5.3, 7.9, 5.4, 8.3, 2.5, 6.8, 4.2) # genera l'array y
mymatrix <- data.frame(x, y) # combina gli array in una matrice
#
mymatrix # mostra la matrice nella quale R impiega il punto (.) come separatore dei decimali
#
write.table(mymatrix,file="C:/Rdati/esporta_csv.csv",sep=";", dec=",", quote=FALSE,
row.names=FALSE) # esporta i dati in un file di testo .csv sostituendo il punto (.) con la virgola (,)
#
```

Le prime tre righe servono semplicemente a generare la matrice [2] impiegata come esempio.

Quindi viene mostrata nella `Console di R` la matrice **mymatrix** con il punto (.) come separatore dei decimali e con gli identificativi univoci/nomi di riga (1, 2, eccetera) generati automaticamente da **R**:

```
> mymatrix # mostra la matrice nella quale R impiega il punto (.) come separatore dei
decimali
     x   y
1  4.2 3.1
2  6.8 2.2
3  2.5 6.7
4  8.3 5.3
5  5.4 7.9
6  7.9 5.4
7  5.3 8.3
8  6.7 2.5
9  2.2 6.8
10 3.1 4.2
```

La funzione **write.table()** con i suoi argomenti:
→ esporta i dati dell'oggetto **mymatrix**;
→ genera in uscita il file **esporta_csv.csv** nella posizione **C:/Rdati/**;
→ impiega come separatore di campo il punto e vigola (**sep=";"**)
→ impiega come separatore dei decimali la virgola (**dec=","**)
→ non chiude tra le virgolette "" i campi non numerici (**quote=FALSE**)
→ non esporta i nomi delle righe (**row.names=FALSE**)

Questo è il contenuto del file `C:\Rdati\esporta_csv.csv` generato con la funzione **write.table()** e aperto con un editor di testo:

```
x;y
4,2;3,1
6,8;2,2
2,5;6,7
8,3;5,3
5,4;7,9
7,9;5,4
5,3;8,3
6,7;2,5
2,2;6,8
3,1;4,2
```

Compaiono come previsto nello script il punto e virgola (;) come separatore di campo e la virgola (,) come separatore dei decimali. Notare che grazie all'argomento **row.names=FALSE** nel file esportato non compaiono gli identificativi/nomi di riga di **R**:

Potete importare senza difficoltà il file `esporta_csv.csv` in *Excel* o in un programma appartenente alla categoria del software libero [3] come *OpenOffice calc* o *LibreOffice calc*.

Ma in realtà **R** oltre alla funzione **write.table()** fornisce altre due funzioni per esportare i dati in formato `.csv`:
→ la funzione **write.csv()** che consente di esportare i dati impiegando la virgola (,) come separatore di campo [4] e il punto (.) come separatore dei decimali e quindi impiega i separatori secondo le convenzioni vigenti nel mondo anglosassone;
→ la funzione **write.csv2()** che consente di esportare i dati impiegando il punto e virgola (;) come separatore di campo e la virgola (,) come separatore dei decimali e quindi impiega i separatori secondo le convenzioni vigenti in Italia, Spagna, Francia e in altri Paesi europei.

Per illustrare queste modalità nello script che segue viene costruita una tabella che viene esportata nella cartella `C:\Rdati\` in quattro file differenti impiegando le tre funzioni:
→ write.table()
→ write.csv()
→ write.csv2()

Copiate e incollate lo script nella `Console di R` e premete `↵ Invio`:

```
# ESPORTA I DATI DI UNA TABELLA
#
# genera cinque array
id <- c("MT", "GF", "MC", "SB", "FE", "AB", "RF")
sesso <- c("M", "F", "F", "M", "F", "M", "F")
anni <- c(69, 56, 53, 28 ,61, 46, 31)
peso_kg <- c(76, 63, 71, 73, 54, 92, 81)
altezza_m <- c(1.78, NA, 1.60, 1.78, 1.54, 1.84, 1.56)
mytable <- data.frame(id, sesso, anni, peso_kg, altezza_m) # combina gli array in una tabella
#
mytable # mostra la tabella nella quale R impiega il punto (.) come separatore dei decimali
#
# (a) write.table consente sia (.) sia (,) come separatore dei decimali
write.table(mytable, file="C:/Rdati/write_table_p.csv", quote=FALSE, sep=";", dec=".",
na="", col.names=TRUE, row.names=FALSE)
#
# (b) write.table consente sia (.) sia (,) come separatore dei decimali
write.table(mytable, file="C:/Rdati/write_table_v.csv", quote=FALSE, sep=";", dec=",",
na="", col.names=TRUE, row.names=FALSE)
#
```

```
# (c) write.csv usa (,) come separatore di campo e (.) come separatore dei decimali
write.csv(mytable, file="C:/Rdati/write_csv.csv", quote=FALSE, na="",
row.names=FALSE)
#
# (d) write.csv2 usa (;) come separatore di campo e (,) come separatore dei decimali
write.csv2(mytable, file="C:/Rdati/write_csv2.csv", quote=FALSE, na="",
row.names=FALSE)
#
```

Le prime sei righe servono semplicemente a generare la tabella impiegata come esempio.

Questo è il contenuto della tabella **mytable** [2] con il punto (.) come separatore dei decimali e con gli identificativi/nomi di riga (1, 2, eccetera) generati automaticamente da **R**:

```
> mytable # mostra la tabella nella quale R impiega il punto (.) come separatore dei
decimali
  id sesso anni peso_kg altezza_m
1 MT     M    69      76      1.78
2 GF     F    56      63        NA
3 MC     F    53      71      1.60
4 SB     M    28      73      1.78
5 FE     F    61      54      1.54
6 AB     M    46      92      1.84
7 RF     F    31      81      1.56
```

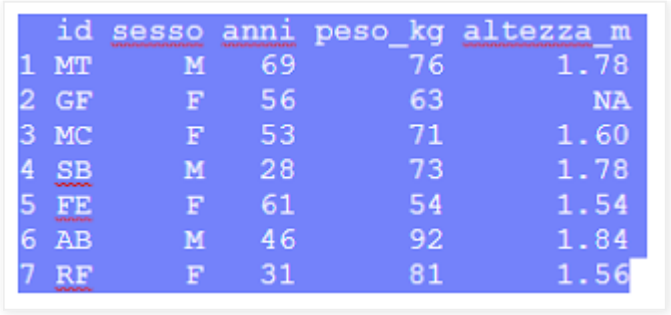
Dopo avere mostrato il contenuto della tabella:
→ in **(a)** la funzione **write.table()** viene impiegata una prima volta per esportare i dati nel file `write_table_p.csv` nel quale è previsto il punto (.) come separatore dei decimali;
→ in **(b)** la funzione **write.table()** viene impiegata una seconda volta per esportare i dati nel file `write_table_v.csv` nel quale è prevista la virgola (,) come separatore dei decimali;
→ in **(c)** la funzione **write.csv()** viene impiegata per esportare i dati nel file `write_csv.csv` impiegando i separatori secondo le convenzioni vigenti nel *mondo anglosassone*, cioè la virgola (,) come separatore di campo [4] e il punto (.) come separatore dei decimali;
→ in **(d)** la funzione **write.csv2()** viene impiegata per esportare i dati nel file `write_csv2.csv` impiegando i separatori secondo le convenzioni vigenti in *Europa*, cioè il punto e virgola (;) come separatore di campo e la virgola (,) come separatore dei decimali.

Da notare che il separatore dei decimali nelle funzioni **write.csv()** e **write.csv2()** è fissato all'interno delle funzioni stesse e non può essere cambiato.

I file generati sono salvati nella cartella `C:\Rdati\` e anche in questo caso possono essere aperti con un editor di testo per verificarne il contenuto e possono essere importati in *Excel* o in *OpenOffice calc* o in *LibreOffice calc*.

Dopo avere illustrato il metodo canonico per esportare i dati da **R** [5], aggiungo che in realtà esiste anche un metodo naïf ma a dire il vero molto rapido e perfettamente funzionante.

Supponiamo che vi interessi esportare, esattamente come è stata visualizzata nella `Console di R` con **mytable**, la tabella riportata qui sopra. Selezionate il testo della tabella in questo modo



	id	sesso	anni	peso_kg	altezza_m
1	MT	M	69	76	1.78
2	GF	F	56	63	NA
3	MC	F	53	71	1.60
4	SB	M	28	73	1.78
5	FE	F	61	54	1.54
6	AB	M	46	92	1.84
7	RF	F	31	81	1.56

quindi copiatelo e incollatelo, aggiungendo un ↲ `Invio` al termine dell'ultima riga, in un editor di file di testo come ad esempio il `Blocco note` di Windows, infine salvatelo nella cartella `Rdati` con il nome `naif` - lasciate pure che l'editor aggiunga l'estensione di default `.txt` e quindi il nome completo del file sarà `naif.txt`.

Sappiamo che un file `.txt` è identico a un file `.csv` a parte, appunto, l'estensione del nome del file. In questo caso abbiamo un file di testo che impiega come separatore di campo un tabulatore `<TAB>` e come separatore dei decimali il punto (.).

Se ora copiate e incollate nella `Console di R` questa riga

mynaif <- read.table("c:/Rdati/naif.txt", header=TRUE, sep="", dec=".")

dove tra le virgolette in **sep=""** ho avuto cura di inserire (anche se non lo vedete) il carattere `<TAB>`, ritrovate in **mynaif** i vostri dati:

```
> mynaif <- read.table("c:/Rdati/naif.txt", header=TRUE, sep="	", dec=".")
> mynaif
  id sesso anni peso_kg altezza_m
1 MT     M   69      76      1.78
2 GF     F   56      63        NA
3 MC     F   53      71      1.60
4 SB     M   28      73      1.78
5 FE     F   61      54      1.54
6 AB     M   46      92      1.84
7 RF     F   31      81      1.56
```

In buona sostanza avete esportato i dati semplicemente copiandoli direttamente dalla `Console di R` e salvandoli in un file di testo - dal quale li potete rileggere con **R** ma anche con quasiasi altro applicativo - senza digitare nemmeno una riga di codice!

---

[1] Vedere i post:
→ Importazione dei dati da un file .csv
→ Importazione dei dati da un file di solo testo
→ Importazione dei dati da un file .xls o .xlsx

[2] Parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14

6	7	18	12
11	8	17	13
7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

[3] Per il significato di "Software libero", di "Software open source" e di "Software di dominio pubblico" vedere: *The Free Software Foundation. GNU Operating System. Categories of free and nonfree software*. URL consultato il 28/11/2018: https://goo.gl/MqwM2Q

[4] L'impiego della virgola come separatore di campo (,) è alla base della modalità `comma separated value` che definisce la struttura dati dei relativi file e dal cui acronimo è derivato il nome `.csv` dell'estensione.

[5] Un esempio che illustra come impiegare la funzione **write.table()** per esportare in un file `.csv` uno dei numerosi set di dati forniti con **R** e con i suoi pacchetti aggiuntivi è riportato nel post Il set di dati ais.

giovedì 29 novembre 2018

Il set di dati galton

Il set di dati **galton**, inizialmente contenuto nella libreria **psych** ma recentemente spostato nella libreria **psychTools** (in caso di difficoltà verificate entrambe le librerie, non si sa mai...) contiene 928 coppie di valori di altezza (misurata in pollici) di altrettanti padri e dei relativi figli. Questa in breve la storia del set di dati.

Francis Galton [1] era cugino di Charles Darwin e dopo la lettura de "*L'origine delle specie*" si era convinto che la preminenza delle persone nei vari campi fosse dovuta essenzialmente a fattori genetici. Si era quindi dedicato alla ricerca delle leggi della genetica. E l'osservazione di fenomeni di "regressione verso la media" dell'altezza dei figli rispetto all'altezza dei padri lo portò a concepire l'idea della necessità di opporvisi, applicando quelli che riteneva essere i principi di una "buona" genetica ("eugenetica").

In un suo famoso lavoro [2] Galton così si esprime:

"*... This memoir contains the data upon which the remarks on the Law of Regression were founded ... My object is to place beyond doubt the existence of a simple and far-reaching law that governs the hereditary transmission of, I believe, every one of those simple qualities which all possess, though in unequal degrees.*"

"*It is some years since I made an extensive series of experiments on the produce of seeds of different size but of the same species. They yielded results that seemed very noteworthy, and I used them as the basis of a lecture before the Royal Institution on February 9th, 1877. It appeared from these experiments that the offspring did not tend to resemble their parent seeds in size, but to be always more mediocre than they - to be smaller than the parents, if the parents were large; to be larger than the parents, if the parents were very small.*"

"*It was anthropological evidence that I desired, caring only for the seeds as means of throwing light on heredity in man ... inducing me to make an offer of prizes for Family Records, which was largely responded to, and furnished me last year with what I wanted ... An analysis of the Records fully confirms and goes far beyond the conclusions I obtained from the seeds. It gives the numerical value of the regression towards mediocrity in the case of human stature ... My data consisted of the heights of 930 adult children and of their respective parentages, 205 in number...*"

A pagina 254 del lavoro Galton cita "*... a total of 928 children ...*" che corrisponde al numero delle coppie di valori riportati in **R** nel set di dati **galton**.

L'espressione "regressione" è quindi stata associata al metodo statistico per il calcolo dell'equazione della retta, che è diventata così la "retta di regressione", in seguito agli studi di Galton. Per i problemi che nascono applicando ai dati di Galton il modello tradizionale di regressione lineare, quello più largamente impiegato, vedere il post Regressione lineare semplice parametrica e il post La regressione lineare: assunti e modelli.

Per utilizzare il set di dati **galton** dovete scaricare e installare dal **CRAN** il pacchetto **psych** quindi caricare il set di dati con la riga di codice
library(psychTools)
come riportato nei vari script che impiegano questi dati.

In alternativa per caricare il set di dati potete fare il download del file galton.csv come indicato alla pagina Dati, salvare il file `galton.csv` nella cartella `C:\Rdati\` [3] quindi negli script sostituire la riga
library(psychTools)
con la riga
galton <- read.table("c:/Rdati/galton.csv", header=TRUE, sep=";", dec=",")
Da notare che invece del punto viene impiegata la virgola come separatore delle cifre decimali (**dec=","**) per garantire la compatibilità con l'installazione standard in lingua italiana di Excel, OpenOffice Calc, eccetera.

Potete anche copiare le righe riportate qui sotto e salvarle in `C:\Rdati\` nel file di testo denominato `galton.txt`. Attenzione: l'ultimo carattere nel file deve essere un ↵ Invio (cioè un "a capo") che dovete avere cura di aggiungere immediatamente dopo il 73,7 finale prima di salvare il file.

70,5;61,7\n68,5;61,7\n65,5;61,7\n64,5;61,7\n64;61,7\n67,5;62,2\n67,5;62,2\n67,5;62,2\n66,5;62,2\n66,5;62,2\n66,5;62,2\n64,5;62,2\n70,5;63,2\n69,5;6

In questo caso come separatore tra record / righe non viene impiegato un ↵ `Invio` ma viene impiegato un `\n` che **R** interpreta come un ↵ `Invio` se nella funzione **read.table()** che importa i dati viene abilitato il riconoscimento dei caratteri di controllo con l'argomento **allowEscapes=TRUE**. Il vantaggio è rappresentato da un file di testo molto più compatto. Per importare i dati dal file `galton.txt` impiegate questo codice:
galton <- read.table("c:/Rdati/galton.txt", header=TRUE, col.names=c("parent","child"), sep=";", dec=",", allowEscapes=TRUE)

Questo script infine carica il pacchetto **psychTools** ed esporta nel file `C:\Rdati\galton.csv` il set di dati **galton** impiegando il punto e virgola come separatore di campo (**sep=";"**) e la virgola come separatore delle cifre decimali (**dec=","**).

```
# ESPORTA DA R IN UN FILE .csv IL SET DI DATI galton DEL PACCHETTO psychTools
# impiega come separatore di campo il punto e virgola (;)
# impiega come separatore delle cifre decimali la virgola (,)
```

```
#
library(psychTools) # carica il pacchetto incluso il set di dati galton
write.table(galton, file="C:/Rdati/galton.csv", quote=FALSE, sep=";", dec=",", na="",
col.names=TRUE, row.names=FALSE) # esporta i dati, notare / invece di \ su windows
#
```

---

[1] *Francis Galton*. MacTutor History of Mathematics archive. School of Mathematics and Statistics. University of St Andrews, Scotland. URL consultato il 29/11/2018: https://goo.gl/BVMXGn

[2] Francis Galton. *Regression Towards Mediocrity in Hereditary Stature*. The Journal of the Anthropological Institute of Great Britain and Ireland, 1886, Vol. 15, pp. 246-263. URL consultato il 29/11/2018: https://goo.gl/VWvBau

[3] Se non esiste, la cartella `C:\Rdati\` va creata.

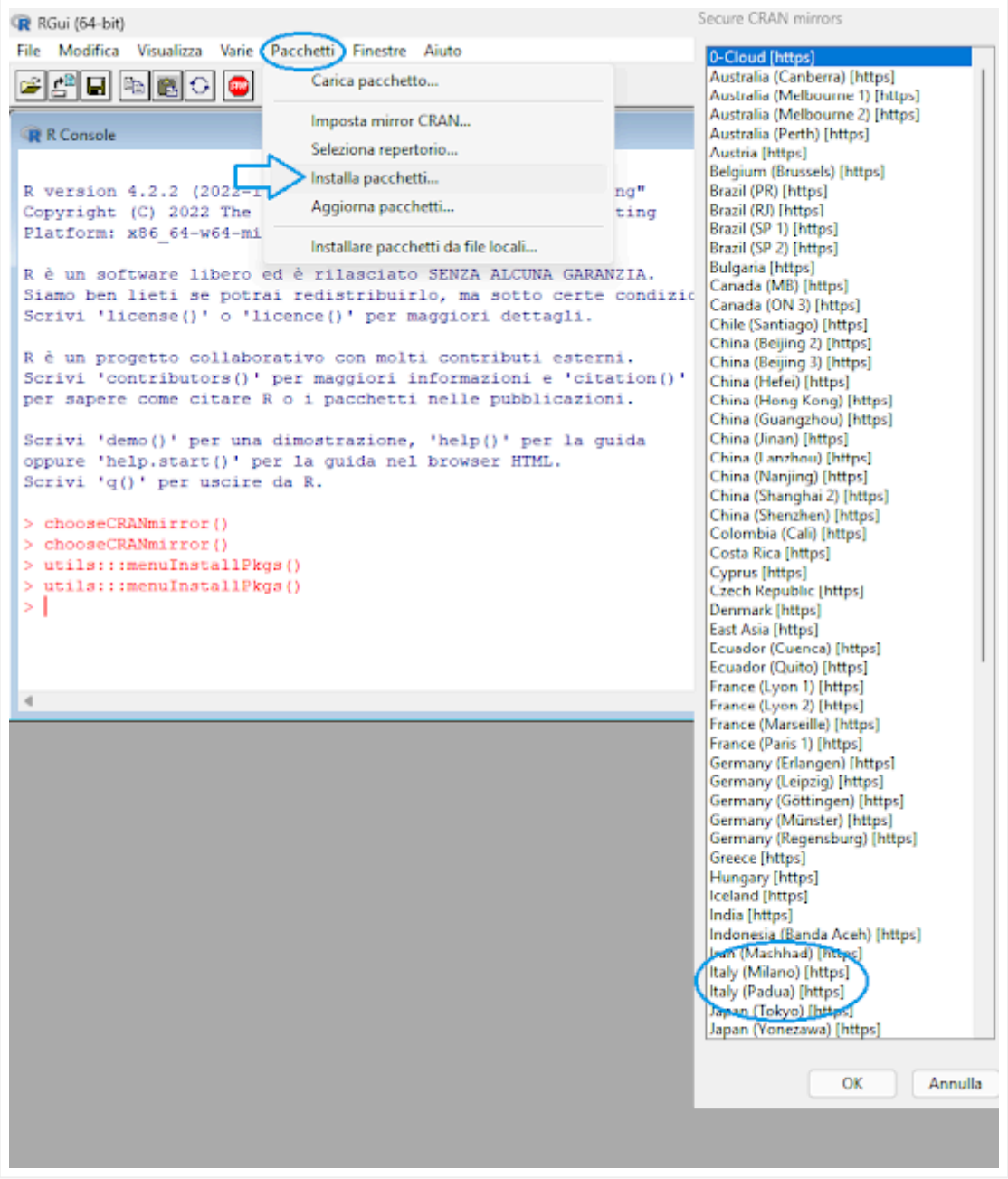
giovedì 29 novembre 2018

Pacchetti aggiuntivi di statistica e grafica

Il programma **R** che si scarica e si installa sul proprio PC o notebook da sito ufficiale di **R** [1] include una serie di *funzioni statistiche e grafiche di base*, che tuttavia possono essere integrate da una miriade di funzioni statistiche e grafiche contenute nei **pacchetti aggiuntivi** disponibili, ciascuno orientato alla risoluzione di un problema specifico.

Per scaricare un pacchetto aggiuntivo e installarlo sul vostro PC o notebook:
→ selezionare dalla `RGui` l'opzione `Pacchetti`
→ selezionare dal menù a tendina che compare `Installa pacchetti...`

Si aprirà prima una finestra per selezionare nel **CRAN** (**C**omprehensive **R** **A**rchive **N**etwork) il sito dal quale scaricare il pacchetto, per l'Italia potete selezionare `Italy (Milano)` o `Italy (Padua)`, entrambi siti `https`.



Una volta selezionato il sito si aprirà una finestra con l'elenco dei pacchetti disponibili. Selezionate il pacchetto e premete `OK` per installarlo.

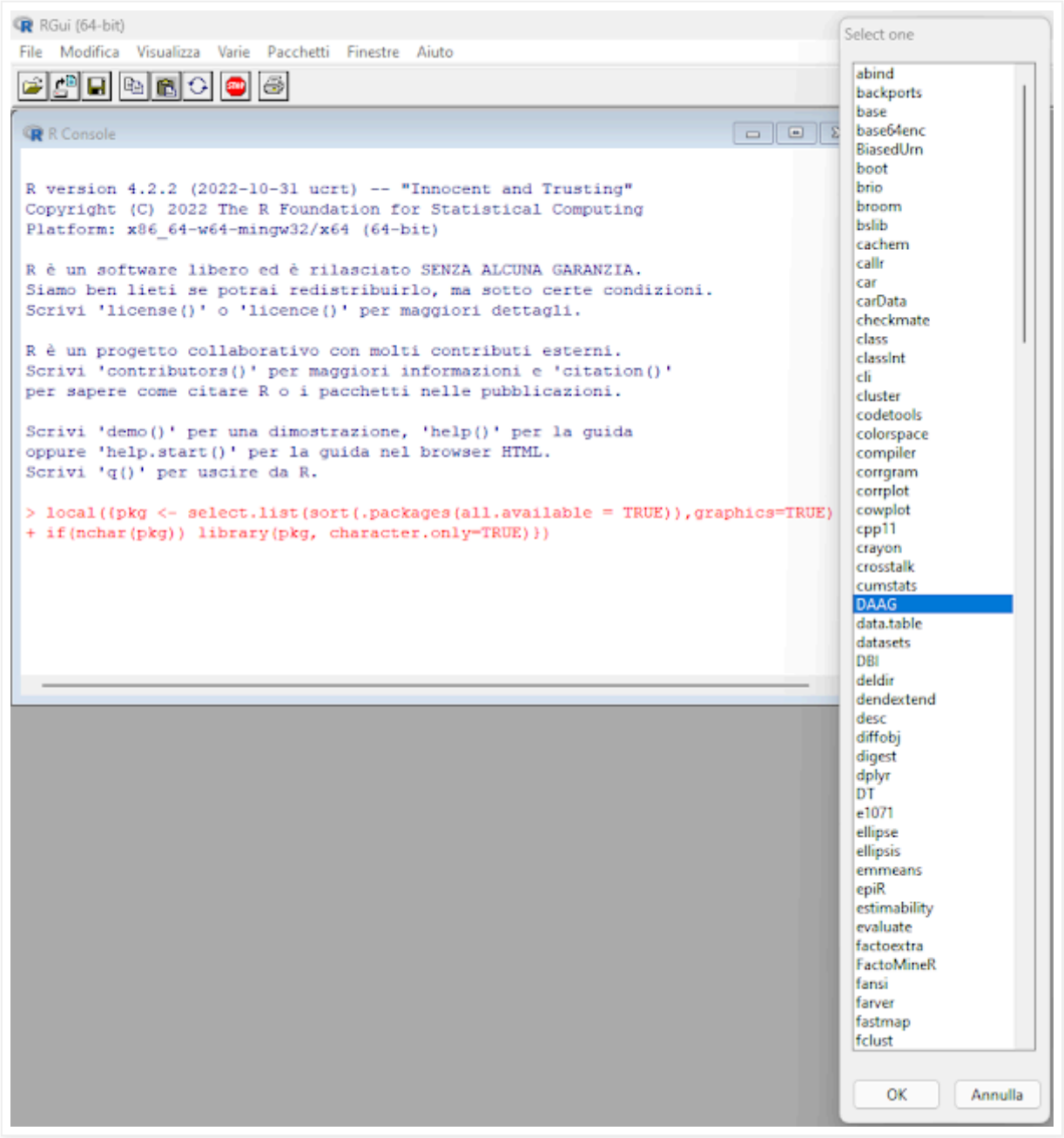
Quando si installa un pacchetto è possibile che questo richieda che sul PC o notebook siano installati altri pacchetti, cui esso si appoggia. Se questi non sono installati verrà effettuato automaticamente il download e l'installazione di più pacchetti, tutti quelli tra loro concatenati e quindi necessariamente e contemporaneamente richiesti per garantire la funzionalità dell'unico pacchetto che si desiderava installare. Ora scaricate e installate il pacchetto **DAAG** che ci serve tra poco (vedere il post il set di dati ais).

In **R** un "**pacchetto**" è una raccolta di funzioni, di set di dati e di codice **R** compilato. Quando si scarica e si installa un pacchetto, questo viene immagazzinato nella locale **libreria**. Ecco perché se nella `Console di R` digitate

library

potete vedere l'elenco dei pacchetti che avete installato e che sono disponibili in **R** sul vostro PC o notebook.

Per utilizzare le **funzioni** e i **set di dati** contenuti in un pacchetto è necessario caricarlo come segue:
→ dalla `RGui` fare click su `Pacchetti`;
→ selezionare `Carica pacchetto...`;
→ nella finestra che compare selezionare il pacchetto da caricare.



In alternativa è possibile, ed è più pratico, caricare i pacchetti impiegando all'interno degli script la funzione **library()** o la funzione **require()** e riportando come argomento all'interno della parentesi **()** il nome del pacchetto.

Una volta caricato un pacchetto è possibile avere l'elenco di tutti i set di dati che sono stati caricati con quel pacchetto, e digitando nella `Console di R` il nome di uno di questi set di dati poterne vederne il contenuto:

library(DAAG) # carica il pacchetto DAAG
data(package="DAAG") # elenca i set di dati caricati con il pacchetto DAAG
ais # mostra il contenuto del set di dati ais

Mediante la funzione **data()** possono essere caricati set di dati senza caricare il pacchetto che li contiene, così ad esempio è possibile caricare selettivamente il set di dati **ais** senza caricare il pacchetto **DAAG** digitando nella `Console di R`

data(ais, package="DAAG")

Digitando nella `Console di R`

help(package="DAAG")

si accede alla documentazione delle funzioni e dei set di dati contenuti nel pacchetto.

Per scaricare il manuale di riferimento in formato `.pdf` di un pacchetto:
→ accedere al **CRAN** (per esempio su https://cran.stat.unipd.it/);
→ fare click sulla voce Packages nella sezione *Software*;
→ fare click sul link Table of available packages, sorted by name;
→ fare click sul nome del pacchetto;
→ il manuale si trova alla voce *Reference manual*.

Per aggiornare periodicamente i pacchetti installati:
→ dalla `RGui` selezionare l'opzione `Pacchetti`;
→ dal menù a tendina che compare selezionare `Aggiorna pacchetti...`

I pacchetti non più manutenuti vengono rimossi dal **CRAN**, ma sono ancora resi disponibili nell'archivio storico di **R** al link https://bit.ly/2FVFufj (URL consultato il 14/01/2023).

---

[1] *The R Project for Statistical Computing*. URL consultato il 07/08/2018: https://www.r-project.org/

sabato 1 dicembre 2018

Test chi-quadrato 1 riga · n colonne

Le leggi della genetica applicate alla riproduzione umana consentono di stabilire che la probabilità che un nuovo nato sia di sesso maschile o di sesso femminile è la stessa ed è pari a `p = 0.5` per entrambi i sessi. Nel Dipartimento materno-infantile di un ospedale in un anno sono nati 817 femmine e 756 maschi. I dati sono organizzati in una tabella di una riga per due colonne:

Femmine	Maschi
817	756

Apparentemente sono nate più femmine. Ci si domanda se la differenza tra i casi osservati e le frequenze attese previste sia significativa.

Per rispondere alla domanda impieghiamo questo breve script che esegue il **test chi-quadrato** (χ^2). Copiate e incollate nella `Console di R` lo script e premete `↵ Invio`:

```
# TEST CHI-QUADRATO - 1 riga · n colonne
#
casi.osservati <- c(817, 756) # sono immessi i casi osservati
prob.teor <- c(0.5, 0.5) # sono immesse le probabilità previste
chiquad <- chisq.test(casi.osservati, p=prob.teor) # i risultati del test sono salvati nell'oggetto chiquad
chiquad$observed # mostra i casi osservati
chiquad$expected # mostra le frequenze attese
chiquad # mostra i risultati del test chi-quadrato
#
```

Come prima cosa mediante la funzione **c()** costruiamo il vettore che contiene i casi osservati (**817,756**), che per comodità e leggibilità del codice che seguirà salviamo nell'oggetto **casi.osservati**. Quindi nella seconda riga di codice facciamo la stessa cosa per le probabilità teoriche che salviamo nell'oggetto **prob.teor**.

Nella terza riga viene eseguito il test chi-quadrato impiegando la funzione **chisq.test()** [1] che ha come argomento i casi osservati e le probabilità teoriche, che devono essere inserite con l'argomento **p=**.

Il risultato del test viene salvato (**<-**) nell'oggetto **chiquad**, dal quale sono prima estratti e mostrati i dati osservati (**chiquad$observed**), poi le frequenze attese calcolate sulla base dei casi osservati e delle relative probabilità teoriche (**chiquad$expected**). Infine sono mostrati i risultati del test chi-quadrato (**chiquad**).

Questi sono i risultati forniti dallo script:

```
> chiquad$observed # mostra i casi osservati
[1] 817 756
> chiquad$expected # mostra le frequenze attese
[1] 786.5 786.5
> chiquad # mostra i risultati del test chi-quadrato

Chi-squared test for given probabilities
```

```
data: casi.osservati
X-squared = 2.3655, df = 1, p-value = 0.124
```

La probabilità `p` riportata rappresenta la probabilità che la differenza osservata sia attribuibile al caso. Se tale probabilità è sufficientemente bassa, si assume che la differenza non sia attribuibile al caso, ovvero che la differenza sia significativa. In genere la soglia viene posta ad un valore di `p = 0.05` quindi per valori di `p < 0.05` la differenza osservata viene ritenuta significativa.

Nel caso del nostro test chi-quadrato la probabilità `p` che la differenza sia dovuta al caso è elevata (`p-value = 0.124`) e pertanto la conclusione è che il numero di femmine e di maschi osservato non è significativamente diverso da quello atteso che prevede femmine e maschi in ugual numero.

Un altro esempio è fornito da Marubini [2]. In una ricerca sull'ibridazione di specie vegetali ci si attende, in base alle leggi delle genetica, la produzione di individui appartenenti alle varietà **AB**, **Ab**, **aB**, **ab** nel rapporto di 9:3:3:1. Questo rapporto, espresso in termini di probabilità `p` diventa

0.5625:0.1875:0.1875:0.0625

essendo

9+3+3+1=16 e 9/16=0.5625, 3/16=0.1875, 3/16=0.1875, 1/16=0.0625

(notare che la somma di queste probabilità è necessariamente uguale a 1).

Il numero di individui prodotti per ciascuna delle varietà in un esperimento di ibridazione è riportato in una tabella di una riga per quattro colonne:

AB	**Ab**	**aB**	**ab**
72	29	36	12

Per verificare se la differenza tra i casi osservati e le probabilità previste sia significativa copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

```
# TEST CHI-QUADRATO - 1 riga · n colonne
#
casi.osservati <- c(72, 29, 36, 12) # sono immessi i casi osservati
prob.teorica <- c(0.5625, 0.1875, 0.1875, 0.0625) # sono immesse le probabilità previste
chiquad <- chisq.test(casi.osservati, p=prob.teorica) # i risultati del test salvati nell'oggetto chiquad
chiquad$observed # mostra i casi osservati
chiquad$expected # mostra le frequenze attese
chiquad # mostra i risultati del test chi-quadrato
#
```

Questi sono i risultati del test chi-quadrato:

```
> chiquad$observed # mostra i casi osservati
[1] 72 29 36 12
> chiquad$expected # mostra le frequenze attese
[1] 83.8125 27.9375 27.9375 9.3125
> chiquad # mostra i risultati del test chi-quadrato

Chi-squared test for given probabilities
```

```
data: casi.osservati
X-squared = 4.8076, df = 3, p-value = 0.1864
```

La probabilità `p` che la differenza tra i casi osservati e le frequenze attese sia dovuta al caso è elevata (`p-value = 0.1864`) e pertanto se ne deduce che il rapporto tra individui appartenenti alle varietà **AB**, **Ab**, **aB**, **ab** ottenuto nell'esperimento non è significativamente diverso da quello atteso di 9:3:3:1 previsto. La rappresentazione grafica di questi dati può essere fatta mediante un grafico a torta.

I due script, come tutti gli altri qui riportati, possono essere salvati e rapidamente riadattati per ripetere il test chi-quadrato su altri dati analoghi, in situazioni nelle quali si disponga dei casi osservati e delle relative probabilità teoriche.

---

[1] Digitate **help(chisq.test)** nella `Console di R` per la documentazione della funzione **chisq.test()**.

[2] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp. 293-295.

lunedì 3 dicembre 2018

Test chi-quadrato 2 righe · 2 colonne

Quando le osservazioni sono organizzate in una tabella di 2 righe · 2 colonne si applica questa regola:
→ si impiega il **test chi-quadrato** se le osservazioni sono indipendenti e sono numerose;
→ si impiega il test di Fisher se le osservazioni sono indipendenti e sono poche;
→ si impiega il test di McNemar nel caso di osservazioni non indipendenti (cioè nel caso di dati appaiati).

Il **test chi-quadrato** (χ^2) lo vediamo applicato a uno studio canadese sulla relazione tra fumo e mortalità divenuto famoso in quanto tra gli anni '70 e i primi anni '80 le sue conclusioni finirono col favorire tra i fumatori un revival dell'uso della pipa. Nell'arco di sei anni venne registrato il numero di decessi avvenuti in un gruppo di non fumatori e in un gruppo di fumatori di pipa con i seguenti risultati [1]:

Esito	Non fumatori	Fumatori di pipa
Deceduti	117	54
Viventi	950	348

Di 1 067 non fumatori 117 (il 10,97%) erano deceduti. Dei 402 fumatori di pipa 54 (il 13.43%) erano deceduti.

La domanda è: *esiste una differenza reale tra le mortalità nei due gruppi, o la differenza può ancora essere attribuita al caso*?

Questo script prevede di inserire i dati manualmente. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# TEST CHI-QUADRATO - 2 righe · 2 colonne
#
cells <- c(117, 54, 950, 348) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Deceduti", "Viventi") # genera l'array rnames con i nomi delle righe
cnames <- c("Non_fumatori", "Fumatori_di_pipa") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la matrice dei dati
mydata # mostra i dati
chisq.test(mydata, correct=TRUE) # test chi quadrato con la correzione di Yates per la continuità
#
```

Nelle prime tre righe sono generati con la funzione **c()**:
→ il vettore che contiene i quattro dati (che come si vede devono essere inseriti in sequenza leggendoli da sinistra a destra e dall'alto in basso) che sono salvati nell'oggetto **cells**;
→ il vettore che contiene i nomi delle righe, salvato nell'oggetto **rnames**;
→ il vettore che contiene i nomi delle colonne, salvato nell'oggetto **cnames**.

I tre vettori sono combinati a formare la matrice dei dati mediante la funzione **matrix()** che impiega gli argomenti che indicano:
→ l'oggetto/vettore contenente i dati (**cells**);
→ il numero di righe (**nrow**) e il numero di colonne (**ncol**) della matrice;

→ la modalità di riempimento della matrice, che deve essere riempita per righe (**byrow=TRUE**) quindi da sinistra a destra e dall'alto in basso;
→ i nomi da assegnare alle righe e alle colonne (**dimnames=list(rnames, cnames)**).

La matrice viene salvata nell'oggetto **mydata**, che viene mostrato per un controllo della corretta immissione e strutturazione dei dati.

Infine con la funzione **chisq.test()** [2] viene effettuato il calcolo del test chi-quadrato che, ponendo l'argomento **correct=TRUE**, prevede la correzione di Yates per la continuità.

Nota bene: il test chi-quadrato con 1 grado di libertà è esatto solo asintoticamente per dimensioni molto grandi dei campioni per cui si consiglia di applicare sempre alle tabelle 2x2 la correzione di Yates per la continuità [3].

Questi sono i risultati:

```
> mydata # mostra i dati
              Non_fumatori  Fumatori_di_pipa
Deceduti               117                54
Viventi                950               348
> chisq.test(mydata, correct=TRUE) # test chi quadrato con la correzione di Yates per
la continuità

        Pearson's Chi-squared test with Yates' continuity correction

data:  mydata
X-squared = 1.4969, df = 1, p-value = 0.2212
```

Come si vede la probabilità di osservare per caso il valore chi-quadrato `1.4969` è `p = 0.2212` quindi è elevata. La conclusione è che la differenza tra le mortalità nei due gruppi può ancora essere attribuita al caso ovvero, in termini statistici, la mortalità osservata nel gruppo dei fumatori di pipa non differisce significativamente dalla mortalità osservata nel gruppo dei non fumatori.

Con quest'altra riga

chisq.test(mydata, correct=FALSE) # test chi quadrato senza la correzione di Yates

potete calcolare il chi-quadrato senza la correzione di Yates per la continuità (`p = 0.1886`) che risulta inferiore al precedente. Notate quindi che applicando la correzione di Yates per la continuità il valore di `p` risulta superiore e questo rende la differenza meno significativa.

In alternativa potete anche copiare le tre righe riportate qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarle in `C:\Rdati\` nel file di testo denominato `chi_2x2.csv` (attenzione all'estensione al momento del salvataggio del file).

```
Esito;Non_fumatori;Fumatori_di_pipa
Deceduti;117;54
Viventi;950;348
```

Questo script prevede di eseguire il test chi-quadrato sui dati letti dal file `chi_2x2.csv`. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# TEST CHI-QUADRATO - 2 righe · 2 colonne
#
```

```
mydata <- read.table("C:/Rdati/chi_2x2.csv", header=TRUE, sep=";",
row.names="Esito") # importa i dati
mydata # mostra i dati
chisq.test(mydata, correct=TRUE) # test chi quadrato con la correzione di Yates per la continuità
#
```

Come vedete lo script è più compatto del precedente, le uniche cose da notare sono gli argomenti della funzione **read.table()**:
→ **"C:/Rdati/chi_2x2.csv"** che specifica nome e posizione del file dal quale importare i dati;
→ **header=TRUE** che indica che nella prima riga del file sono contenuti i nomi delle variabili;
→ **sep=";"** che specifica il separatore di campo impiegato nel file;
→ **row.names="Esito"** che indica che i nomi delle righe sono contenuti nel campo che ha questo nome.

Anche se ovviamente i risultati sono sempre gli stessi, il senso di queste due alternative è semplice. Se vi va bene intervenire ogni volta sullo script per adattarlo a nuovi dati, potete impiegare la prima. Potete invece impiegare la seconda se volete evitare di mettere mano ogni volta allo script, e preferite intervenire solamente sul file di dati. A voi la scelta, magari dopo averle sperimentate entrambe.

La rappresentazione grafica dei dati può essere effettuata mediante un grafico a barre.

---

[1] Best EWR et al. *A Canadian Study on Smoking and Health (Final Report)*. Dept. Natl. Health and Welfare, Canada, 1966. Citato in: Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6, p. 124.

[2] Digitate **help(chisq.test)** nella `Console di R` per la documentazione della funzione **chisq.test()**.

[3] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, Milano, 1979, p. 137.

venerdì 7 dicembre 2018

Test di McNemar

Quando le osservazioni sono organizzate in una tabella di 2 righe · 2 colonne si applica questa regola:
→ si impiega il test chi-quadrato se le osservazioni sono indipendenti e sono numerose;
→ si impiega il test di Fisher se le osservazioni sono indipendenti e sono poche;
→ si impiega il **test di McNemar** nel caso di osservazioni non indipendenti (cioè nel caso dati appaiati).

Un esempio tipico di applicazione del **test di McNemar** lo abbiamo in uno studio clinico nel quale lo stesso soggetto viene esposto in tempi diversi a due trattamenti differenti, secondo una sequenza casuale, garantendo che né il paziente né l'operatore sanitario sappiano quale è il trattamento effettuato in quella fase. In questo modo tutti i soggetti, ignorando ogni volta di quale trattamento si tratti, ricevono tutti e due i trattamenti.

Campbell [1] riporta i risultati di uno studio nel quale 250 pazienti sofferenti di artrite erano stati sottoposti ciascuno sia al trattamento con il farmaco A sia al trattamento con il farmaco B. Era stato poi rilevato il grado di soddisfazione dei pazienti rispetto all'uno e all'altro trattamento, con i risultati qui riportati:

Esito	Soddisfatto da A	Non soddisfatto da A
Soddisfatto da B	150	20
Non soddisfatto da B	30	50

In totale 150 soggetti si erano mostrati soddisfatti di entrambi i trattamenti, 50 avevano espresso insoddisfazione per entrambi, mente altri 50 si erano mostrati insoddisfatti chi dell'uno e chi dell'altro.

Questo script prevede di inserire i dati manualmente. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# TEST DI MCNEMAR - 2 righe · 2 colonne
#
cells <- c(150, 20, 30, 50) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Soddisfatto_da_B", "Non_soddisfatto_da_B") # genera l'array rnames con i nomi delle righe
cnames <- c("Soddisfatto_da_A", "Non_soddisfatto_da_A") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la matrice dei dati
mydata # mostra i dati
mcnemar.test(mydata, correct=TRUE) # esegue il test di McNemar
#
```

Nelle prime tre righe sono generati con la funzione **c()**:
→ il vettore che contiene i quattro dati (che come si vede devono essere inseriti in sequenza leggendoli da sinistra a destra e dall'alto in basso) che sono salvati nell'oggetto **cells**;
→ il vettore che contiene i nomi delle righe, salvato nell'oggetto **rnames**;
→ il vettore che contiene i nomi delle colonne, salvato nell'oggetto **cnames**.

I tre vettori sono combinati a formare la matrice dei dati mediante la funzione **matrix()** che impiega gli argomenti che indicano:

→ l'oggetto/vettore contenente i dati (**cells**);
→ il numero di righe (**nrow**) e il numero di colonne (**ncol**) della matrice;
→ la modalità di riempimento della matrice, che deve essere riempita per righe (**byrow=TRUE**) quindi da sinistra a destra e dall'alto in basso;
→ i nomi da assegnare alle righe e alle colonne (**dimnames=list(rnames, cnames)**).

La matrice dei dati viene salvata nell'oggetto **mydata**, che viene mostrato per un ovvio controllo della corretta immissione e strutturazione dei dati.

Infine con la funzione **mcnemar.test()** [2] viene effettuato il calcolo del test. Da notare che anche nel test di McNemar, che è di fatto una statistica chi-quadrato, viene applicata la correzione di Yates per la continuità [3] ponendo l'argomento **correct=TRUE**.

Questi sono i risultati

```
> mydata # mostra i dati
                     Soddisfatto_da_A Non_soddisfatto_da_A
Soddisfatto_da_B                  150                   20
Non_soddisfatto_da_B               30                   50
> matrix <- data.matrix(mydata) # viene generato l'oggetto matrice richiesto dalla fase successiva
> mcnemar.test(matrix, correct=TRUE) # esegue il test di McNemar

        McNemar's Chi-squared test with continuity correction

data:  matrix
McNemar's chi-squared = 1.62, df = 1, p-value = 0.2031
```

che dimostrano che la differenza tra i gradi di soddisfazione espressi dai pazienti in merito ai due trattamenti non è significativa (`p = 0.2031`).

In alternativa potete anche copiare le tre righe riportate qui sotto aggiungendo un `↲ Invio` al termine dell'ultima riga e salvarle in `C:\Rdati\` nel file di testo denominato `chi_McNemar.csv` (attenzione all'estensione al momento del salvataggio del file).

```
Esito;Soddisfatto_da_A;Non_soddisfatto_da_A
Soddisfatto_da_B;150;20
Non_Soddisfatto_da_B;30;50
```

Questo script prevede di eseguire il test di McNemar sui dati letti dal file `chi_McNemar.csv`. Copiatelo e incollatelo nella `Console di R` e premete `↲ Invio`:

```
# TEST DI MCNEMAR - 2 righe · 2 colonne
#
mydata <- read.table("C:/Rdati/chi_McNemar.csv", header=TRUE, sep=";",
row.names="Esito") # importa i dati
mydata # mostra i dati
matrix <- data.matrix(mydata) # viene generato l'oggetto matrice richiesto dalla fasi successive
mcnemar.test(matrix, correct=TRUE) # esegue il test di McNemar
#
```

Come vedete lo script è più compatto del precedente, le uniche cose da notare sono gli argomenti della funzione **read.table()**:
→ **"C:/Rdati/chi_McNemar.csv"** che specifica nome e posizione del file dal quale importare i dati;
→ **header=TRUE** che indica che nella prima riga del file sono contenuti i nomi delle variabili;
→ **sep=";"** che specifica il separatore di campo impiegato nel file;

→ **row.names="Esito"** che indica che i nomi delle righe sono contenuti nel campo “Esito”.

Anche se ovviamente i risultati sono sempre gli stessi, il senso di queste due alternative è semplice. Se vi va bene intervenire ogni volta sullo script per adattarlo a nuovi dati, potete impiegare la prima. Potete invece impiegare la seconda se volete evitare di mettere mano ogni volta allo script, e preferite intervenire solamente sul file di dati. A voi la scelta, magari dopo averle sperimentate entrambe.

La rappresentazione grafica dei dati può essere effettuata mediante un grafico a barre.

---

[1] Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, pp. 148-150.

[2] Digitate **help(mcnemar.test)** nella `Console di R` per la documentazione della funzione **mcnemar.test()**.

[3] Vedere il post: Test chi-quadrato 2 righe · 2 colonne.

sabato 8 dicembre 2018

Test chi-quadrato n righe · n colonne

Il **test chi-quadrato** può essere nella sua forma più generale esteso a **n righe · n colonne**. Il problema è che più sono le righe e le colonne e più è difficile individuare il valore (o i valori) che contribuiscono a rendere eventualmente significative le differenze. Per questa ragione in genere le tabelle con più righe e più colonne sono scomposte in tabelle più piccole al fine di valutare meglio i contributi dei valori osservati: per questo tipo di analisi, piuttosto delicata, si rimanda ai testi di statistica.

Qui riporto un esempio di una tabella di 2 righe per 5 colonne tratto da Marubini [1] nella quale si riesce a individuare senza troppe complicazioni il risultato che contribuisce a rendere significativa la differenza tra le frequenze osservate.

Si trattava di valutare l'efficacia di cinque differenti vaccini influenzali. I vaccini vennero inoculati a novembre in 1080 soggetti sani volontari e nel mese di marzo dell'anno successivo vennero contati i casi di influenza tra i vaccinati.

Questi erano stati i risultati della sperimentazione. Da notare che viene riportato il numero dei casi cioè il conteggio dei soggetti affetti da influenza e non affetti da influenza, come richiesto dal test chi-quadrato, nel quale invece non devono mai essere riportati i dati espressi in percentuale:

Esito	Vaccino_1	Vaccino_2	Vaccino_3	Vaccino_4	Vaccino_5
Influenza_si	39	44	20	41	42
Influenza_no	177	166	200	183	168

Questo script prevede di inserire i dati manualmente. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# TEST CHI-QUADRATO - n righe · n colonne
#
cells <- c(39, 44, 20, 41, 42, 177, 166, 200, 183, 168) # genera l'array cells con il numero dei casi
rnames <- c("Influenza_si", "Influenza_no") # genera l'array rnames con i nomi delle righe
cnames <- c("Vaccino_1", "Vaccino_2", "Vaccino_3", "Vaccino_4", "Vaccino_5") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=5, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la matrice dati
mydata # mostra i dati
chisq.test(mydata) # calcola il chi-quadrato
#
```

Nelle prime tre righe sono generati con la funzione **c()**:
→ il vettore che contiene i quattro dati (che come si vede devono essere inseriti in sequenza leggendoli da sinistra a destra e dall'alto in basso) che sono salvati nell'oggetto **cells**;
→ il vettore che contiene i nomi delle righe, salvato nell'oggetto **rnames**;
→ il vettore che contiene i nomi delle colonne, salvato nell'oggetto **cname**s.

I tre vettori sono combinati a formare la matrice dei dati mediante la funzione **matrix()** che impiega gli argomenti che indicano:
→ l'oggetto/vettore contenente i dati (**cells**);
→ il numero di righe (**nrow**) e il numero di colonne (**ncol**) della matrice;

→ la modalità di riempimento della matrice, che deve essere riempita per righe (**byrow=TRUE**) quindi da sinistra a destra e dall'alto in basso;
→ i nomi da assegnare alle righe e alle colonne (**dimnames=list(rnames, cnames)**).

La matrice viene salvata nell'oggetto **mydata**, che viene mostrato per un ovvio controllo della corretta immissione e strutturazione dei dati.

Infine con la funzione **chisq.test()** [2] viene effettuato il calcolo del test.

Questi sono i risultati:

```
> mydata # mostra i dati
             Vaccino_1 Vaccino_2 Vaccino_3 Vaccino_4 Vaccino_5
Influenza_si         39        44        20        41        42
Influenza_no        177       166       200       183       168
> chisq.test(mydata) # calcola il chi-quadrato

        Pearson's Chi-squared test

data:  mydata
X-squared = 13.678, df = 4, p-value = 0.008395
```

Il test chi-quadrato riporta un valore elevato della statistica chi-quadrato (`13.678`) cui corrisponde un valore di probabilità `p = 0.008395` che conferma quindi la presenza di una differenza significativa nell'efficacia dei vaccini.

In alternativa potete anche copiare le tre righe riportate qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarle in `C:\Rdati\` nel file di testo denominato `chi_rxc.csv` (attenzione all'estensione al momento del salvataggio del file).

```
Esito;Vaccino_1;Vaccino_2;Vaccino_3;Vaccino_4;Vaccino_5
Influenza_si;39;44;20;41;42
Influenza_no;177;166;200;183;168
```

Questo script prevede di eseguire il test chi-quadrato sui dati letti dal file `chi_rxc.csv`. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# TEST CHI-QUADRATO - n righe · n colonne
#
mydata <- read.table("C:/Rdati/chi_rxc.csv", header=TRUE, sep=";",
row.names="Esito") # importa i dati
mydata # mostra i dati
chisq.test(mydata) # calcola il chi-quadrato
#
```

Come vedete lo script è più compatto del precedente, le uniche cose da notare sono gli argomenti della funzione **read.table()**:
→ **"C:/Rdati/chi_rxc.csv"** che specifica nome e posizione del file dal quale importare i dati;
→ **header=TRUE** che indica che nella prima riga del file sono contenuti i nomi delle variabili;
→ **sep=";"** che specifica il separatore di campo impiegato nel file;
→ **row.names="Esito"** che indica che i nomi delle righe sono contenuti nel campo "Esito".

Anche se ovviamente i risultati sono sempre gli stessi, il senso di queste due alternative è semplice. Se vi va bene intervenire ogni volta sullo script per adattarlo a nuovi dati, potete impiegare la prima. Potete invece impiegare la seconda se volete evitare di mettere mano ogni volta allo script, e

preferite intervenire solamente sul file di dati. A voi la scelta, magari dopo averle sperimentate entrambe.

Se riprendiamo i casi osservati (Influenza_si / Influenza_no) e li trasformiamo in percentuale

Esito	Vaccino_1	Vaccino_2	Vaccino_3	Vaccino_4	Vaccino_5
Influenza_si	18.1	21.0	9.1	18.3	20.0
Influenza_no	81.9	79.0	90.8	81.7	80.0

vediamo che l'eterogeneità dei risultati evidenziata dal test chi-quadrato è da attribuire al vaccino 3, con il quale si è avuto nei vaccinati un tasso di malattia del 9.1%, a fronte di un tasso di malattia omogeneo e praticamente doppio, che va dal 18.1% al 21.0%, che si è avuto nei soggetti vaccinati con gli altri quattro vaccini.

Un utile complemento all'analisi dei dati può essere realizzato mediante la loro rappresentazione grafica dei dati sotto forma di grafico a barre affiancate e grafico a barre sovrapposte.

---

[1] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp. 290-291.

[2] Digitate **help(chisq.test)** nella `Console di R` per la documentazione della funzione **chisq.test()**.

[3] Digitate **help(barplot)** nella `Console di R` per la documentazione della funzione **barplot()**.

domenica 9 dicembre 2018

Grafici a torta

Per la rappresentazione grafica di dati qualitativi il metodo più generico, ma anche il più limitato, in quanto adatto essenzialmente alla rappresentazione dei rapporti percentuali, è rappresentato dai **grafici a torta** o **pie chart**. A causa del fatto che l'occhio umano ha una scarsa capacità nella valutazione degli angoli i grafici a torta non sono raccomandati nella rappresentazione di dati scientifici [1]. Nonostante questo anche in **R** sono disponibili strumenti grafici per la realizzazione dei grafici a torta.

Nell'esempio, tratto da Marubini [2], il numero di individui prodotti per ciascuna delle varietà **AB**, **Ab**, **aB**, **ab** in un esperimento di ibridazione di specie vegetali era:

AB	**Ab**	**aB**	**ab**
72	29	36	12

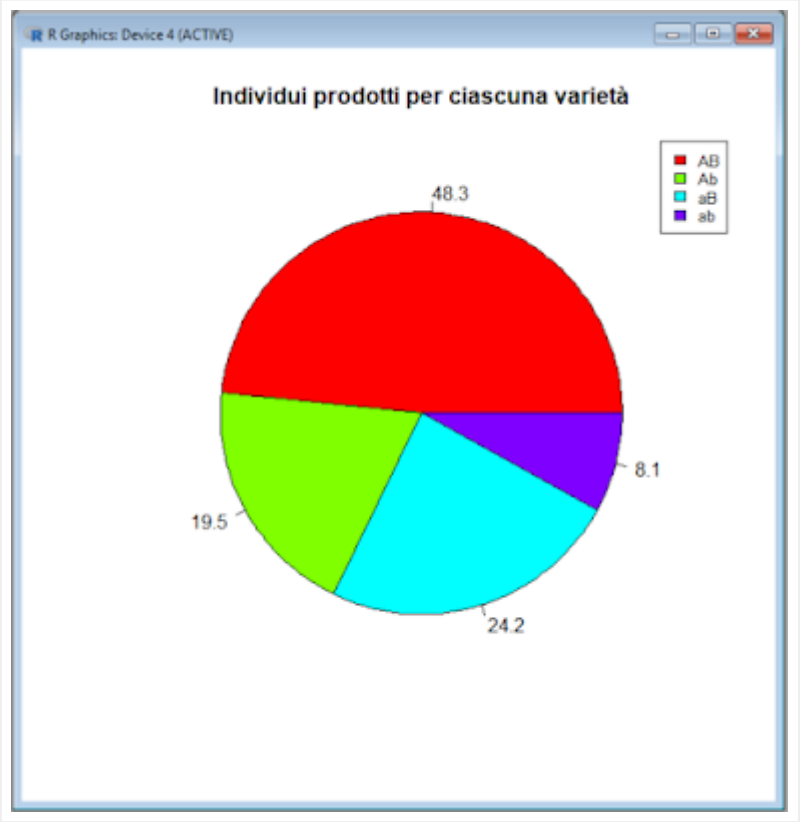
Con questo script viene generato un grafico a torta con i valori originali trasformati in percentuale e con una legenda esterna. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GRAFICO A TORTA
#
val <- c(72, 29, 36, 12) # valori
eti <- c("AB","Ab","aB","ab") # etichette
percent <- round(100*val/sum(val), 1) # percentuale per ciascun valore
#
windows() # apre una nuova finestra
pie(val, labels = percent, main = "Individui prodotti per ciascuna varietà", col =
rainbow(length(val))) # grafico a torta con percentuali e legenda esterna
legend("topright", eti, cex = 0.8, fill = rainbow(length(val))) # riporta la legenda
#
```

Con le prime tre righe di codice sono generati:
→ l'array **val** che contiene i valori corrispondenti alle quattro "fette" della torta;
→ l'array **eti** che contiene le etichette/descrizioni associate alle "fette";
→ l'array **percent** che contiene i valori dell'array **val** trasformati in percentuale (ciascun valore **val** viene diviso per la somma totale dei valori **sum(val)** e moltiplicato per 100 (**100***), quindi il risultato viene arrotondato a un decimale con l'argomento **, 1** e viene infine salvato (**<-**) nell'array **percent**).

Dopo avere aperto alla quarta riga una nuova finestra grafica con la funzione **windows()**, alla quinta riga la funzione **pie()** traccia un grafico a torta impiegando i seguenti argomenti:
→ i valori **val** da rappresentare;
→ le relative etichette **label** con i valori percentuali **percent** calcolati in precedenza;
→ il titolo del grafico **main**;
→ il colore **col** che viene cambiato a ciascuna "fetta" con la funzione **rainbow()** e impiegando tanti colori quante sono le fette con la funzione **lenght(val)**.

Infine viene riportata la legenda con la funzione **legend()** che prevede i seguenti argomenti:
→ **"topright"** per la posizione in alto a destra;
→ **eti** per le etichette da impiegare;
→ **cex = 0.8** per le dimensioni;
→ **fill** per i colori che riprendono ovviamente quelli definiti nella funzione **pie()**.

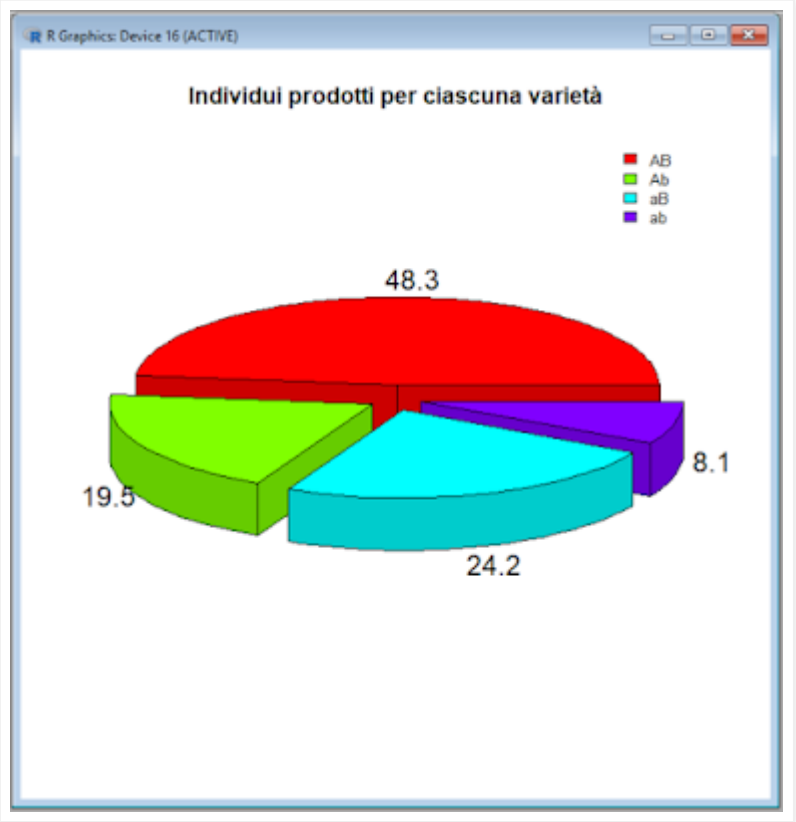


Oltre ai grafici a torta tradizionali è possibile generare grafici a torta 3D mediante un pacchetto aggiuntivo, il pacchetto **plotrix**, che deve essere scaricato dal **CRAN.**

In quest'altro script le novità rispetto al precedente sono:
→ l'impiego del pacchetto **plotrix**;
→ la funzione **pie3D()** in sostituzione della funzione **pie()**;
→ l'argomento **explode=0.1** che consente di variare quanto le fette sono distanziate l'una dall'altra (è possibile scegliere non solo valori superiori ma anche valori inferiori a **0.1**);
→ l'argomento **bty="n"** che consente di eliminare il riquadro delle didascalie che era presente nella figura precedente.

Se non l'avete già fatto, prima di eseguire lo script dovete scaricare e installare il pacchetto **plotrix**. Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GRAFICO A TORTA 3D
#
library(plotrix) # carica il pacchetto
#
val <- c(72, 29, 36, 12) # valori
eti <- c("AB","Ab","aB","ab") # etichette
percent <- round(100*val/sum(val), 1) # percentuale per ciascun valore
#
windows() # apre una nuova finestra
pie3D(val, labels = percent, explode = 0.1, main = "Individui prodotti per ciascuna varietà", col = rainbow(length(val))) # grafico a torta 3D con percentuali e legenda esterna
legend("topright", eti, cex = 0.8, fill = rainbow(length(val)), bty="n") # riporta la legenda
#
```



Altre modalità di rappresentazione dei grafici a torta sono possibili impiegando il pacchetto **plotly** [3] e il pacchetto **ggplot2** [4].

---

[1] "*Percentages ... can also be expressed using a pie-chart, but since the human eye is very poor at comparing angles, we do not recommend these for display purposes*". Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 54.

[2] Vedere il post: Test chi-quadrato 1 riga · n colonne.

[3] Vedere: Pie Charts in R. How to make pie charts in R using plotly. URL consultato il 9/12/2018.

[4] Vedere: ggplot2 pie chart in STDHA. *Statistical tools for high-throughput data analysis.* URL consultato il 9/12/2018.

unedì 10 dicembre 2018

Grafici a barre [1]

Con i grafici a torta, i **grafici a barre** sono i più adatti a rappresentare i risultati dei conteggi. Riprendiamo i dati di Snedecor relativi ai decessi avvenuti in un gruppo di non fumatori e in un gruppo di fumatori di pipa [1], trasformandoli ora in percentuale:

Esito	Non_fumatori	Fumatori_di_pipa
Deceduti	11.0	13.4
Viventi	89.0	86.6

Questo script permette di generare due differenti **grafici a barre** per questi dati. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GRAFICI A BARRE AFFIANCATE
#
cells <- c(11.0,13.4,89.0,86.6) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Deceduti", "Viventi") # genera l'array rnames con i nomi delle righe
cnames <- c("Non_fumatori", "Fumatori_di_pipa") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la matrice dati
#
mydata # mostra la matrice dati
t(mydata) # mostra la matrice dati trasposta
#
windows() # apre una nuova finestra
barplot(mydata,beside=TRUE, legend=TRUE, ylim=c(0,150), col=c("darkblue","red"), ylab="Frequenze osservate", xlab="", main="Grafico a barre matrice originale") # grafico a barre con la matrice originale
#
windows() # apre una nuova finestra
barplot(t(mydata),beside=TRUE, legend=TRUE, ylim=c(0,150), col=c("darkblue","red"), ylab="Frequenze osservate", xlab="", main="Grafico a barre matrice trasposta") # grafico a barre con la matrice trasposta
#
```

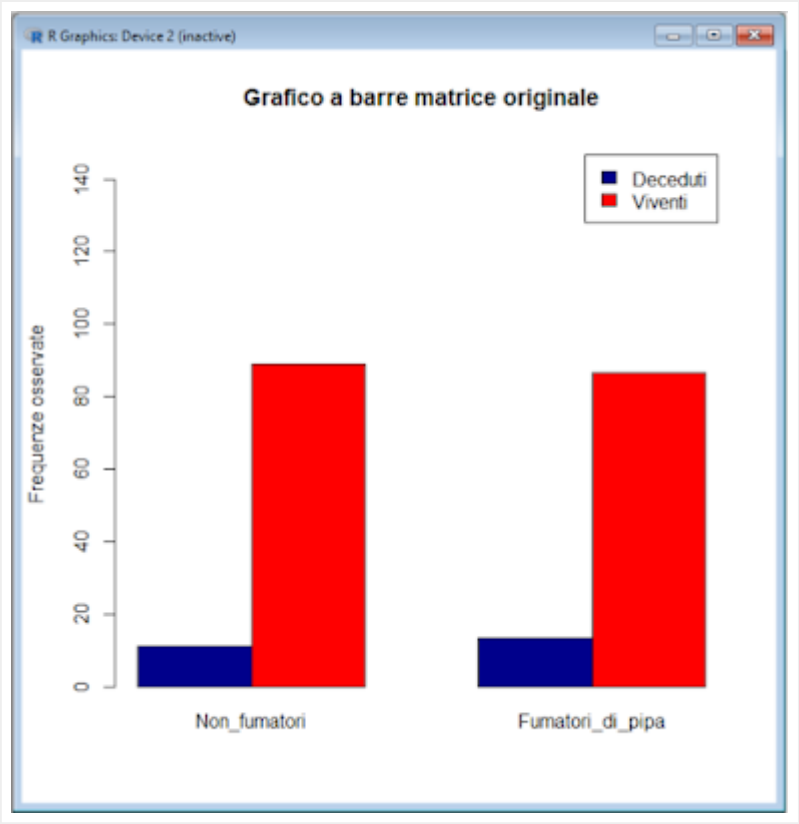
Nelle prime tre righe sono generati con la funzione **c()**:
→ il vettore che contiene i quattro dati (che come si vede devono essere inseriti in sequenza leggendoli dalla tabella da sinistra a destra e dall'alto in basso) che sono salvati nell'oggetto **cells**;
→ il vettore che contiene i nomi delle righe, salvato nell'oggetto **rnames**;
→ il vettore che contiene i nomi delle colonne, salvato nell'oggetto **cnames**.

I tre vettori sono combinati a formare la matrice dei dati mediante la funzione **matrix()** che impiega gli argomenti che indicano:
→ l'oggetto/vettore contenente i dati (**cells**);
→ il numero di righe (**nrow**) e il numero di colonne (**ncol**) della matrice;
→ la modalità di riempimento della matrice, che deve essere riempita per righe (**byrow=TRUE**) quindi da sinistra a destra e dall'alto in basso;
→ i nomi da assegnare alle righe e alle colonne (**dimnames=list(rnames, cnames)**).

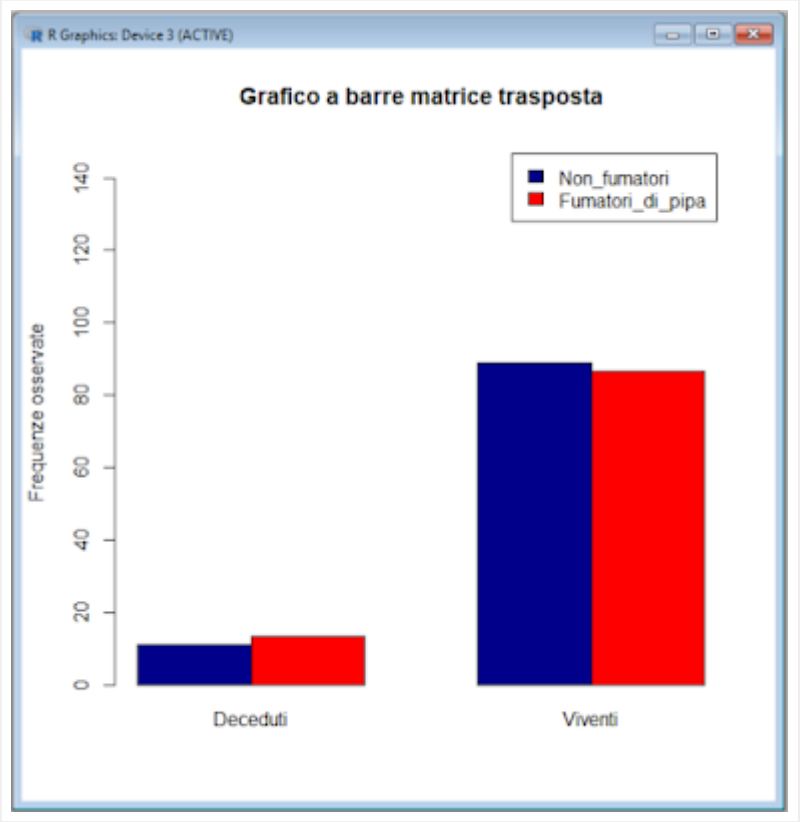
A questo punto per fare il punto della situazione sono mostrate sia la matrice dati **mydata** sia la sua matrice trasposta **t(mydata)** [2]:

```
> mydata # matrice dati
          Non_fumatori Fumatori_di_pipa
Deceduti            11              13.4
Viventi             89              86.6
> t(mydata) # matrice trasposta
                 Deceduti Viventi
Non_fumatori         11.0    89.0
Fumatori_di_pipa     13.4    86.6
```

Viene quindi aperta una nuova finestra con la funzione **windows()** e viene generato un primo grafico a barre con la funzione **barplot()** [3] impiegando questi argomenti:
→ per i dati viene impiegata la matrice dati originale **mydata**;
→ l'argomento **beside=TRUE** specifica che le barre sono affiancate e non sovrapposte in pila;
→ con l'argomento **legend=TRUE** viene generata una legenda;
→ con l'argomento **ylim=c(0,150)** sono impostati il valore minimo e il valore massimo per l'asse delle y;
→ con l'argomento **col** è impostato il colore delle barre.



Viene quindi aperta una nuova finestra con la funzione **windows()** una nuova finestra (attenzione al fatto che ogni nuova finestra grafica si sovrappone alla precedente, spostatela per visualizzarle entrambe) e viene generato con la funzione **barplot()** un secondo grafico a barre con gli stessi argomenti del precedente ma impiegando i dati della matrice trasposta **t(mydata)**.



Riprendiamo ora i dati sui quali è stato effettuato in precedenza il test di McNemar [4]. Nella sperimentazione 250 pazienti sofferenti di artrite erano sottoposti al trattamento con il farmaco A e al trattamento con il farmaco B. Era stato poi rilevato il grado di soddisfazione di ciascuno di essi rispetto all'uno e all'altro trattamento:

Esito	Soddisfatto_da_A	Non_soddisfatto_da_A
Soddisfatto_da_B	150	20
Non_soddisfatto_da_B	30	50

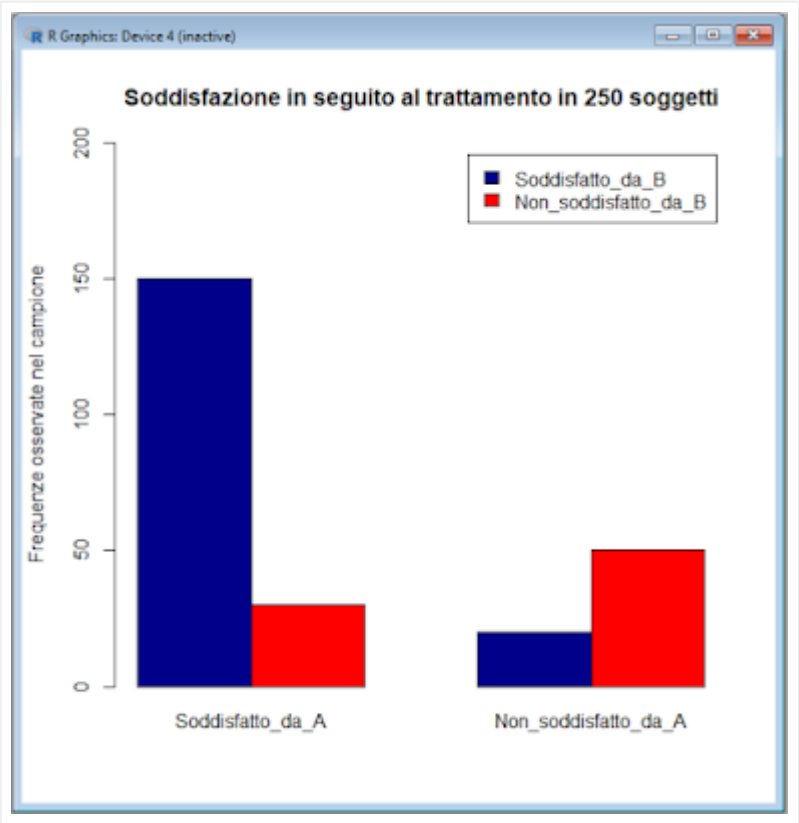
In totale 150 soggetti si erano mostrati soddisfatti di entrambi i trattamenti, 50 avevano espresso insoddisfazione per entrambi, mente altri 50 si sono mostrati insoddisfatti 20 dell'uno e 30 dell'altro. In questo caso è inutile trasformare i dati in percentuali, in quanto le proporzioni non cambierebbero.

Questo script permette di generare per questi dati due differenti grafici a barre, il primo a barre affiancate e e il secondo a barre sovrapposte. Copiatelo e incollatelo nella `Console di R` e premete ↵ `Invio`:

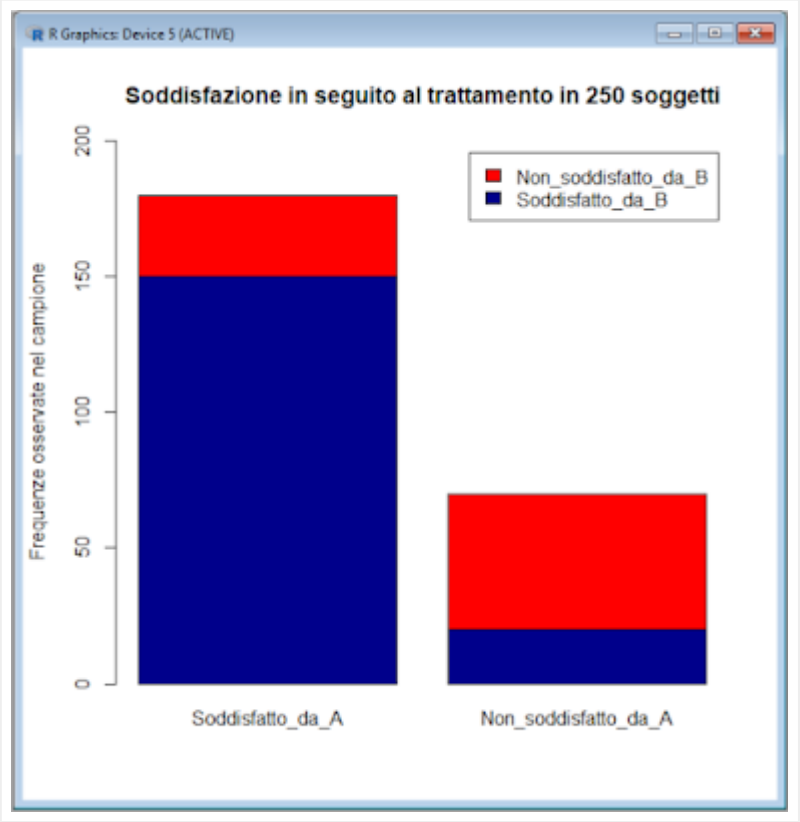
```
# GRAFICI A BARRE AFFIANCATE E A BARRE SOVRAPPOSTE
#
cells <- c(150,20,30,50) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Soddisfatto_da_B", "Non_soddisfatto_da_B") # genera l'array rnames con i nomi delle righe
cnames <- c("Soddisfatto_da_A", "Non_soddisfatto_da_A") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la matrice dati
#
mydata # mostra la matrice dati
#
windows() # apre una nuova finestra
barplot(mydata,beside=TRUE, legend=TRUE, ylim=c(0,200), col=c("darkblue","red"), ylab="Frequenze osservate nel campione", xlab="", main="Soddisfazione in seguito al
```

trattamento in 250 soggetti") # i risultati sono arricchiti con un grafico a barre
#
windows() # apre una nuova finestra
barplot(mydata,beside=FALSE,legend=TRUE, ylim=c(0,200), col=c("darkblue","red"), ylab="Frequenze osservate nel campione", xlab="", main="Soddisfazione in seguito al trattamento in 250 soggetti") # i risultati sono arricchiti con un grafico a barre
#

Qui a parte i diversi limiti **ylim=c(0,200)** per la scala dell'asse delle y non compare nulla di rilevante rispetto allo script precedente, se non l'argomento **beside** che nel primo grafico viene posto **beside=TRUE** per generare il grafico a barre affiancate



mentre nel secondo grafico viene posto **beside=FALSE** per generare un grafico a barre sovrapposte



Notare come queste rappresentazioni grafiche sono complementari all'analisi statistica presentata per questi casi. Nella seconda parte dei grafici a barre vedremo altri esempi utili con script che possono essere salvati per essere riadattati al bisogno.

---

[1] Vedere il post test chi-quadrato 2 righe · 2 colonne.

[2] Digitate **help(t)** nella Console di R per la documentazione della funzione **t()**.

[3] Digitate **help(barplot)** nella Console di R per la documentazione della funzione **barplot()**.

[4] Vedere il post Test di McNemar.

lunedì 10 dicembre 2018

Grafici a barre [2]

Dopo la prima parte dedicata ai grafici a barre [1] continuiamo ora estendendo la rappresentazione a un numero maggiore di dati con l'esempio tratto da Marubini [2] relativo al grado di protezione rilevato per cinque differenti vaccini contro l'influenza. Questi sono i dati espressi in percentuale:

Esito	Vaccino_1	Vaccino_2	Vaccino_3	Vaccino_4	Vaccino_5
Influenza_si	18.1	21.0	9.1	18.3	20.0
Influenza_no	81.9	79.0	90.8	81.7	80.0

Questo script permette di generare un classico grafico a barre affiancate. Copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GRAFICO A BARRE PER L'EFFICACIA DI 5 VACCINI
#
cells <- c(18.1,21,9.1,18.3,20,81.9,79,90.9,81.7,80) # genera l'array cells con i valori di percentuale
rnames <- c("Influenza_si", "Influenza_no") # genera l'array rnames con i nomi delle righe
cnames <- c("Vaccino_1", "Vaccino_2", "Vaccino_3", "Vaccino_4", "Vaccino_5") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=5, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la matrice dati
#
mydata # mostra la matrice dati
t(mydata) # mostra la matrice dati trasposta
#
windows() # apre una nuova finestra
barplot(t(mydata), beside=TRUE, legend=TRUE, ylim=c(0,150), col=c("red","orange", "yellow", "green", "skyblue"), ylab="Frequenze osservate", xlab="Esiti delle vaccinazioni", main="Valutazione dell'efficacia di 5 vaccini influenzali") # traccia il grafico a barre verticali
#
```

Lo script segue lo schema già visto, nelle prime tre righe sono generati con la funzione **c()**:
→ il vettore che contiene i dieci dati (che come si vede devono essere inseriti in sequenza leggendoli dalla tabella da sinistra a destra e dall'alto in basso) che sono salvati nell'oggetto **cells**;
→ il vettore che contiene i nomi delle righe, salvato nell'oggetto **rnames**;
→ il vettore che contiene i nomi delle colonne, salvato nell'oggetto **cnames**.

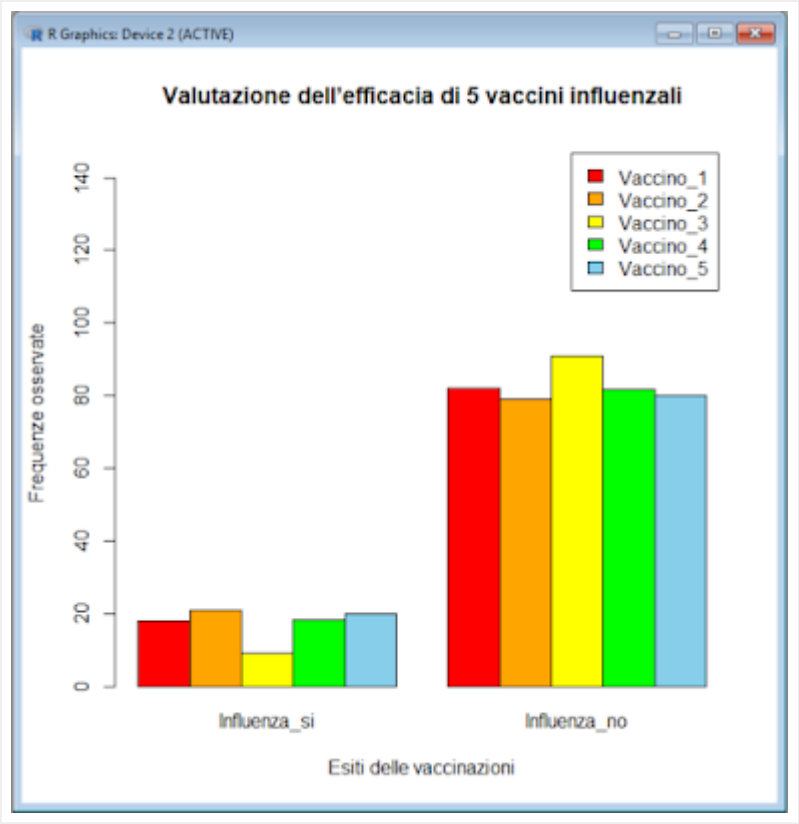
I tre vettori sono combinati a formare la matrice dei dati mediante la funzione **matrix()** che impiega gli argomenti che indicano:
→ l'oggetto/vettore contenente i dati (**cells**);
→ il numero di righe (**nrow**) e il numero di colonne (**ncol**) della matrice;
→ la modalità di riempimento della matrice, che deve essere riempita per righe (**byrow=TRUE**) quindi da sinistra a destra e dall'alto in basso;
→ i nomi da assegnare alle righe e alle colonne (**dimnames=list(rnames, cnames)**).

A questo punto per fare il punto della situazione sono mostrate sia la matrice dati **mydata** sia la sua matrice trasposta **t(mydata)** [3]:

```
> mydata # mostra la matrice dati
             Vaccino_1 Vaccino_2 Vaccino_3 Vaccino_4 Vaccino_5
Influenza_si      18.1        21       9.1      18.3        20
```

```
Influenza_no      81.9        79      90.9      81.7        80
> t(mydata) # mostra la matrice dati trasposta
          Influenza_si Influenza_no
Vaccino_1         18.1         81.9
Vaccino_2         21.0         79.0
Vaccino_3          9.1         90.9
Vaccino_4         18.3         81.7
Vaccino_5         20.0         80.0
```

Viene quindi aperta una nuova finestra con la funzione **windows()** e viene generato un primo grafico a barre con la funzione **barplot()** [4] impiegando questi argomenti:
→ per i dati viene impiegata la matrice dati trasposta **t(mydata)**;
→ l'argomento **beside=TRUE** specifica che le barre sono affiancate e non sovrapposte in pila;
→ con l'argomento **legend=TRUE** viene generata una legenda;
→ con l'argomento **ylim=c(0,150)** sono impostati il valore minimo e il valore massimo per l'asse delle y;
→ con l'argomento **col** è impostato il colore delle barre.



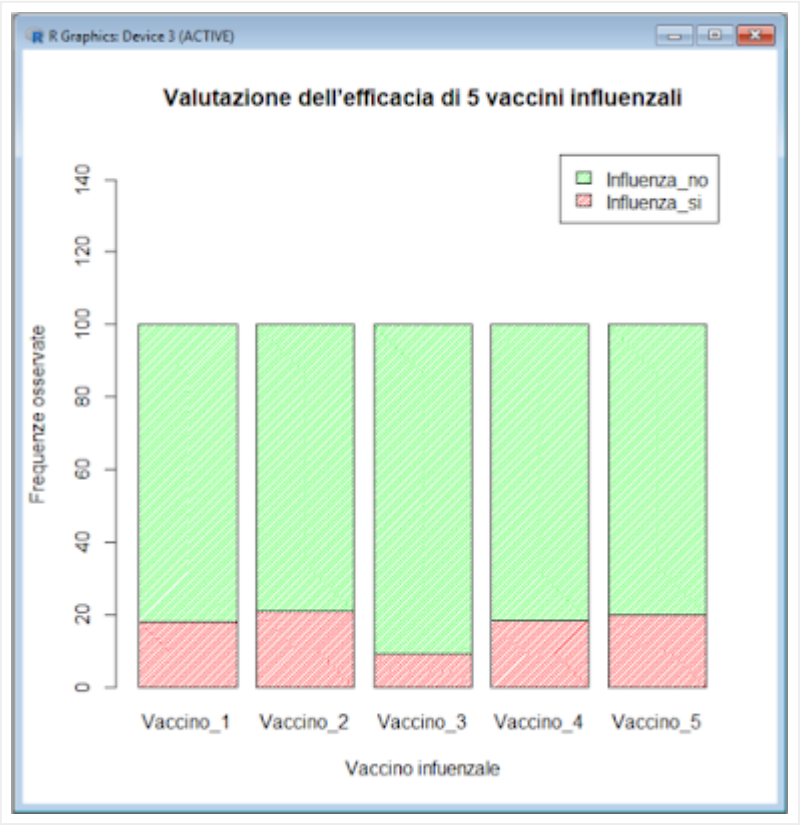
Ora copiate e incollate nella `Console di R` queste due righe di codice e premete `↵ Invio`:

```
#
windows() # apre una nuova finestra
barplot(mydata, beside=FALSE, legend=TRUE, ylim=c(0,150), col=c("red", "green"),
density = 40, angle = 45, ylab="Frequenze osservate", xlab="Vaccino infuenzale",
main="Valutazione dell'efficacia di 5 vaccini influenzali") # traccia il grafico a barre verticali
#
```

Con lo script dopo avere aperto una nuova finestra grafica (**windows()**) viene realizzato un nuovo grafico a barre impiegando la funzione **barplot()** con queste variazioni rispetto al precedente:
→ per i dati viene impiegata la matrice **mydata**;
→ l'argomento **beside=FALSE** specifica che le barre sono sovrapposte in pila;
→ con l'argomento **col=c("yellow", "green")** viene fornito l'array contenente i colori da applicare;

→ con gli argomenti **density = 40, angle = 45** sono specificati la densità e l'angolo delle linee colorate che andranno a riempire le barre.

Il risultato è ora questo grafico con le barre sovrapposte in pila (attenzione al fatto che ogni nuova finestra grafica si sovrappone alla precedente, spostatela per visualizzarle entrambe):



Finora i grafici a barre sono stati generati specificando per ciascuna barra il numero dei casi. Ma se i dati sono numerosi è possibile impiegare la funzione **table()** che genera tabelle di contingenza che riportano i dati dei conteggi effettuati per noi da **R** [5].

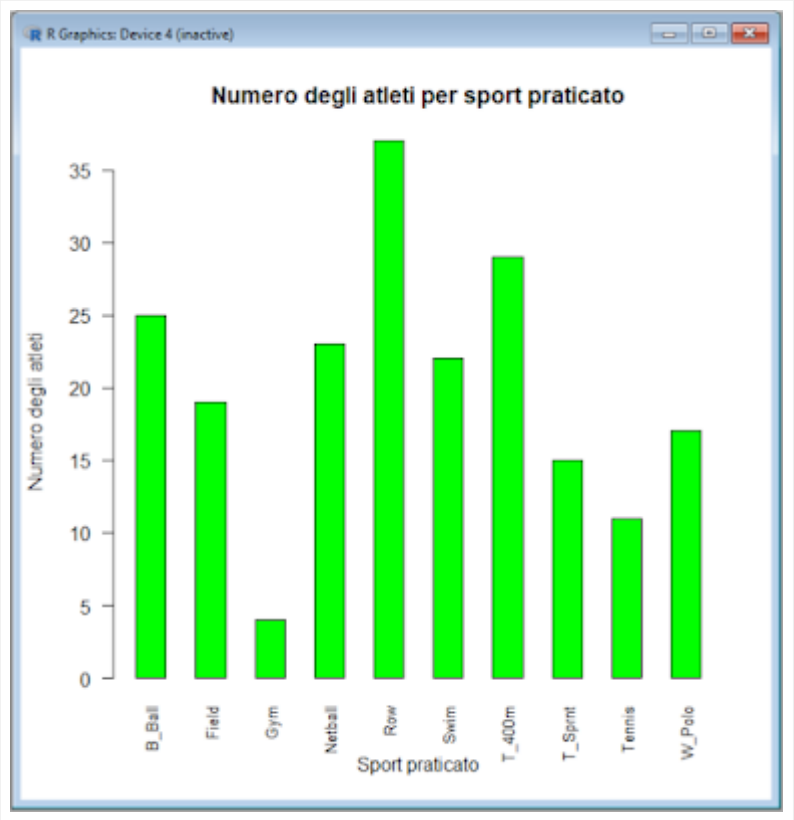
Per farlo è necessario il pacchetto **DAAG** [6]. Il pacchetto contiene tra gli altri il set di dati **ais** sul quale effettuiamo il conteggio e la rappresentazione sotto forma di grafico a barre prima del numero degli atleti per `sport` praticato e poi del numero degli atleti per `sesso` e per `sport` praticato.

Ora copiate e incollate nella `Console di R` queste righe di codice e premete ↵ `Invio`:

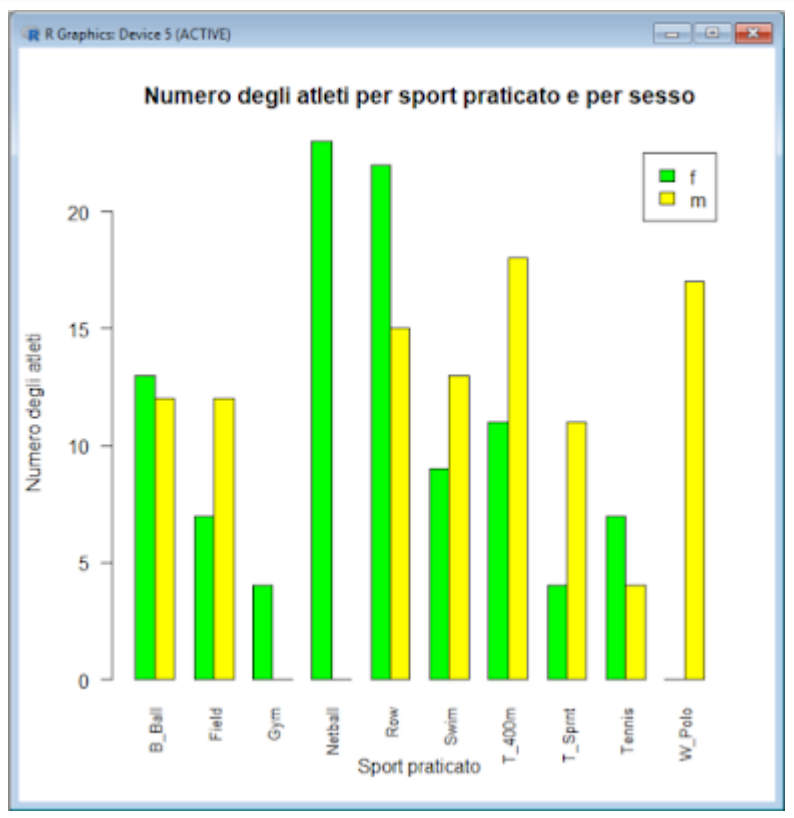
```
# GRAFICI A BARRE DA UNA TABELLA DI CONTINGENZA
#
library(DAAG) # carica il pacchetto e il set di dati ais
#
# grafico a barre per sport praticato
#
tab <- table(ais$sport) # tabula i dati per sport
windows() # apre una nuova finestra
barplot(t(tab), beside=TRUE, legend=TRUE, cex.names = 0.8, las = 2, xlab= "Sport praticato", ylab = "Numero degli atleti", main = "Numero degli atleti per sport praticato", col = "green") # grafico a barre
#
# grafico a barre per sport praticato e per sesso
#
tab <- table(ais$sport, ais$sex) # tabula i dati per sport e per sesso
windows() # apre una nuova finestra
```

barplot(t(tab), beside=TRUE, legend=TRUE, cex.names = 0.8, las = 2, xlab= "Sport praticato", ylab = "Numero degli atleti", main = "Numero degli atleti per sport praticato e per sesso", col = c("green", "yellow")) # grafico a barre
#

Questo è il grafico a barre con il numero degli atleti per sport praticato



e questo è il grafico a barre con il numero degli atleti per sport praticato ulteriormente suddiviso per sesso dell'atleta



In entrambi i casi viene impiegata la matrice trasposta **t(tab)** ma le cose più interessanti sono:
→ l'espressione **table(ais$sport)** impiegata per tabulare i dati in base alla variabile sport (lo sport praticato dall'atleta);
→ l'espressione **table(ais$sport, ais$sex)** impiegata per tabulare i dati in base alla variabile sport e suddividerli ulteriormente in base alla variabile sex (il sesso dell'atleta);
→ l'argomento **cex.names = 0.8** che rimpicciolisce le etichette che riportano lo sport corrispondente a ciascuna della barre;
→ l'argomento **las = 2** che le ruota verticalmente.

---

[1] Vedere il post Grafici a barre [1].

[2] Vedere il post Test chi-quadrato n righe · n colonne.

[3] Digitate **help(t)** nella `Console di R` per la documentazione della funzione **t()**.

[4] Digitate **help(barplot)** nella `Console di R` per la documentazione della funzione **barplot()**.

[5] Digitate **help(table)** nella `Console di R` per la documentazione della funzione **table()**.

[6] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza installare il pacchetto **DAAG**.

La distribuzione gaussiana

Innazitutto una premessa. Uno dei punti più delicati nella statistica di base è rappresentato dalla distribuzione gaussiana (l'altro punto delicato è rappresentato dal coefficiente di correlazione r).

Media, *deviazione standard*, *ANOVA*, *test t di Student*, solo per citare i principali, sono test statistici (o più semplicemente statistiche) basati sull'assunto che i dati seguano una distribuzione gaussiana.

Poiché media e deviazione standard sono i "*parametri*" che descrivono una distribuzione gaussiana, i metodi statistici che sono basati sull'assunto che i dati siano distribuiti in modo gaussiano sono detti **metodi parametrici**, in contrapposizione a quelli che non prevedono questo assunto, che sono detti **metodi non parametrici** [1, 2].

Applicare metodi parametrici a variabili che non sono distribuite in modo gaussiano è inappropriato e porta a conclusioni sbagliate. Vediamo un esempio impiegando i valori della concentrazione della ferritina nel sangue (espressa in µg/L) rilevati in 202 atleti australiani.

Copiate il testo che segue, incollatelo nella `Console di R` e premete `↲ Invio`:

x <- c(60, 68, 21, 69, 29, 42, 73, 44, 41, 44, 38, 26, 30, 48, 30, 29, 43, 34, 53, 59, 43, 40, 92, 48, 77, 71, 37, 71, 73, 85, 64, 19, 39, 41, 13, 20, 59, 22, 30, 78, 21, 109, 102, 71, 64, 68, 78, 107, 39, 58, 127, 102, 86, 40, 50, 58, 33, 51, 82, 25, 86, 22, 30, 27, 60, 115, 124, 54, 29, 164, 50, 36, 40, 62, 90, 12, 36, 45, 43, 51, 22, 44, 26, 16, 58, 46, 43, 34, 41, 57, 73, 88, 182, 80, 88, 109, 69, 41, 66, 63, 34, 97, 55, 76, 93, 46, 155, 99, 35, 124, 176, 107, 177, 130, 64, 109, 125, 150, 115, 89, 93, 183, 84, 70, 118, 61, 72, 91, 58, 97, 110, 72, 36, 53, 72, 39, 61, 49, 35, 8, 106, 32, 41, 20, 44, 103, 50, 41, 101, 73, 56, 74, 58, 87, 139, 82, 87, 80, 67, 132, 43, 212, 94, 213, 122, 76, 53, 91, 36, 101, 184, 44, 66, 220, 191, 40, 189, 141, 212, 97, 53, 126, 234, 50, 94, 156, 124, 87, 97, 102, 55, 117, 52, 133, 214, 143, 65, 90, 38, 122, 233, 32)

Ora calcoliamo su questi dati la classica misura di posizione, la *media*, e la classica misura di dispersione dei dati, la *deviazione standard* (ds). Digitate o copiate e incollate nella `Console di R` queste due righe e premete `↲ Invio`:

mean(x) # calcola la media della ferritina
sd(x) # calcola la deviazione standard della ferritina

Questo è il risultato:

```
> mean(x) # calcola la media della ferritina
[1] 76.87624
> sd(x) # calcola la deviazione standard della ferritina
[1] 47.50124
```

Comunicate i risultati di questa ricerca a un vostro amico, il quale vi fa notare quanto segue:
→ in base alle proprietà della distribuzione gaussiana sappiamo che tra la `media - 1.96 · ds` e la `media + 1.96 · ds` si deve trovare il `95%` dei dati campionari;
→ il valore della media della ferritina (arrotondato) è `76.9` µg/L;
→ la deviazione standard della ferritina è `47.5` µg/L;
→ moltiplicando `47.5` µg/L per `1.96` otteniamo `93.1` µg/L.
→ dal fatto che `76.9 - 93.1 = -16.2` µg/L e `76.9 + 93.1 = 170` µg/L dovremmo dedurre che il `95%` dei valori di ferritina rilevati negli atleti australiani cadeva nell'intervallo

`-16.2 ÷ 170 µg/L`

e pertanto concludere che in una certa quota degli atleti la concentrazione nel sangue della ferritina aveva valori inferiori a zero.

Voi ci restate malissimo, ma il punto è chiaro. Un evidente controsenso consente di individuare un grave errore nella metodologia statistica impiegata: non si possono applicare media e deviazione standard a dati che non sono distribuiti in modo gaussiano - e che questo sia proprio quello che accade nel caso della ferritina lo vedete seguendo il percorso logico indicato in [3].

Nel caso delle statistiche elementari, l'alternativa a media e deviazione standard è rappresentata dalle statistiche non parametriche mediana e deviazione assoluta mediana (MAD), dalla distanza interquartile e dai quantili non parametrici (quartili, decili, percentili).

Ma c'è un altro aspetto interessante che riguarda le statistiche non parametriche. Prendiamo questi dati

8	6	11	7	9	14	3

la cui media è `8.3`, quindi ordiniamoli in ordine crescente

3	6	7	8	9	11	14

per calcolarne la mediana che è `8` (il valore centrale nella lista dei dati ordinati in ordine crescente).

Supponiamo ora di aggiungere una nuova osservazione che si discosta molto dalle precedenti ma di non avere ragioni plausibili per scartarla

8	6	11	7	9	14	3	45

Calcoliamo di nuovo la media, che ora è diventata `12.9`, quindi di nuovo ordiniamo i dati in ordine crescente

3	6	7	8	9	11	14	45

per calcolarne la mediana, che ora è diventata `(8 + 9) / 2 = 8.5`.

Mentre con l'inserimento del nuovo dato la media è passata da 8.3 a 12.9, la mediana è passata da 8 a 8.5. Il fatto che la mediana sia una misura di posizione **più robusta** nei confronti di dati estremi rispetto alla media è un altro argomento a favore dell'impiego delle statistiche non parametriche [1, 2].

L'esempio della ferritina conferma la necessità che le statistiche parametriche siano impiegate solamente dopo che una attenta valutazione dei dati ha confermato che questi sono distribuiti in modo gaussiano.

Arriviamo quindi al tema. Mentre per la *teoria* della distribuzione gaussiana si rimanda ovviamente ai test di statistica, vediamo alcuni possibili modi per impiegare le funzioni di **R** nella valutazione dell'assunto che i dati siano distribuiti in modo gaussiano.

Le funzioni sono applicate a una **distribuzione gaussiana**, in modo da rapprsentare il quadro di riferimento da punto di vista numerico e da punto di vista grafico, quindi nell'ordine vediamo:
→ i test di normalità dei dati;
→ il confronto tra media e mediana (in un distribuzione gaussiana devono essere uguali);

→ il confronto tra deviazione standard e MAD (in un distribuzione gaussiana devono essere uguali);
→ il confronto tra percentili parametrici e non parametrici (in un distribuzione gaussiana devono essere uguali);
→ un istogramma e un kernel density plot dei dati campionari (in un distribuzione gaussiana devono essere simmetrici);
→ la distribuzione campionaria (deve sovrapporsi alla corrispondente curva gaussiana teorica);
→ la distribuzione cumulativa dei dati campionari (deve sovrapporsi alla funzione di distribuzione cumulativa teorica);
→ i quantili campionari (devono essere allineati sulla retta della distribuzione teorica).

Cinquemila valori di una distribuzione gaussiana con media `0` e deviazione standard `1` sono generati con la funzione **rnorm()** [4] e salvati nel file `gauss.csv` con questo script, copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# SALVA I DATI DI UNA DISTRIBUZIONE GAUSSIANA IN UN FILE CSV
#
mydata <- rnorm(5000, mean = 0, sd = 1) # genera cinquemila valori con media=0, ds=1
#
write.table(mydata,file="C:/Rdati/gauss.csv",sep=";", dec=",", quote=FALSE,
row.names=FALSE) # esporta i dati in un file di testo .csv sostituendo il punto (.) con la virgola (,)
#
```

Questo è quindi il nostro campione, che ci consente di illustrare cosa ci si deve attendere dall'analisi di una variabile se questa è distribuita in modo gaussiano.

Se non l'avete già fatto, prima di continuare dovete scaricare e installare dal CRAN il pacchetto **moments** e il pacchetto **nortest**.

I prossimi script impiegano i dati salvati nel file `gauss.csv` [5].

Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# TEST DI NORMALITA' (GAUSSIANITA')
#
library(moments) # carica il pacchetto
library(nortest) # carica il pacchetto
data <- read.table("C:/Rdati/gauss.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata <- unlist(data) # li converte in formato numerico
#
# asimmetria, curtosi e altri quattro test di normalità
#
agostino.test(mydata) # test di D'Agostino per il coefficiente di asimmetria
anscombe.test(mydata) # test di Anscombe-Glynn per il coefficiente di curtosi
ad.test(mydata) # test di Anderson-Darling per la gaussianità
cvm.test(mydata) # test di Cramer-von Mises per la gaussianità
pearson.test(mydata) # test chi-quadrato di Pearson per la gaussianità
sf.test(mydata) # test di Shapiro-Francia per la gaussianità
#
```

Come atteso tutti i test di normalità confermano per i dati una distribuzione gaussiana [3]:

```
> agostino.test(mydata) # test di D'Agostino per il coefficiente di asimmetria

        D'Agostino skewness test

data:  mydata
```

```
skew = -0.0050374, z = -0.1456345, p-value = 0.8842
alternative hypothesis: data have a skewness

> anscombe.test(mydata) # test di Anscombe-Glynn per il coefficiente di curtosi

        Anscombe-Glynn kurtosis test

data:  mydata
kurt = 3.03034, z = 0.48405, p-value = 0.6283
alternative hypothesis: kurtosis is not equal to 3

> ad.test(mydata) # test di Anderson-Darling per la gaussianità

        Anderson-Darling normality test

data:  mydata
A = 0.13271, p-value = 0.9808

> cvm.test(mydata) # test di Cramer-von Mises per la gaussianità

        Cramer-von Mises normality test

data:  mydata
W = 0.019132, p-value = 0.9754

> pearson.test(mydata) # test chi-quadrato di Pearson per la gaussianità

        Pearson chi-square normality test

data:  mydata
P = 47.14, p-value = 0.8452

> sf.test(mydata) # test di Shapiro-Francia per la gaussianità

        Shapiro-Francia normality test

data:  mydata
W = 0.99982, p-value = 0.9497
```

In una distribuzione gaussiana statistiche parametriche e statistiche non parametriche devono fornire gli stessi risultati. Copiate quindi questo script, incollatelo nella `Console di R` e premete ↲ `Invio`:

```
# TEST AGGIUNTIVI DI NORMALITA' (GAUSSIANITA')
#
data <- read.table("C:/Rdati/gauss.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata <- unlist(data) # li converte in formato numerico
#
# confronta media e mediana
#
media <- mean(mydata) # calcola la media
mediana <- median(mydata) # calcola la mediana
data.frame(media, mediana) # le mette a confronto
#
# confronta deviazione standard e MAD
#
ds <- sd(mydata) # calcola la deviazione standard
```

mad <- mad(mydata) # calcola la Median Absolute Deviation (about the median) o MAD
data.frame(ds, mad) # le mette a confronto
#
confronta quantili parametrici e non parametrici
#
qpar <- round(qnorm(c(seq(0.025, 0.975, 0.025)), mean=0, sd=1), digits=2) # calcola i quantili parametrici
qnon <- round(quantile(mydata, probs=seq(0.025, 0.975, 0.025)), digits=2) # calcola i quantili non parametrici
data.frame(qnon, qpar) # li mette a confronto
#

A meno di una differenza insignificante (dipendente dal generatore di numeri casuali di **R**) media e mediana sono uguali

```
       media      mediana
1 0.01061802 0.0009510756
```

deviazione standard e MAD [6] sono uguali

```
        ds       mad
1 1.000571 0.9948351
```

come pure i quantili parametrici e i quantili non parametrici

```
       qnon  qpar
2.5%  -1.97 -1.96
5%    -1.64 -1.64
7.5%  -1.42 -1.44
10%   -1.25 -1.28
12.5% -1.14 -1.15
15%   -1.02 -1.04
17.5% -0.92 -0.93
20%   -0.83 -0.84
22.5% -0.75 -0.76
25%   -0.66 -0.67
27.5% -0.59 -0.60
30%   -0.51 -0.52
32.5% -0.44 -0.45
35%   -0.37 -0.39
37.5% -0.31 -0.32
40%   -0.24 -0.25
42.5% -0.18 -0.19
45%   -0.12 -0.13
47.5% -0.05 -0.06
50%    0.00  0.00
52.5%  0.07  0.06
55%    0.13  0.13
57.5%  0.20  0.19
60%    0.26  0.25
62.5%  0.33  0.32
65%    0.41  0.39
67.5%  0.47  0.45
70%    0.54  0.52
72.5%  0.61  0.60
75%    0.68  0.67
```

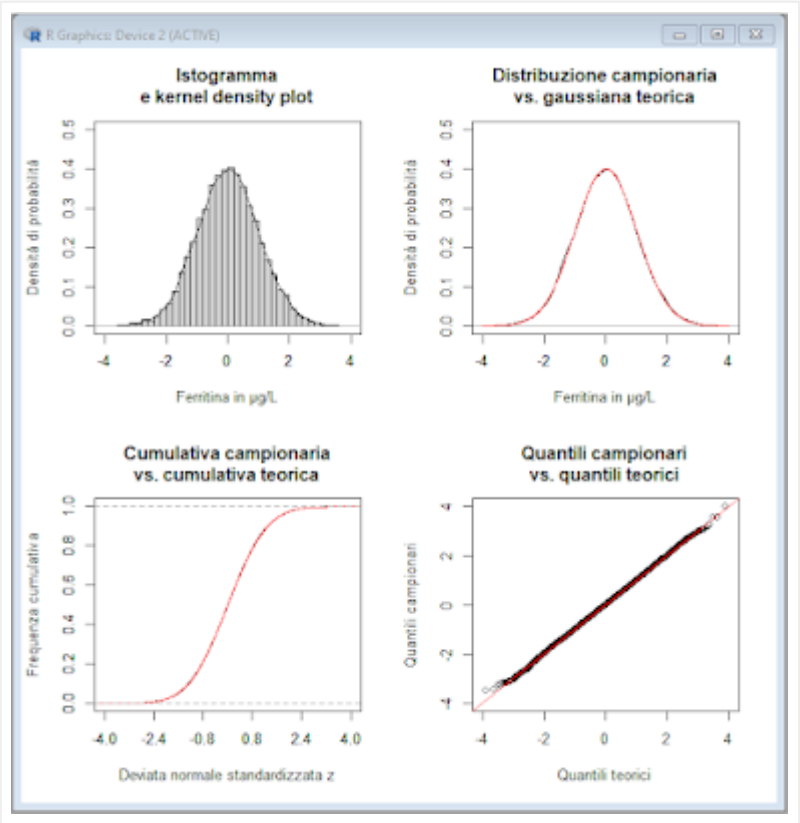
```
77.5%  0.76  0.76
80%    0.84  0.84
82.5%  0.94  0.93
85%    1.06  1.04
87.5%  1.18  1.15
90%    1.31  1.28
92.5%  1.45  1.44
95%    1.64  1.64
97.5%  1.95  1.96
```

Ai valori numerici forniti dai test di normalità e dalle statistiche aggiuntive aggiungiamo ora quattro grafici. Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# VALUTAZIONE GRAFICA DELLA NORMALITA' (GAUSSIANITA')
#
data <- read.table("C:/Rdati/gauss.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata <- unlist(data) # li converte in formato numerico
#
windows() # apre una nuova finestra
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
# istogramma e kernel density plot
#
hist(mydata, xlim=c(-4, 4), ylim=c(0, 0.5), freq=FALSE, breaks=50,
main="Istogramma\ne kernel density plot", xlab="Ferritina in µg/L", ylab="Densità di
probabilità") # traccia l'istogramma dei dati
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
plot(density(mydata, n=1000, from=-4, to=4), xlim=c(-4, 4), ylim=c(0, 0.5), yaxt="n",
col="black") # sovrappone il kernel density plot della distribuzione campionaria
#
# distribuzione campionaria vs. gaussiana teorica
#
plot(density(mydata, n=1024, from=-4, to=4), xlim=c(-4, 4), ylim=c(0, 0.5)) # traccia il
kernel density plot della distribuzione campionaria
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
x <- seq(-4, 4, length.out=1000) # calcola i valori in ascisse della gaussiana teorica
y <- dnorm(x, mean=mean(mydata), sd=sd(mydata)) # calcola i valori in ordinate della
gaussiana teorica
plot(x, y, xlim=c(-4, 4), ylim=c(0, 0.5), yaxt="n", col="red", type="l") # sovrappone la
distribuzione gaussiana teorica in colore rosso
title(main="Distribuzione campionaria\nvs. gaussiana teorica", xlab="Ferritina in µg/L",
ylab="Densità di probabilità") # aggiunge titolo e legende degli assi
#
# distribuzione cumulativa campionaria vs. distribuzione cumulativa teorica
#
par(new=FALSE, ann=TRUE) # per mostrare titolo e legende degli assi
plot(ecdf(scale(mydata)), main="Cumulativa campionaria\nvs. cumulativa teorica",
xlab="Deviata normale standardizzata z", ylab="Frequenza cumulativa", xlog = FALSE,
ylog = FALSE, xlim=c(-4, 4), ylim=c(0, 1), xaxp=c(-4, 4, 5), yaxp=c(0, 1, 5)) # traccia il
grafico della distribuzione cumulativa campionaria
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
plot(ecdf(rnorm(10000, mean=0, sd=1)), col="red", xlog=FALSE, ylog=FALSE, xlim=c(-4,
4), ylim=c(0, 1), xaxp=c(-4, 4, 5), yaxp=c(0, 1, 5)) # sovrappone la distribuzione cumulativa
teorica in colore rosso
#
# quantili campionari vs. quantili teorici
```

#
par(new=FALSE, ann=TRUE) # per mostrare titolo e legende degli assi
qqnorm(scale(mydata), main="Quantili campionari\nvs. quantili teorici", xlab="Quantili teorici", ylab="Quantili campionari", xlim=c(-4, 4), ylim=c(-4, 4)) # traccia il grafico della distribuzione dei quantili campionari
abline (0, 1, col="red") # sovrappone la distribuzione dei quantili teorica in colore rosso
#

Questi sono i quattro grafici risultanti, con i dati campionari (in nero) che si sovrappongono perfettamente alla distribuzione teorica (in rosso), quello che aspettiamo appunto nel caso di dati distribuiti in modo gaussiano:



Con la funzione **par(mfrow=c(2,2))** la finestra grafica viene suddivisa in quattro quadranti, che verranno riempiti da sinistra a destra e dall'alto verso il basso dai quattro grafici generati con i successivi blocchi di codice, migliorando la sintesi dei risultati.

Per le funzioni qui riportate [7] sono previsti molti altri possibili argomenti, per i quali si rimanda alla documentazione di **R**. Quelli che vi capiterà di utilizzare più frequentemente, a parte ovviamente la variabile da analizzare, che è sempre il primo argomento in ogni funzione, sono:
→ **main**, il titolo del grafico, che compare in alto;
→ **xlab**, l'etichetta dell'asse delle ascisse x;
→ **ylab**, l'etichetta dell'asse delle ordinate y;
→ **xlim**, che indica limite inferiore e limite superiore della scala da applicare all'asse delle ascisse x;
→ **ylim**, che indica limite inferiore e limite superiore della scala da applicare all'asse delle ordinate y;
→ **xaxp**, che specifica il limite inferiore, il limite superiore e il numero di intervalli da applicare alla scala dell'asse delle ascisse x;
→ **yaxp**, che specifica il limite inferiore, il limite superiore e il numero di intervalli da applicare alla scala dell'asse delle ordinate y.

Per riutilizzare gli script è sufficiente importare i vosti dati nell'oggetto **mydata**.

Conclusione: l'analisi statistica e grafica qui effettuata fornisce un esempio didattico di quello che tipicamente ci si deve attendere dall'analisi di una variabile se questa è distribuita in modo gaussiano.

---

[1] Siegel S, Castellan NJ. *Statistica non parametrica*. McGraw-Hill Libri Italia, Milano, 1992, ISBN 88-386-0620-X.

[2] Maritz JS. *Distribution-free statistical methods*. Chapman and Hall, New York, 1984, ISBN 0-412-25410-7.

[3] Il percorso logico prevede, per il calcolo delle statistiche elementari di una singola variabile, i seguenti passi:
→ analisi esplorativa dei dati;
→ esecuzione dei test di normalità (gaussianità), per valutare se i dati seguono una distribuzione gaussiana;
→ calcolo delle statistiche elementari parametriche (media, deviazione standard, varianza, quantili parametrici) se i dati seguono una distribuzione gaussiana;
→ calcolo delle statistiche elementari non parametriche (mediana, deviazione assoluta mediana o MAD, quartili e altri quantili non parametrici) se i dati non seguono una distribuzione gaussiana.

[4] Digitate **help(rnorm)** nella `Console di R` per la documentazione della funzione **rnorm()**.

[5] Nuovi valori sono generati ogni volta che viene eseguita la funzione **rnorm()**, quindi se la eseguissimo più volte i risultati sarebbero sempre lievemente diversi, se non predisponessimo il file `gauss.csv`.

[6] La Median Absolute Deviation about median o MAD ovvero la deviazione assoluta mediana (dalla mediana) è l'equivalente non parametrico della deviazione standard. Vedere: Rousseeuw PJ, Croux C. *Alternatives to the Median Absolute Deviation*. Journal of the American Statistical Association 88 (424), 1273-1283, 1993. URL consultato il 04/01/2019: https://goo.gl/4Rh53b

[7] Per la documentazione delle funzioni digitare **help(***nomedellafunzione***)** nella `Console di R`. Per gli argomenti **xlog**, **ylog**, **xlim**, **ylim**, **xaxp**, **yaxp** vedere anche la funzione **par()** con **help(par)**.

giovedì 13 dicembre 2018

Correlazione e causazione

Per ridurre la componente di incertezza nelle nostre decisioni, oltre che utilizzare metodi di ragionamento "razionali", è necessario acquisire dati e informazioni "corretti". Oggi si sente spesso dire che ci si deve basare sull'evidenza. E i numeri sono per definizione portatori di "evidenza".

Ma cosa è l'evidenza? Vediamo un esempio di evidenza statistica impiegando variabili rilevate negli anni dal 1998 al 2002 dall'Istituto Nazionale di Statistica e dall'Istituto Superiore di Sanità.

La prima variabile è **Procedimenti civili di primo grado pendenti a fine anno**, e la porremo sull'asse delle ascisse.
I valori sono tratti da "*ISTAT, Annuario statistico italiano 2004, pag. 136. Tavola 6.1 - Movimento dei procedimenti civili per grado di giudizio e ufficio giudiziario - Anni 1998-2002*" [1].

La seconda variabile è **Casi [di AIDS] diagnosticati**, e la porremo sull'asse delle ordinate.
I valori sono tratti da "*Notiziario dell'Istituto Superiore di Sanità, Supplemento 1-2007, Aggiornamento dei casi di AIDS notificati in Italia e delle nuove diagnosi di infezione da HIV, pag. 4. Tabella 1 - Distribuzione annuale dei casi di AIDS, dei casi corretti per ritardo di notifica, dei decessi e del tasso di letalità*". [2]

I valori riportati in questa tabella

Anno	Procedimenti civili ... pendenti a fine anno (x)	Casi [di AIDS] diagnosticati (y)
1998	10376	2441
1999	9159	2135
2000	8290	1948
2001	7924	1812
2002	6872	1756

sono analizzati con uno script con il quale sono calcolati il coefficiente di correlazione **r** e l'equazione della retta di regressione (prima di eseguire lo script è necessario scaricare e installare il pacchetto **Hmisc**). Copiate questa prima parte dello script e incollatela nella `Console di R` e premete `↵ Invio`:

```
# r di Pearson ed equazione della retta y = a + b · x
#
x <- c(10376,9159,8290,7924,6872) # valori in ascisse
y <- c(2441,2135,1948,1812,1756) # valori in ordinate
mydata <- data.frame(x,y) # per la funzione cor()
mymatrix <- as.matrix(mydata) # per la funzione rcorr()
#
coefficients(lm(y ~ x)) # intercetta (a) e coefficiente angolare (b)
#
cor(mydata, method="pearson") # coefficiente di correlazione r
library(Hmisc) # carica il pacchetto
rcorr(mymatrix, type="pearson") # significatività di r
#
```

Questi sono i risultati:

```
> coefficients(lm(y ~ x)) # intercetta (a) e coefficiente angolare (b)
(Intercept)           x
270.6801141   0.2050304
> #
> cor(mydata, method="pearson") # coefficiente di correlazione r
          x         y
x 1.0000000 0.9748914
y 0.9748914 1.0000000
> library(Hmisc) # carica il pacchetto
Carico il pacchetto richiesto: lattice
Carico il pacchetto richiesto: survival
Carico il pacchetto richiesto: Formula
Carico il pacchetto richiesto: ggplot2

Attaching package: 'Hmisc'

The following objects are masked from 'package:base':

    format.pval, units

> rcorr(mymatrix, type="pearson") # significatività di r
     x    y
x 1.00 0.97
y 0.97 1.00

n= 5

P
  x      y
x        0.0048
y 0.0048
```

L'equazione della retta di regressione è

$$y = 270.6801141 + 0.2050304 \cdot x$$

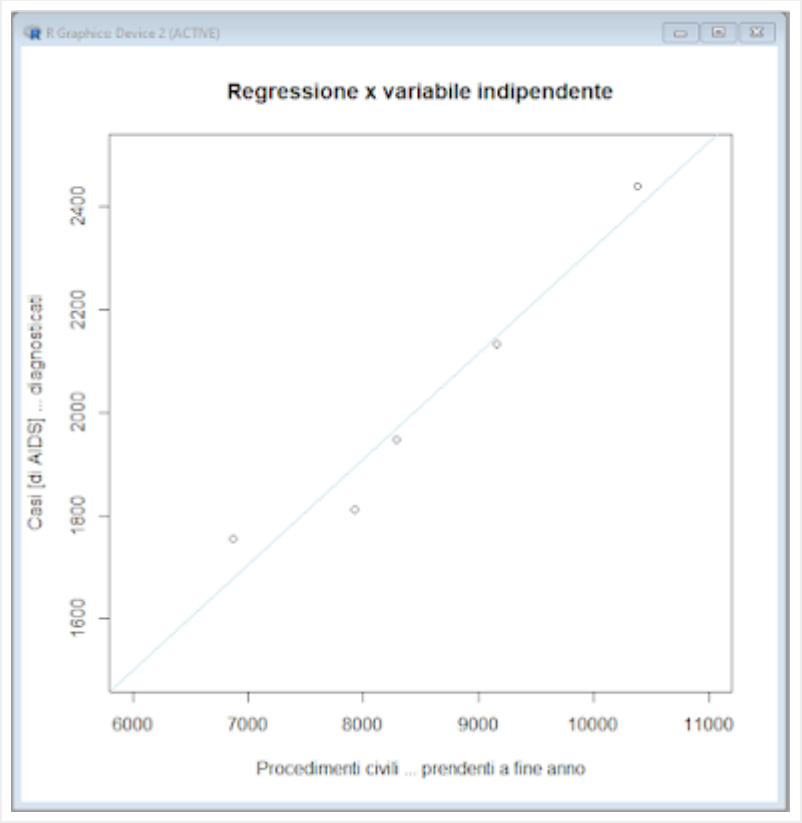
e il coefficiente di correlazione **r** è uguale a `0.9748914`. La probabilità di osservare per caso questo valore è p = `0.0048` cioè è pari allo 0.48%. Anche se assumiamo un livello di significatività dell'1%, molto più prudente del classico 5%, possiamo concludere che tra le due variabili esiste, statisticamente parlando, una **correlazione** significativa.

Le fonti sono documentate e autorevoli. Ed esiste una "*evidenza*" suffragata dai numeri che le due variabili variano congiuntamente. Tuttavia l'evidenza dei numeri è in contrasto con la ragionevolezza: perché non abbiamo un'ipotesi plausibile in grado di spiegare come all'aumentare dell'efficienza nell'amministrazione della giustizia civile la gente si ammali meno di AIDS. I due fatti sono dal punto di vista causale "evidentemente" scollegati, anche se dal punto di vista statistico "evidentemente" correlati.

Possiamo anche arricchire l'analisi statistica tracciando il grafico, copiate questa seconda parte dello script e incollatela nella `Console di R` e premete ↵ `Invio`:

```
#
windows() # apre una nuova finestra
plot(x, y, xlim = c(6000,11000), ylim = c(1500,2500), xlab="Procedimenti civili ... prendenti a fine anno", ylab="Casi [di AIDS] ... diagnosticati", main="Regressione x
```

variabile indipendente") # grafico dei dati
abline(coefficients(lm(y ~ x)), col="lightblue") # retta di regressione x variabile indipendente
#



Immagino che chi si occupa di statistica e/o dati scientifici possa aver avuto o abbia prima o poi l'occasione di vedere un grafico di questo genere associato a una affermazione di "correlazione lineare significativa".

Ma non è sempre così. E l'assunzione implicita di un rapporto di causa-effetto in seguito ad un coefficiente di correlazione significativo è un errore da evitare. Richiami all'attenzione li potete trovare in Marubini [3] e in Campbell [4], e a tutt'oggi esiste almeno un sito web interamente dedicato ad illustrare esempi grotteschi su questo tema [5].

Per un approccio razionale al tema dell'inferenza causale in statistica rimando a [6].

---

[1] *ISTAT, Annuario statistico italiano 2004*. URL consultato il 13/12/2018: https://goo.gl/3cTskK - attualmente link non disponibile.

[2] *Notiziario dell'Istituto Superiore di Sanità, Supplemento 1-2007*. URL consultato il 13/12/2018: https://goo.gl/Gn68Si - attualmente link non disponibile.

[3] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp. 89-90.

[4] "*Correlation is not causation*". In: Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 103.

[5] *Spurious correlation*s. URL consultato il 13/12/2018: https://goo.gl/8ZJHNC

[6] Judea Pearl. "*Causal inference in statistics: An overview.*" Statist. Surv. 3 96 - 146, 2009. https://doi.org/10.1214/09-SS057

sabato 15 dicembre 2018

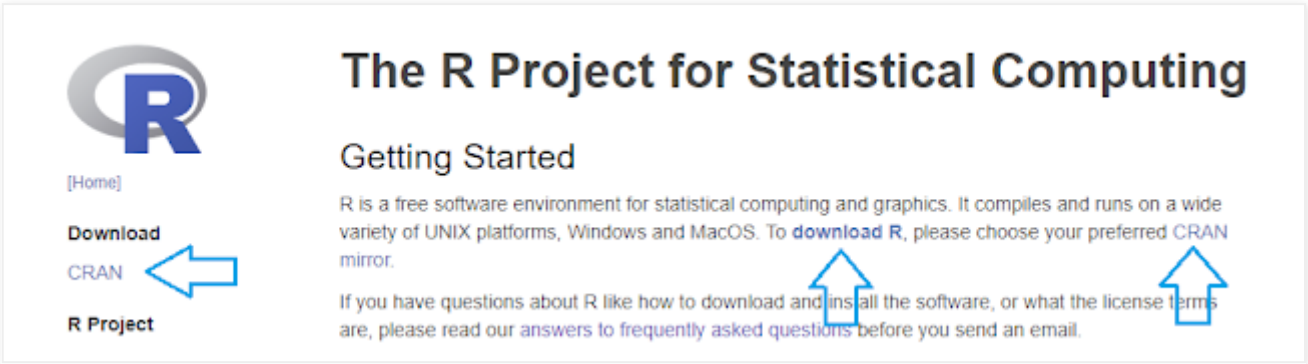
Il programma e l'interprete di comandi

R è un ambiente di sviluppo che impiega il **linguaggio R**, un linguaggio di programmazione ad oggetti orientato alla soluzione di problemi di *analisi statistica* e di *rappresentazione grafica* dei dati.

Si tratta di un *linguaggio interpretato* cioè nel quale le istruzioni in linguaggio **R** fornite dall'utilizzatore sono lette riga per riga da un `Interprete di comandi` che si incarica di trasformarle in linguaggio macchina e di farle eseguire.

Cuore dell'ambiente di sviluppo è il programma R (d'ora in poi semplicemente **R**) che:
→ mette a disposizione una serie di **funzioni** statistiche e grafiche di base che possono essere integrate e ampliate con le funzioni contenute in migliaia di *pacchetti aggiuntivi*, che offrono soluzioni di analisi statistica e rappresentazione grafica per (praticamente) qualsiasi problema;
→ consente di sviluppare nuovi **script** (impiegando le funzioni già sviluppate e contenute nell'installazione base e nei pacchetti aggiuntivi) e nuove funzioni (impiegando il linguaggio di programmazione **R**) per realizzare le elaborazioni statistiche e grafiche desiderate;
→ contiene l'`Interprete di comandi` necessario per eseguire le istruzioni in linguaggio **R** fornite dall'utilizzatore.

Per effettuare il download di **R** collegatevi alla pagina del sito ufficiale facendo click sul link [1] a fondo pagina o digitando semplicemente **R** per ricercarla su Google, quindi:
→ fate click su CRAN o su **download R** oppure su CRAN mirror (è indifferente);

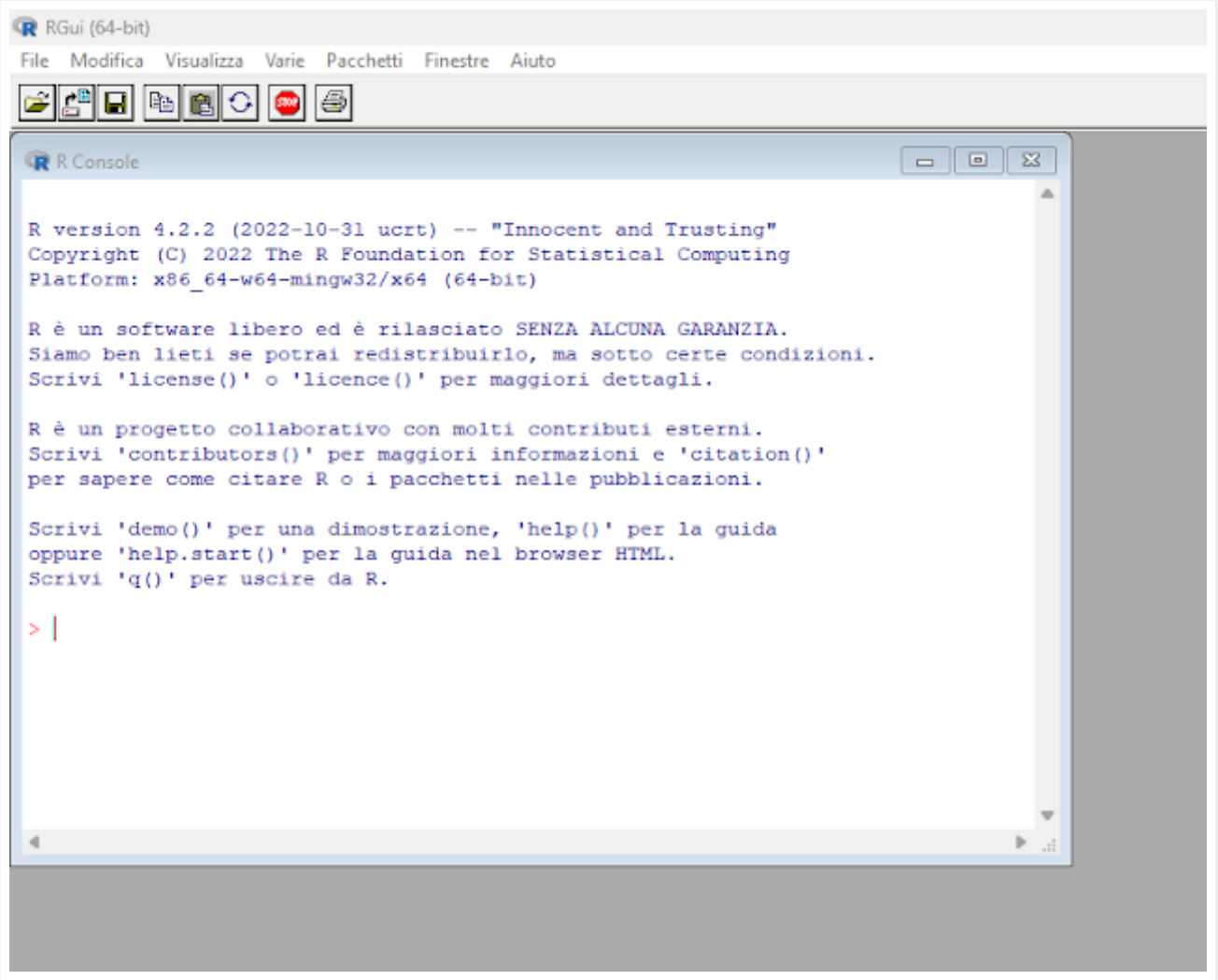


→ selezionate il server del **CRAN** (**C**omprehensive **R** **A**rchive **N**etwork) dal quale fare il download;
→ selezionate la versione di **R** da installare (per Linux, per MacOS, per Windows);
→ al termine del download eseguite la procedura di installazione conformemente al vostro sistema operativo.

Per gli approfondimenti sulle procedure di installazione potete consultare il manuale *R Installation and Administration* in versione .pdf [2] o in versione html [3].

Quando avviate il programma vi appare l'interfaccia grafica di **R** con in alto la classica `Barra dei menù`.

L'interfaccia grafica o GUI (**G**raphical **U**ser **I**nterface) di **R** (`RGui`) è limitata alla gestione delle operazioni più generali con i menù `File`, `Modifica`, `Visualizza`, `Varie`, `Pacchetti`, `Finestre`, `Aiuto` e presenta in una finestra separata la `Console di R` [4].



La `Console di R` è una semplice interfaccia a carattere o CUI (**C**haracter **U**ser **I**nterface) che fornisce l'`Interprete di comandi` e nella quale le istruzioni per effettuare le elaborazioni statistiche e grafiche dei dati sono inserite digitandole a mano lettera per lettera, parola per parola e riga per riga. Questo processo, che garantisce il massimo di flessibilità, in quanto consente di eseguire qualsiasi cosa (purché abbia un senso in **R**), è peraltro assai dispendioso quando le elaborazioni si fanno complesse.

Per elaborazioni complesse e ripetitive, cioè nella maggior parte dei casi, ci si serve degli script, cioè di blocchi di linee di codice **R**, scritti allo scopo di effettuare specifiche elaborazioni statistiche e grafiche, che vengono salvati sotto forma di file di testo che al bisogno vengono copiati e incollati nella `Console di R`, dove sono letti ed eseguiti.

Il simbolo **>** nella `Console di R` è il prompt: indica che attende che venga scritto qualcosa sulla tastiera per eseguire quello che viene scritto, ovvero che venga inserito uno script, per eseguire il codice contenuto nello script.

Per chi affronta **R** per la prima volta esempi iniziali molto semplici sono riportati nel post Eseguire uno script e nel post Pacchetti aggiuntivi di statistica e grafica.

Nota bene: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata non solo negli script ma anche nei dati e in tutto il testo.

---

[1] *The R Project for Statistical Computing*. URL consultato il 15/12/2018: https://goo.gl/MW66w1

[2] R Core Team. *R Installation and Administration*. URL consultato il 15/12/2018: https://goo.gl/kzzyXY

[3] R Core Team. *R Installation and Administration*. URL consultato il 15/12/2018: https://goo.gl/bAxtxE

[4] Le indicazioni e l'iconografia riportate in questo blog fanno riferimento alla versione di **R** per Windows.

I dati di BMI (indice di massa corporea)

L'indice di massa corporea o **BMI** [1] viene calcolato come rapporto tra il peso espresso in kg e il quadrato dell'altezza espressa in metri, ed è pertanto espresso in kg/m^2. I valori di BMI sono classificati dal punto di vista medico come segue:

Situazione peso	**Minimo (kg)**	**Massimo (kg)**
Obesità di III classe (gravissima)	> 40,00	
Obesità di II classe (grave)	35,01	40
Obesità di I classe (moderata)	30,01	35
Sovrappeso	25,01	30
Regolare	18,51	25
Leggermente sottopeso	17,51	18,5
Sottopeso	16,01	17,5
Grave magrezza (inedia)	< 16,01	

Questi sono i risultati dell'indagine **EHIS 2015** come riportati dall'Istat [2].

Tavola 6.1 - Persone di 18 anni e più per Indice di Massa Corporea (in classi), per paese europeo. Anno 2015 (a)

(per 100 persone con le stesse caratteristiche)

PAESI	INDICE MASSA CORPOREA				Totale
	Sottopeso	Normale	Sovrappeso	Obeso	
Italia	**3,3**	**51,9**	**34,1**	**10,8**	**100**
Unione europea (28 paesi)	**2,3**	**46,1**	**35,7**	**15,9**	**100**
Austria	2,4	49,6	33,3	14,7	100
Belgio	2,7	48,0	35,3	14,0	100
Bulgaria	2,2	43,8	39,2	14,8	100
Cipro	3,9	47,8	33,8	14,5	100
Croazia	1,9	40,7	38,7	18,7	100
Danimarca	2,2	50,0	32,9	14,9	100
Estonia	2,2	43,9	33,5	20,4	100
Finlandia	1,2	44,1	36,4	18,3	100
Francia	3,2	49,6	31,9	15,3	100
Germania	1,8	46,1	35,2	16,9	100
Grecia	1,9	41,3	39,4	17,3	100
Irlanda	1,9	42,3	37,0	18,7	100
Lettonia	1,7	41,8	35,2	21,3	100
Lituania	1,9	42,5	38,3	17,3	100
Lussemburgo	2,8	49,3	32,4	15,6	100
Malta	2,0	37,0	35,0	26,0	100
Olanda	1,6	49,0	36,0	13,3	100
Polonia	2,4	42,9	37,5	17,2	100
Potogallo	1,8	44,6	36,9	16,6	100
Regno Unito	2,1	42,2	35,6	20,1	100
Repubblica Ceca	1,1	42,1	37,6	19,3	100
Romania	1,3	42,9	46,4	9,4	100
Slovacchia	2,1	43,6	38,0	16,3	100
Slovenia	1,6	41,8	37,4	19,2	100
Spagna	2,2	45,4	35,7	16,7	100
Svezia	1,8	48,3	35,9	14,0	100
Ungheria	2,9	41,9	34,0	21,2	100

Fonte: Indagine "European Health Interview Survey" - Anno 2015; Eurostat database http://ec.europa.eu/eurostat/data/database.
(a) Gli indicatori sono calcolati escludendo i missing e le risposte proxy.

I dati sono impiegati come esempio in vari post. Trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un ↵ `Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

```
Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7
Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2
```

---

[1] Anche in Italia viene usualmente impiegato l'acronimo **BMI** derivato dall'inglese Body Mass Index.

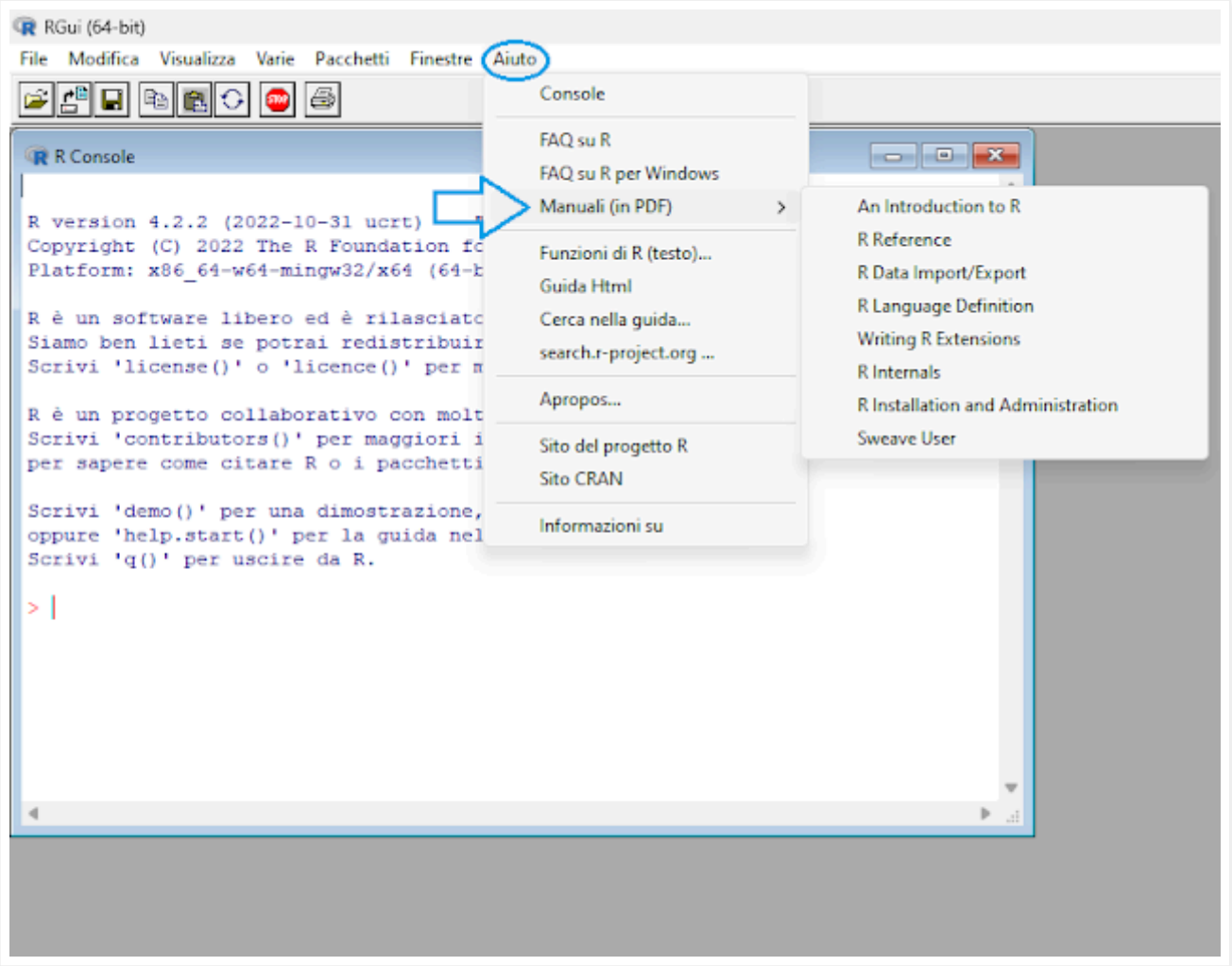
[2] *Prevenzione e stili di vita in Italia e nell'Unione Europea. Indagine EHIS 2015. Tavole*. Tavola 6.1 bmi in europa. URL consultato il 16/12/2018: https://goo.gl/ZxGi9U

domenica 16 dicembre 2018

Documentazione e aiuto di R

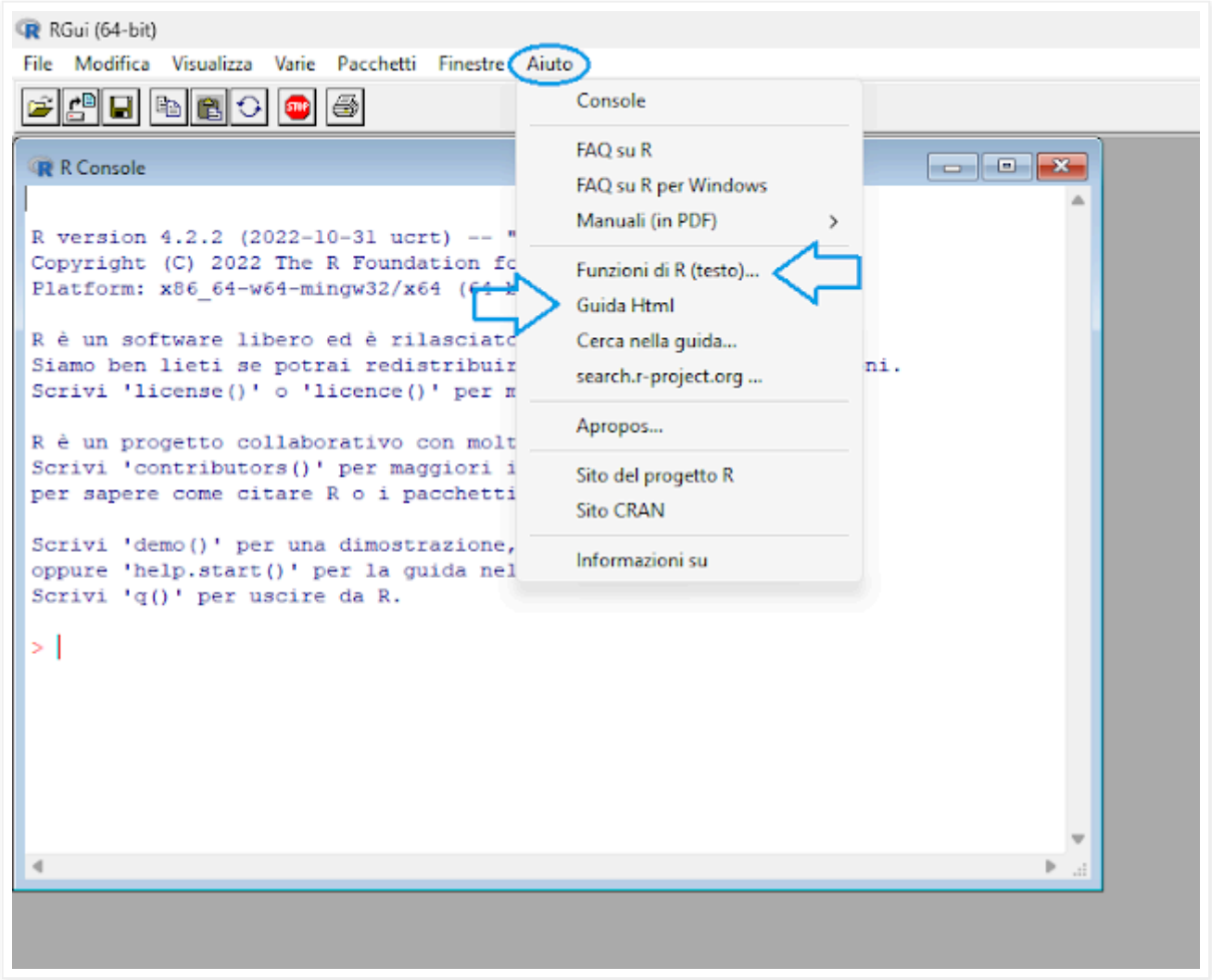
Se nella `RGui` selezionate `Aiuto` compare un menù a tendina nel quale trovate varie opzioni per accedere alla documentazione di **R**.

Se continuate a preferire la carta stampata alla voce `Manuali (in PDF) >` trovate nelle prime tre posizioni i manuali più importanti:
→ *An Introduction to R*
→ *R Reference*
→ *R Data Import/Export*



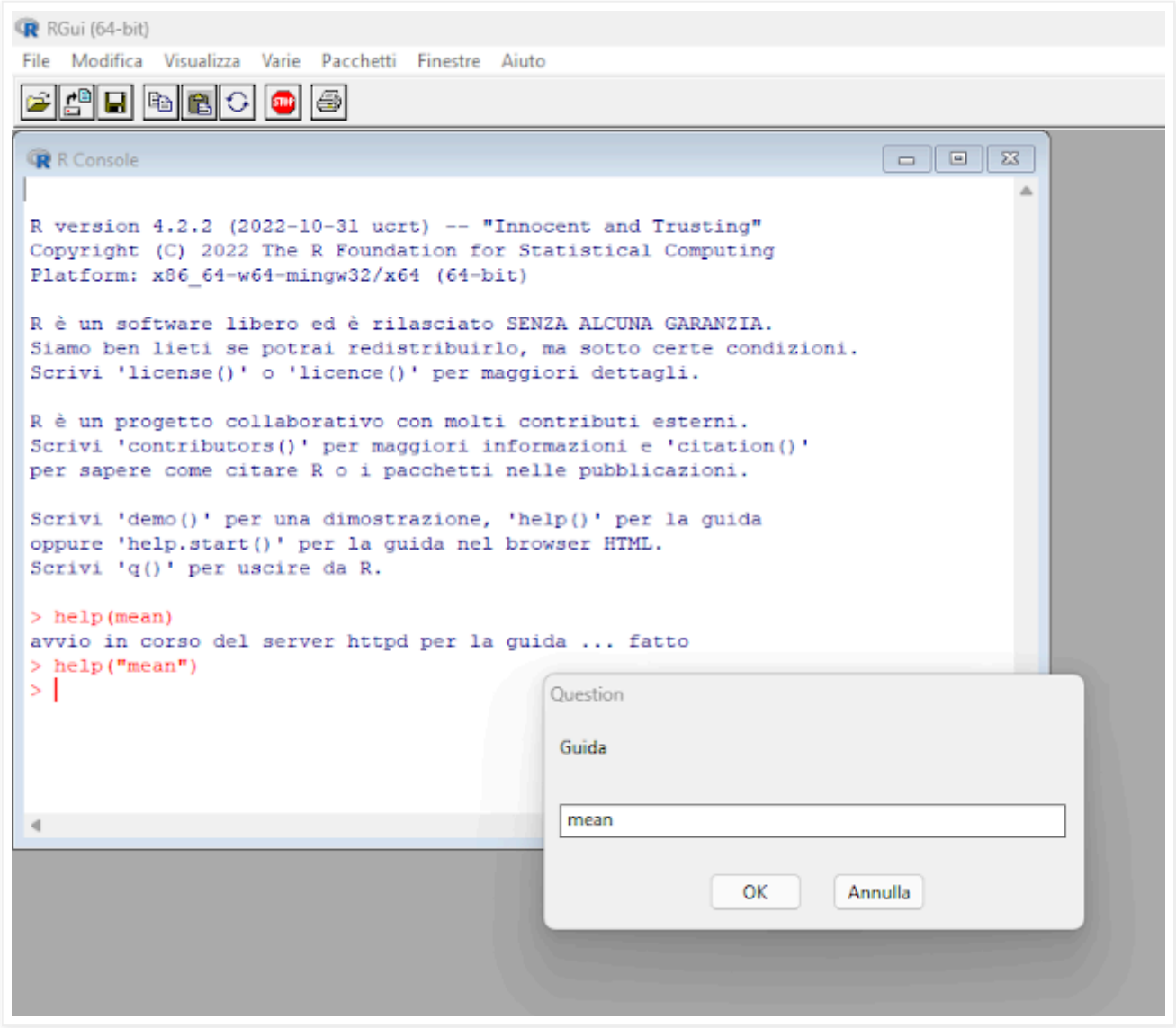
Molte delle informazioni che interessano, inclusa la documentazione delle funzioni incluse nell'installazione base di **R**, si possono trovare proprio nel manuale di riferimento (potete stamparlo ma attenzione, è molto voluminoso).

Se ai manuali in formato `.pdf` preferite i manuali interattivi potete trovare il loro equivalente nel menù `Aiuto` alla voce `Guida Html`.



La ricerca della *documentazione di una specifica funzione* può essere effettuata in due modi diversi ma che forniscono lo stesso risultato: nel browser si apre la pagina web con la documentazione ricercata.

Se come riportato nella figura precedente nel menù `Aiuto` selezionate l'opzione `Funzioni di R (testo)...` si apre una finestra nella quale è possibile inserire il nome della funzione senza le parentesi (vedi figura che segue). Quindi ad esempio:
→ digitate **mean** per la documentazione web della funzione **mean()**
→ digitate **plot** per la documentazione web della funzione **plot()**
→ eccetera...



In alternativa la ricerca della documentazione di una funzione può essere effettuata digitando **help(*nomedellafunzione*)** direttamente nella `Console di R`. Quindi, per continuare con l'esempio:
→ digitate **help(mean)** per la documentazione web della funzione **mean()**
→ digitate **help(plot)** per la documentazione web della funzione **plot()**
→ eccetera...

Personalmente preferisco digitare un **help()** e due parentesi in più e non passare attraverso i menù, ma come potete constatare qui è veramente questione di gusti.

Alla pagina dei Link trovate un elenco di collegamenti sia a documentazione del sito ufficiale di **R** sia a documentazione esterna, suddiviso in documentazione di base, documentazione utile per l'analisi statistica dei dati e documentazione utile per la rappresentazione grafica dei dati.

---

[1] *An Introduction to R.* URL consultato il 16/12/2018: https://goo.gl/PnjsBW

[2] *R: A Language and Environment for Statistical Computing, Reference Index*. URL consultato il 16/12/2018: https://goo.gl/kes91X

[3] *R Data Import/Export*. URL consultato il 16/12/2018: https://goo.gl/8mzH4L

Definizioni della probabilità

Diceva Bruno de Finetti: *"Ci fosse fuori di noi, per voi e per me, ci fosse una signora probabilità mia e una signora probabilità vostra, dico per se stesse, e uguali, immutabili. Non c'è..."* [1].

E proprio per il fatto di essere un concetto intuitivamente facile, ma difficile da formalizzare, sono oggi ben quattro le definizioni di probabilità [2].

La **definizione classica** (di Laplace) dice che:

"*La probabilità è il rapporto fra il numero di eventi favorevoli e il numero di eventi possibili, essendo questi ultimi tutti equiprobabili"* ovvero

$$P(A) = \frac{nA}{n}$$

Da notare che la definizione classica di probabilità contiene un vizio logico, in quanto la probabilità viene utilizzata per definire sé stessa.

La **definizione frequentista** (di Von Mises) dice che:

"*La probabilità di un evento è il rapporto fra il numero di esperimenti in cui esso si è verificato e il numero totale di esperimenti eseguiti nelle stesse condizioni, essendo tale numero opportunamente grande"*

$$P(A) = \lim_{n \to \infty} \frac{nA}{n}$$

La **definizione soggettivista** (di De Finetti) dice che:

"*...la probabilità che qualcuno attribuisce alla verità - o al verificarsi - di un certo evento (fatto singolo univocamente descritto e precisato) altro non è che la misura del grado di fiducia nel suo verificarsi"*

La **definizione assiomatica** (di Kolmogorov) dice che:

"*La probabilità è un numero compreso tra 0 e 1 che soddisfa gli assiomi di Kolmogorov"*

L'impostazione assiomatica della probabilità venne proposta da Andrey Nikolaevich Kolmogorov nel 1933 in *Grundbegriffe der Wahrscheinlichkeitsrechnung* (*Concetti fondamentali del calcolo delle probabilità*), ponendo termine al dibattito fra quanti consideravano la probabilità come limiti di frequenze relative (impostazione frequentista) e quanti cercavano un fondamento logico della stessa. La sua impostazione assiomatica si mostrava adeguata a prescindere dall'adesione a una o all'altra scuola di pensiero.

Un esempio dovuto a De Finetti consente di illustrare la differenza tra le prime tre definizioni. Immaginiamo una partita di calcio per la quale gli eventi possibili sono:
- la vittoria della squadra di casa;
- la vittoria della squadra ospite;
- il pareggio.

Secondo la **teoria classica** esiste 1 probabilità su 3 che avvenga la vittoria della squadra di casa.

Secondo la **teoria frequentista** ci si può dotare di un almanacco, controllare tutte le partite precedenti e calcolare la frequenza di un evento.

Secondo la **teoria soggettiva**, ci si può documentare sullo stato di forma dei calciatori, sul terreno di gioco e così via fino ad emettere un giudizio di probabilità (soggettiva).

---

[1] De Finetti B. *L'invenzione della verit*à. Raffaello Cortina Editore, Milano, 2006, ISBN 88-6030-060-6, p. 35.

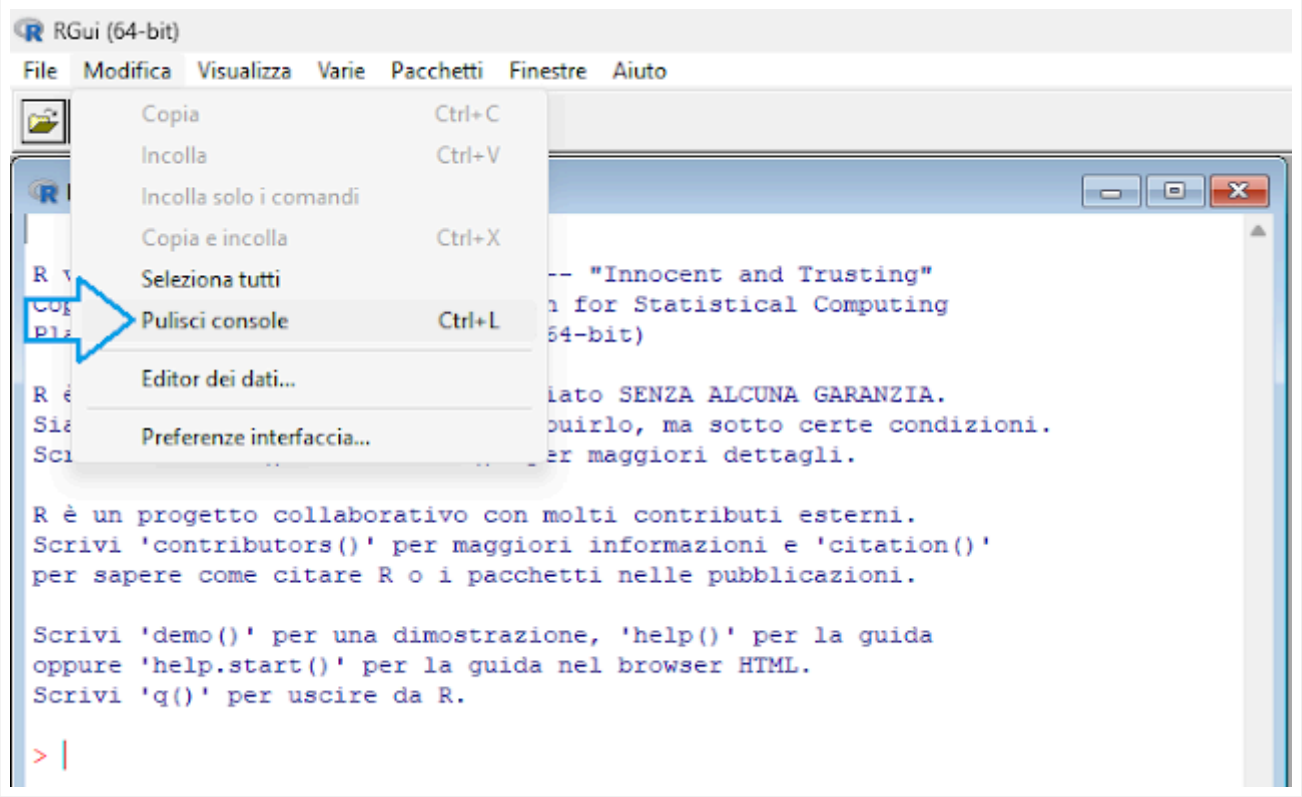
[2] *Probabilità*. Da Wikipedia, l'enciclopedia libera. URL consultato il 14/12/2018: https://goo.gl/DSUyai

domenica 16 dicembre 2018

Pulizia dell'area di lavoro e uscita dal programma

Quando lavorate con **R**, oggetti, codice e dati sono tutti conservati nell'**area di lavoro di R**.

Se volete ripulire la finestra della `Console di R` potete impiegare uno di questi due metodi:
→ nella `RGui` selezionate `Modifica` e nel menù a tendina che compare selezionate `Pulisci console`;
→ fate `ctrl-L` (tenendo premuto il tasto `ctrl` premete il tasto con la lettera `L`).



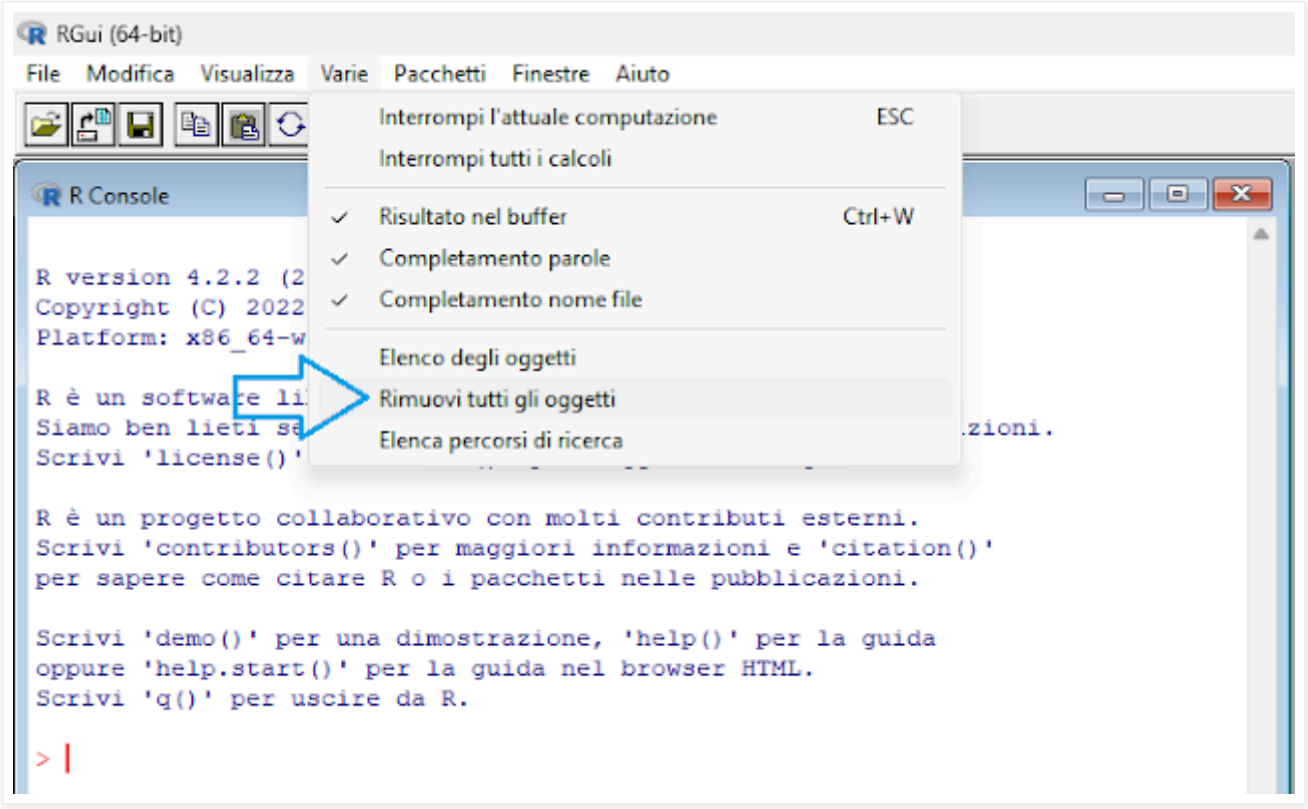
Questo tuttavia non elimina il contenuto della sessione corrente di **R** e quanto è presente nell'area di lavoro.

Potete avere un elenco degli oggetti presenti nell'area di lavoro digitando

ls()

nella `Console di R` [1].

Per eliminare dall'area di lavoro tutti gli oggetti dalla `RGui` selezionate `Varie` quindi nel menù a tendina che compare selezionate `Rimuovi tutti gli oggetti`;



In alternativa potete eliminare dall'area di lavoro tutti gli oggetti digitando nella `Console di R`

rm(list=ls(all=TRUE))

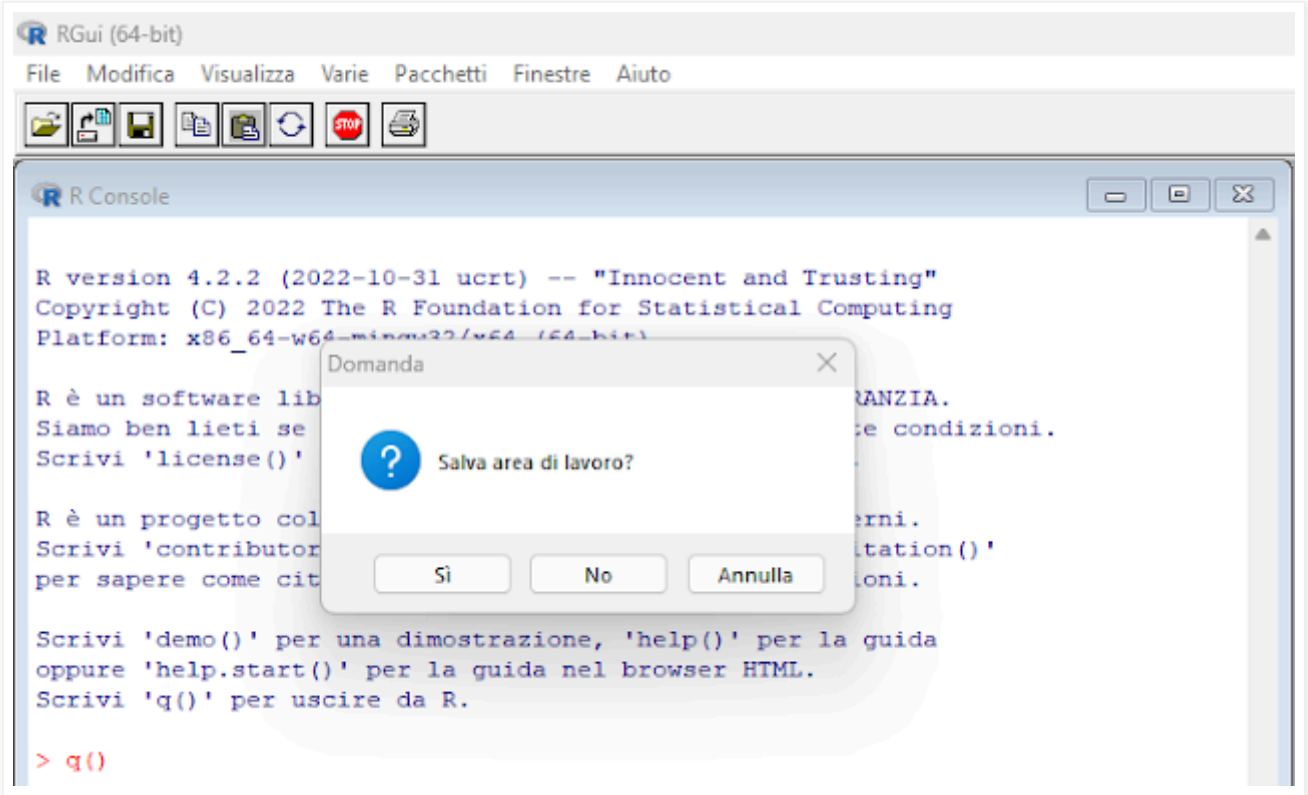
Infine potete eliminare uno specifico oggetto con la funzione **rm()** [2] digitando nella `Console di R`

rm(*nomedell'oggetto***)**

Si può anche effettuare una pulizia completa con questa riga di codice, copiatela, incollatela nella `Console di R` e premete `↵ Invio`:

q(save="no") # esce da R senza salvare l'area di lavoro che al rientro in R risulterà ripulita

Questo chiude il programma senza salvare l'area di lavoro - ovviamente il programma dovrà poi essere riavviato - ed è più sicuro di quanto lo siano uscire dalla `RGui` selezionando `File` e selezionando `Esci`, o chiudendo direttamente la finestra del proramma, perché in questo caso alla domanda `Salva area di lavoro?` si potrebbe inavvertitamente rispondere `Si`.



In **R** salvare l'area di lavoro significa salvare gli oggetti sui quali avete lavorato. Gli oggetti salvati vengono poi ripristinati nella sessione successiva. Il salvataggio dell'area di lavoro avviene in due file denominati

→ `.RData`

→ `.Rhistory`

che si trovano nella cartella `Documenti`. Se doveste salvare l'area di lavoro non volutamente, per ripulirla è sufficiente eliminare questi due file.

Se invece volete proprio salvare l'area di lavoro, ricordate che si tratta di una operazione delicata, che implica il ripristino alla sessiane successiva di pacchetti, dati, oggetti, funzioni e quant'altro della sessione precedente: richiede pertanto un completo controllo da parte vostra sull'ambiente di sviluppo.

Personalmente consiglio piuttosto, al termine di una sessione di lavoro con **R**, di effettuare una pulizia completa dell'area di lavoro per evitare potenziali interferenze con la sessione successiva.

---

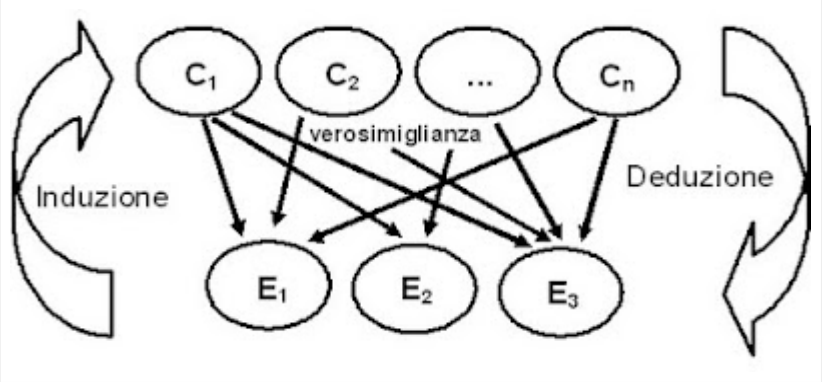
[1] Digitate **help(ls)** nella `Console di R` per la documentazione della funzione **ls()**.

[2] Digitate **help(rm)** nella `Console di R` per la documentazione della funzione **rm()**.

domenica 16 dicembre 2018

Teorema di Bayes e diagnosi medica

Si consideri uno schema come il seguente nel quale **C** rappresenta una delle tante cause osservabili, **E** rappresenta uno dei tanti effetti osservabili, e le frecce dirette dalla causa all'effetto rappresentano i rapporti di causa/effetto.



Il "*problema classico*" della probabilità può essere risolto applicando la deduzione logica. Conoscendo la causa, possiamo da questa dedurre gli effetti corrispondenti. Così sapendo che un dado ha sei facce, quindi conoscendo la causa, possiamo da questa dedurre la probabilità che ha una specifica faccia di comparire al prossimo lancio del dado (l'effetto). La soluzione del cosiddetto "*problema inverso*", ovvero l'induzione logica, è più difficile, in quanto ad un effetto possono corrispondere più cause. Il **teorema di Bayes** [1] consente, a partire dagli effetti osservati, di calcolare la verosimiglianza delle cause - espressa nel solo modo possibile, cioè in termini di probabilità.

Il teorema di Bayes trova applicazione nel campo della diagnostica medica. La quale prevede che sulla base della conoscenza medica - delle malattie (cause) e dei rispettivi sintomi e segni (effetti) - in uno specifico paziente si debba invertire il processo e risalire dai sintomi e dai segni presenti (effetti) alla malattia (causa) che li ha determinati. Compito sovente non facile dato che gli effetti possono essere attribuiti a differenti cause.

Per affrontare il tema sono necessarie alcune definizioni.

La **probabilità marginale** che si verifichi l'evento A è

$$P(A)$$

e si legge semplicemente "*la probabilità di A*". Dal punto di vista numerico la probabilità marginale di un evento è un numero positivo compreso tra 0 (evento che non accade mai) e 1 (evento certo) ovvero

$$0 \leq P(A) \leq 1$$

Corollario: la probabilità che non si verifichi l'evento A è

$$P(nonA) = 1 - P(A)$$

La **probabilità condizionata**

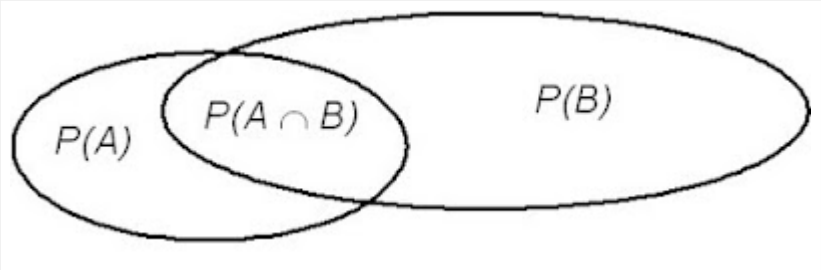
$$P(A|B)$$

è la probabilità di un evento A condizionata ad un evento B, ovvero è la probabilità che si verifichi A a condizione che si sia verificato B e si legge "*la probabilità di A, dato B*" ovvero "*la probabilità di A condizionata a B*"

La **probabilità congiunta**

$$P(A \cap B)$$

è la probabilità di due eventi congiunti, ovvero è la probabilità che si verifichino sia A sia B, e si legge "*la probabilità congiunta di A e B*" ovvero "la *probabilità di A e B*". La probabilità congiunta può essere meglio compresa facendo riferimento al diagramma seguente:



La relazione fra probabilità condizionata e probabilità congiunta è la seguente:

$$P(A \cap B) = P(A|B) \cdot P(B) \quad \text{(i)}$$

ma poiché la probabilità congiunta deve necessariamente essere la stessa sia partendo dalla probabilità di A sia partendo dalla probabilità di B, deve essere anche

$$P(A \cap B) = P(B|A) \cdot P(A) \quad \text{(ii)}$$

Combinando la (i) con la (ii) si ottiene il **teorema delle probabilità composte**

$$P(A \cap B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

e dall'identità

$$P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

si ricava il **teorema di Bayes**:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Se poniamo *A = la causa* e *B = l'effetto*, il teorema di Bayes suona così

$$P(\text{della causa}|\text{dato l'effetto}) = \frac{P(\text{dell'effetto}|\text{data la causa}) \times P(\text{della causa})}{P(\text{dell'effetto})}$$

Se consideriamo il problema specifico della diagnosi medica, nel caso di una malattia M (la *causa*) che può essere presente (soggetti malati, *M+*) o assente (soggetti sani, *M-*), e di un segno T (l'*effetto*) determinato da questa malattia, come è tipicamente il risultato di un test per la sua diagnosi, che può essere positivo (*T+*) o negativo (*T-*), l'espressione precedente può essere riscritta come

$$P(M+|T+) = \frac{P(T+|M+) \cdot P(M+)}{P(T+)} \quad \text{(iii)}$$

Partendo dalla probabilità condizionata

	Malattia +	**Malattia -**
Test +	$P(T+\|M+)$	$P(T+\|M-)$
Test -	$P(T-\|M+)$	$P(T-\|M-)$

e moltiplicando le probabilità della prima colonna (Malattia +) per la probabilità di presenza della malattia nella popolazione $P(M+)$, cioè per la prevalenza della malattia, e moltiplicando le probabilità della seconda colonna (Malattia -) per la probabilità di assenza della malattia nella popolazione $P(M-)$, essendo $P(M-) = 1 - P(M+)$ otteniamo

	Malattia +	**Malattia -**
Test +	$P(T+\|M+) \cdot P(M+)$	$P(T+\|M-) \cdot P(M-)$
Test -	$P(T-\|M+) \cdot P(M+)$	$P(T-\|M-) \cdot P(M-)$

Sostituendo la probabilità di un test positivo $P(T+)$ riportata al denominatore della (iii) con la somma $P(T+|M+) \cdot P(M+) + P(T+|M-) \cdot P(M-)$ delle probabilità riportate nella riga Test + di quest'ultima tabella, possiamo esprimere il teorema di Bayes (iii) nella forma seguente, che consente di calcolare il **valore predittivo di un test positivo** $P(M+|T+)$ ovvero la probabilità di essere malato per un soggetto che presenta un test positivo

$$P(M+|T+) = \frac{P(T+|M+) \cdot P(M+)}{P(T+|M+) \cdot P(M+) + P(T+|M-) \cdot P(M-)} \quad \text{(iv)}$$

Le grandezze in gioco nel calcolo del valore predittivo di un test positivo $P(M+|T+)$ sono [2]:
→ la **sensibilità** $P(T+|M+)$ del test diagnostico, che è la probabilità che il test sia positivo in un soggetto malato;
→ la **specificità** $P(T-|M-)$ del test diagnostico che è la probabilità che il test sia negativo in un soggetto sano;
→ la **prevalenza** della malattia $P(M+)$ cioè la probabilità di incontrare un soggetto malato nella popolazione generale.

Il teorema di Bayes consente di calcolare anche il **valore predittivo di un test negativo** $P(M-|T-)$, ovvero la probabilità di essere sano per un soggetto che presenta un test negativo, nella forma seguente

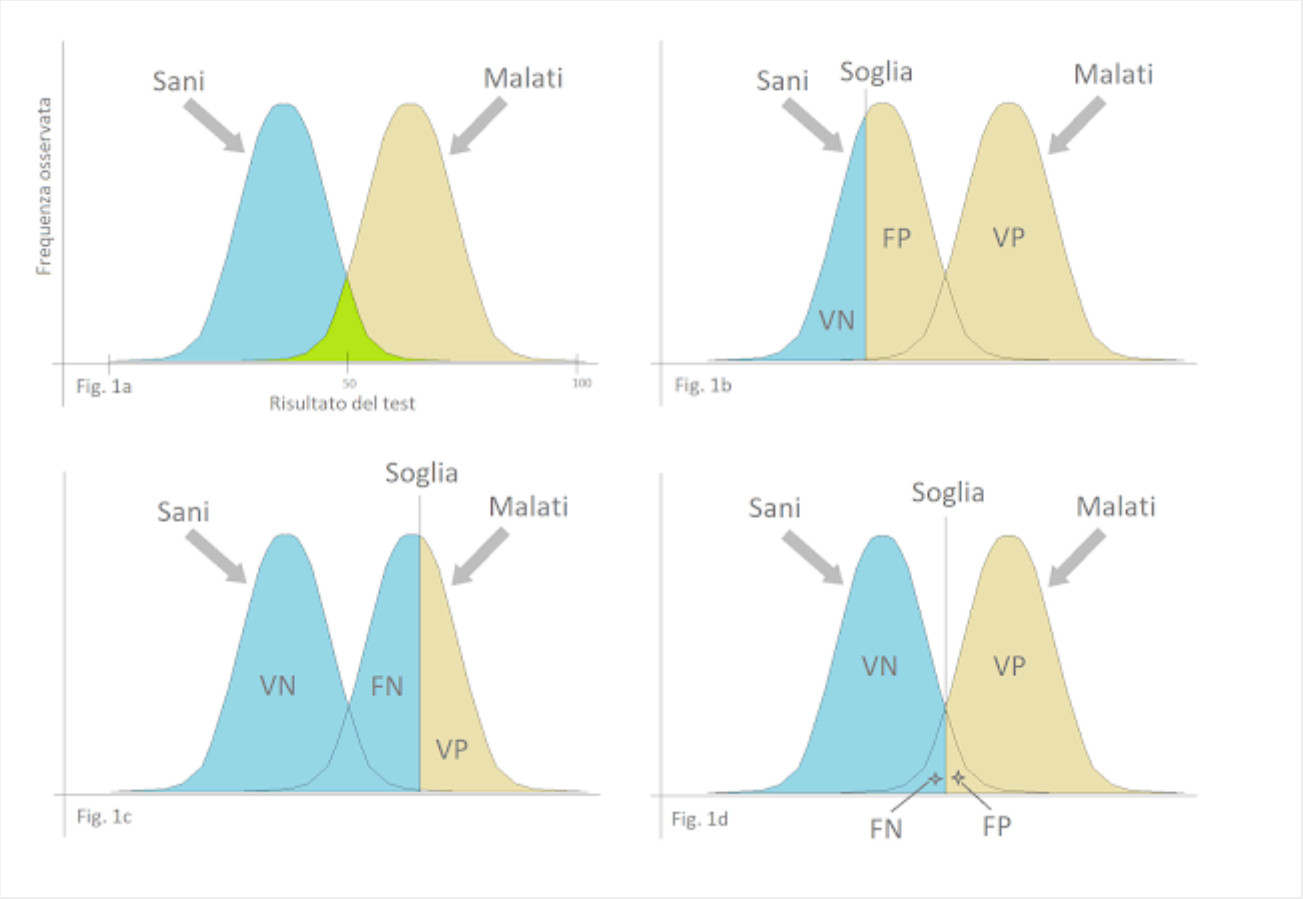
$$P(M-|T-) = \frac{P(T-|M-) \cdot P(M-)}{P(T-|M-) \cdot P(M-) + P(T-|M+) \cdot P(M+)} \quad \text{(v)}$$

In genere, a causa del fatto che le statistiche epidemiologiche sono effettuate contando i casi - cosicché per conoscere ad esempio la prevalenza dell'influenza si conta il numero dei malati di influenza esistenti nella popolazione [3] - spesso si preferisce, in alternativa alle probabilità della (iv) e della (v), impiegare il numero di casi osservati, riportati in questo modo

	Malattia +	**Malattia -**
Test +	VP	FP
Test -	FN	VN

e definiti, per uno specifico test diagnostico e una specifica malattia, come [4]:
→ **veri positivi** (VP), il numero di soggetti malati con il test positivo;
→ **falsi positivi** (FP), il numero di soggetti sani con il test positivo;
→ **falsi negativi** (FN), il numero di soggetti malati con il test negativo;
→ **veri negativi** (VN), il numero di soggetti sani con il test negativo.

Queste quattro grandezze, per un dato test diagnostico, sono quindi dei semplici conteggi dei risultati ottenuti applicando il test in condizioni controllate (nell'ambito di una ricerca clinica). In questa immagine



è riportata in altro a sinistra (Fig. 1a) la distribuzione dei risultati di un ipotetico test in un campione di soggetti non affetti (Sani) e in un campione di soggetti affetti (Malati) da una specifica malattia. Le distribuzioni si sovrappongono parzialmente (area in colore verde): ed è quanto si osserva di fatto, nella stragrande maggioranza dei casi, nella realtà.

Se, ai fini della classificazione dei pazienti che hanno eseguito il test, fissiamo come valore soglia tra sani a malati quello riportato nella Fig. 1b, ci garantiamo di individuare correttamente tutti i soggetti malati (VP), e non avremo falsi negativi (FN), ma a scapito del fatto che classificheremo come malati anche molti soggetti sani (FP) e ridurremo il numero di soggetti sani correttamente classificati come tali (VN). In altre parole, in termini di probabilità, privilegeremo la sensibilità del test a scapito della sua specificità: non avremo alcun falso negativo (FN) ma avremo molti falsi positivi (FP).

Se fissiamo come valore soglia tra sani e malati quello riportato nella Fig. 1c, ci garantiamo di individuare correttamente tutti i soggetti sani (VN), e non avremo falsi positivi (FP) ma a scapito del fatto che classificheremo come sani anche molti soggetti malati (FN) e ridurremo il numero di soggetti malati correttamente classificati come tali (VP). In altre parole, in termini di probabilità, privilegeremo la specificità del test a scapito della sua sensibilità: non avremo alcun falso positivo (FP) ma avremo molti falsi negativi (FN).

Se infine fissiamo come valore soglia tra sani e malati quello riportato nella Fig. 1d, ci garantiamo di individuare correttamente la maggior parte dei soggetti sani (VN) e di individuare correttamente la maggior parte dei soggetti malati (VP), minimizzando contemporaneamente sia la possibilità di sbagliare classificando come sani dei soggetti malati (FN) sia la possibilità di sbagliare classificando come malati dei soggetti sani (FP). In altre parole, in termini di probabilità, realizzeremo un rapporto equilibrato tra la specificità del test e la sua sensibilità.

Nella maggior parte dei casi il valore soglia è scelto sulla base della strategia rappresentata nella Fig. 1d, che minimizza sia i falsi positivi sia i falsi negativi: è proprio questa la strategia seguita dalle curve ROC (vedere il post Curve ROC e valore soglia). Tuttavia in casi particolari è possibile che sia più appropriato impiegare la strategia rappresentata nella Fig. 1b (massimizzare la sensibilità) o più appropriato impiegare la strategia rappresentata nella Fig. 1c (massimizzare la specificità). Per una introduzione generale all'impiego del teorema di Bayes nella diagnosi medica vedere Federspil, Galen, Gigerenzer, Gill, Goodman, Motterlini e Nordenström nella pagina della bibliografia.

Dopo avere definito il valore soglia, dal quale dipendono la sensibilità e la specificità del test:
→ il valore predittivo del test positivo, cioè la probabilità di essere malato per un soggetto che presenta un test positivo [5]
→ il valore predittivo del test negativo, cioè la probabilità di essere sano per un soggetto che presenta un test negativo [5]
possono essere calcolati - a partire dalle probabilità mediante la (iv) e la (v) o, in alternativa, a partire dal numero di casi osservati VP, FP, FN, VN - come illustrato nel post Sensibilità, specificità, valore predittivo e nel post Efficienza e accuratezza diagnostica.

---

[1] Bayes T. *An essay towards solving a problem in the doctrine of chances*. *By the late Rev. Mr.* Bayes*, F. R. S. communicated by Mr.* Price*, in a letter to* John Canton*, A. M. F. R. S*. Philo. Trans. Roy. Soc. 1763;53;370-418, doi: https://doi.org/10.1098/rstl.1763.0053

[2] Altman DG, Bland JM. *Statistics Notes: Diagnostic tests 1: sensitivity and specificity*. BMJ 1994;308;1552. URL consultato il 29/10/2018: https://goo.gl/x7Txjz

[3] In termini di conteggi la prevalenza è il numero di casi di malattia esistenti in un determinato momento in una data popolazione. Non deve essere confusa con l'incidenza, che è il numero di nuovi casi di malattia che si verificano in un determinato arco di tempo in una data popolazione.

[4] Gerhardt W, Keller H. *Evaluation of test data from clinical studies*. Scand J Clin Lab Invest 1986;46(supplement 181). URL consultato il 18/06/2019: http://bit.ly/2IVp4ox

[5] Altman DG, Bland JM. *Statistics Notes: Diagnostic tests 2: predictive values*. BMJ 1994;309;102. URL consultato il 29/10/2018: https://goo.gl/xffgB9

lunedì 17 dicembre 2018

La regressione lineare: assunti e modelli

La **regressione lineare** (**x variabile indipendente**) viene trattata estesamente in tutti i testi di statica, ai quali si rimanda per gli aspetti tecnici [1]. Tuttavia esiste un aspetto relativo al calcolo della regressione lineare che viene spesso sottovalutato, ed è il fatto che fornisce risultati che dipendono da quale delle due variabili in esame sia assunta come **variabile indipendente**, che per definizione, per le esigenze imposte dal modello matematico impiegato, deve essere posta sull'**asse delle ascisse (x)**. L'altra variabile, la **variabile dipendente**, viene posta sull'asse delle ordinate (y).

Si considerino i seguenti dati (simulati):

Variabile 1	Variabile 2
9	9
10	10
11	11
9	10
10	9
10	11
11	10

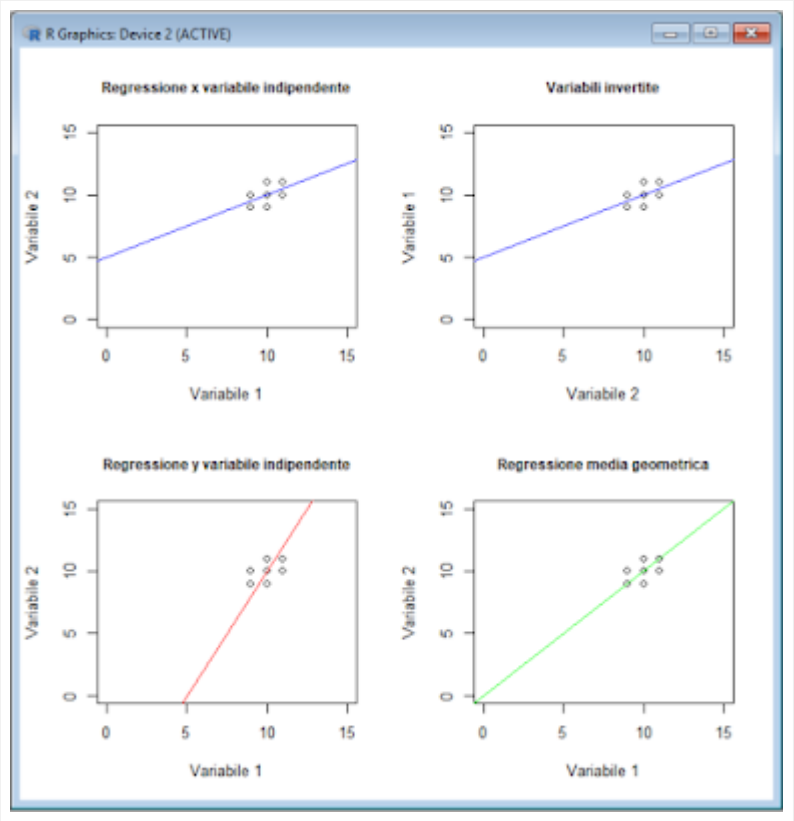
Non avendo alcuna ragione per assumere l'una o l'altra delle due variabili come la variabile indipendente:
→ prima calcoliamo l'equazione della retta di regressione x variabile indipendente ponendo sull'asse delle ascisse (x) la Variabile 1;
→ poi ricalcoliamo l'equazione della retta di regressione x variabile indipendente ponendo sull'asse delle ascisse (x) la Variabile 2;
→ ricaviamo da quest'ultima l'equazione della retta di **regressione y variabile indipendente**;
→ infine calcoliamo l'equazione della retta di regressione con un metodo che non impiega nessuna delle due come variabile indipendente, ma calcola la media geometrica dei coefficienti angolari della regressione x variabile indipendente e della regressione y variabile indipendente (**regressione media geometrica**) [2, 3].

Copiate la prima parte dello script [4], quindi incollatela nella `Console di R` e premete `↵ Invio`:

```
# MODELLI DI REGRESSIONE PARAMETRICA retta y = a + b·x con dati simulati (1/4)
#
windows() # apre una nuova finestra
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
var_1 <- c(9,10,11,9,10,10,11) # variabile 1
var_2 <- c(9,10,11,10,9,11,10) # variabile 2
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 1", ylab="Variabile 2",
main="Regressione x variabile indipendente", cex.main = 0.9) # grafico dei dati
regpar <- lm(y ~ x) # regressione lineare standard
a_yx <- regpar$coefficients[1] # intercetta a
```

```
b_yx <- regpar$coefficients[2] # coefficiente angolare b
abline(a_yx, b_yx, col="blue") # retta di regressione x variabile indipendente
c(a_yx, b_yx) # mostra a e b
#
```

Il risultato di questa prima parte dello script, nel quale l'equazione della retta di **regressione x variabile indipendente** è calcolata ponendo sull'asse delle ascisse (x) la Variabile 1, è il primo grafico in alto a sinistra:



A questo punto scambiamo tra di loro le due variabili e ricalcoliamo l'equazione della retta di regressione con questa seconda parte dello script. Copiatela e incollatela nella `Console di R` e premete ↵ `Invio`:

```
x <- var_2 # variabile 2 in ascisse (2/4)
y <- var_1 # variabile 1 in ordinate
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 2", ylab="Variabile 1",
main="Variabili invertite", cex.main = 0.9) # grafico dei dati
regpar_1 <- lm(y ~ x) # regressione lineare standard
a_yx_1 <- regpar_1$coefficients[1] # intercetta a
b_yx_1 <- regpar_1$coefficients[2] # coefficiente angolare b
abline(a_yx_1, b_yx_1, col="blue") # retta di regressione con variabili scambiate
c(a_yx_1, b_yx_1) # mostra a e b
#
```

Ora la Variabile 2 è riportata in ascisse (e diventa quindi la variabile indipendente) e la Variabile 1 è riportata in ordinate (grafico in alto a destra). L'equazione della retta di regressione è identica alla precedente (questo è atteso visto che Variabile 1 e Variabile 2 comprendono identici valori). Per confrontare questo grafico e questa retta di regressione con quelli del primo grafico dobbiamo capovolgere il grafico orizzontalmente, quindi ruotarlo a destra di 90°, riportando così la Variabile 2 in ordinate e riportando la Variabile 1 in ascisse, cosa che facciamo con questa terza parte dello script. Copiatela e incollatela nella `Console di R` e premete ↵ `Invio`:

```
x <- var_1 # variabile 1 in ascisse (3/4)
```

```
y <- var_2 # variabile 2 in ordinate
a_xy <- - a_yx_1 / b_yx_1 # intercetta a
b_xy <- 1 / b_yx_1 # coefficiente angolare b
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 1", ylab="Variabile 2",
main="Regressione y variabile indipendente", cex.main = 0.9) # grafico dei dati
abline(a_xy, b_xy, col="red") # retta di regressione y variabile indipendente
c(a_xy, b_xy) # mostra a e b
#
```

Abbiamo così ottenuta la **regressione y variabile indipendente**, illustrata nel grafico in basso a sinistra.

Infine calcoliamo l'equazione della retta di regressione espressa come **regressione media geometrica**, copiate la quarta e ultima parte dello script e incollatela nella `Console di R` e premete `↵ Invio`:

```
x <- var_1 # variabile 1 in ascisse (4/4)
y <- var_2 # variabile 2 in ordinate
b_mg <- sqrt(b_yx*b_xy) # coefficiente angolare b
a_mg <- mean(y) - (b_mg * (mean(x))) # intercetta a
plot(x, y, xlim = c(0,15), ylim = c(0,15), xlab="Variabile 1", ylab="Variabile 2",
main="Regressione media geometrica", cex.main = 0.9) # grafico dei dati
abline(a_mg, b_mg, col="green") # retta di regressione media geometrica
c(a_mg, b_mg) # mostra a e b
#
```

Questa regressione (grafico in basso a destra) assume come coefficiente angolare la media geometrica dei coefficienti angolari della regressione x variabile indipendente e della regressione y variabile indipendente. E con la salomonica conclusione che `y = x` fornisce una soluzione intermedia, che appare più convincente rispetto all'una e all'altra regressione lineare, per il fatto che la retta calcolata risulta essere allineata con l'asse maggiore dell'elissoide che include i dati.

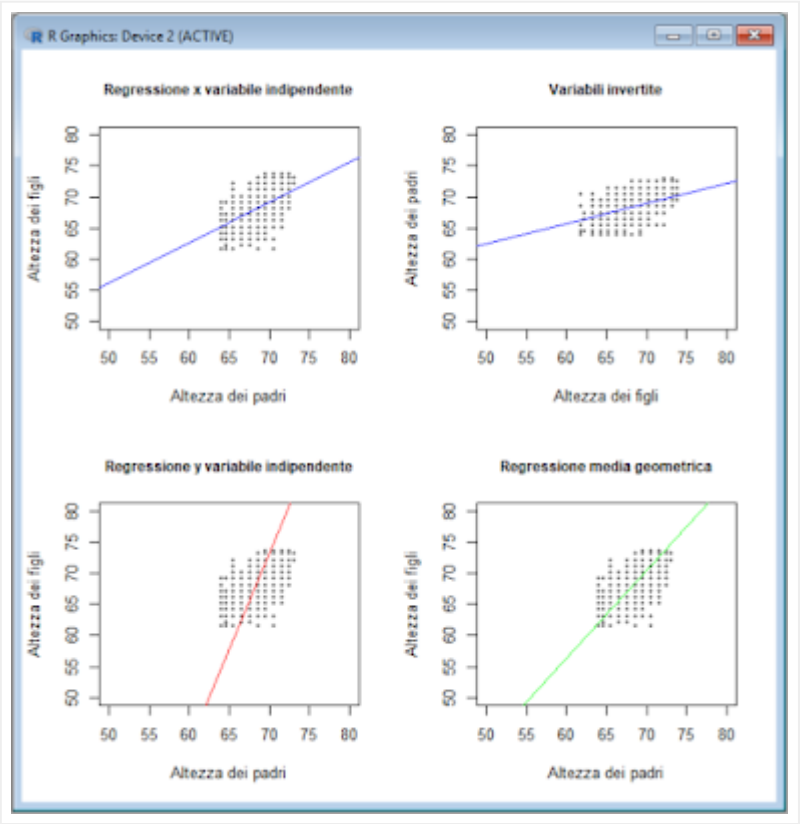
Nel caso del set di dati galton [5] incluso nel pacchetto **psychTools** ci troviamo proprio in una situazione di questo genere. Nell'analizzare mediante la regressione lineare le altezze dei genitori e le altezze dei figli non vi sono ragioni per assumere le prime come variabile indipendente. Ma non vi sono neppure ragioni per assumere le altezze dei figli come variabile indipendente. Possiamo allora effettuare un'analisi di questi dati provando a calcolare anche per essi la regressione lineare nei tre modi possibili qui riportati:
→ **regressione x variabile indipendente**;
→ **regressione y variabile indipendente**;
→ **regressione media geometrica**.

Se non l'avete già fatto, prima di eseguire lo script dovete scaricare e installare il pacchetto aggiuntivo **psychTools**. Copiate per intero lo script riportato qui sotto quindi incollatelo nella `Console di R` e premete `↵ Invio`:

```
# MODELLI DI REGRESSIONE PARAMETRICA retta y = a + b·x set di dati galton
#
library(psychTools) # carica il pacchetto psych incluso il set di dati galton
#
windows() # apre una nuova finestra
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
var_1 <- galton$parent # variabile 1
var_2 <- galton$child # variabile 2
#
```

```
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli", main="Regressione x variabile indipendente", cex = 0.5, cex.main = 0.9) # grafico dei dati
regpar <- lm(y ~ x) # regressione lineare standard
a_yx <- regpar$coefficients[1] # intercetta a
b_yx <- regpar$coefficients[2] # coefficiente angolare b
abline(a_yx, b_yx, col="blue") # retta di regressione x variabile indipendente
c(a_yx, b_yx) # visualizza a e b
#
x <- var_2 # variabile 2 in ascisse
y <- var_1 # variabile 1 in ordinate
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei figli", ylab="Altezza dei padri", main="Variabili invertite", cex = 0.5, cex.main = 0.9) # grafico dei dati
regpar_1 <- lm(y ~ x) # regressione lineare standard
a_yx_1 <- regpar_1$coefficients[1] # intercetta a
b_yx_1 <- regpar_1$coefficients[2] # coefficiente angolare b
abline(a_yx_1, b_yx_1, col="blue") # retta di regressione con variabili scambiate
c(a_yx_1, b_yx_1) # visualizza a e b
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
a_xy <- - a_yx_1 / b_yx_1 # intercetta a
b_xy <- 1 / b_yx_1 # coefficiente angolare b
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli", main="Regressione y variabile indipendente", cex = 0.5, cex.main = 0.9) # grafico dei dati
abline(a_xy, b_xy, col="red") # retta di regressione y variabile indipendente
c(a_xy, b_xy) # visualizza a e b
#
x <- var_1 # variabile 1 in ascisse
y <- var_2 # variabile 2 in ordinate
b_mg <- sqrt(b_yx * b_xy) # coefficiente angolare b
a_mg <- mean(y) - (b_mg * (mean(x))) # intercetta a
plot(x, y, xlim = c(50,80), ylim = c(50,80), xlab="Altezza dei padri", ylab="Altezza dei figli", main="Regressione media geometrica", cex = 0.5, cex.main = 0.9) # grafico dei dati
abline(a_mg, b_mg, col="green") # retta di regressione media geometrica
c(a_mg, b_mg) # visualizza a e b
#
```

Questo è il grafico risultante.



Come si vede chiaramente assumendo come variabile indipendente i padri o i figli i risultati cambiano. Il problema, generato a scopo didattico nei dati simulati analizzati con il primo script e riportati nella prima figura e che si presenta spontaneamente nei dati di Francis Galton riportati in quest'ultima figura [6], nasce dal fatto che l'equazione della retta di regressione è determinata dall'interazione di due fattori:
→ gli **assunti** alla base del **modello** di regressione lineare impiegato;
→ l'**informazione** che il modello di regressione lineare può ricavare dai dati forniti.

Più i dati sono allineati su una retta:
→ più consistente è l'informazione in essi contenuta;
→ più **r** si avvicina a 1;
→ meno rilevante è l'influenza degli assunti alla base del modello di regressione sulle conclusioni.
Tanto che nel caso limite di dati perfettamente allineati su una retta, quando **r** = 1 e l'informazione contenuta nei dati è univoca, tutti i modelli di regressione forniscono lo stesso identico risultato, indipendentemente dagli assunti alla base del modello di regressione.

Meno i dati sono allineati su una retta:
→ meno consistente è l'informazione in essi contenuta;
→ meno **r** è vicino a 1 (più **r** si avvicina a 0);
→ più rilevante è l'influenza degli assunti alla base del modello di regressione sulle conclusioni.

Nel caso limite di dati distribuiti in una palla rotonda, quando **r** = 0 e l'informazione contenuta nei dati è nulla (nessuno sa come far passare una retta da punti così distribuiti), i modelli di regressione manifestano il massimo di divergenza nei risultati, che a questo punto riflettono esclusivamente gli assunti alla base del modello di regressione.

I dati di Galton presentano un coefficiente di correlazione **r** uguale a `0.4587624` e un coefficiente di determinazione $\mathbf{R}^2$ = 0.21 che indica che solamente il 21% della variabilità osservata viene spiegato dalla retta. Ci troviamo di fronte a un caso tipico di una situazione nella quale le conclusioni dipendono più dagli assunti alla base del modello impiegato per interpretare i dati che dalla oggettività fornita dai dati stessi. Non potendo assumere né le altezze dei padri né le altezze dei figli come variabile indipendente, è opportuno accettare le conclusioni tratte dal calcolo della (retta di

regressione espressa come) regressione media geometrica, la cui caratteristica più convincente è di essere allineata con l'asse maggiore dell'ellissoide che include i dati (**Fig. 2** grafico in basso a destra).

---

[1] Vedere i riferimenti nel post: Regressione lineare semplice parametrica.

[2] Ling Leng et al. *Ordinary least square regression, orthogonal regression, geometric mean regression and their applications in aerosol science*. J. Phys. 2007;Conf. Ser. 78 012084. DOI: 10.1088/1742-6596/78/1/012084

[3] Shaoji Xu. *A Property of Geometric Mean Regression*. The American Statistician 2014;68(4);277-281. DOI: 10.1080/00031305.2014.962763

[4] Nei link che seguono trovate i dettagli:
→ sui simboli dei punti di R che è possibile impiegare nella funzione **plot()**;
→ gli stili delle linee di R che è possibile impiegare nella funzione **abline()**;
→ i colori di R che potete impiegare;
→ la possibilità di aggiungere la legenda a un grafico.

[5] Vedere il post Il set di dati galton.

[6] Il codice per riportare le tre rette su di un unico grafico è riportato nel post: Aggiungere la legenda a un grafico.

giovedì 27 dicembre 2018

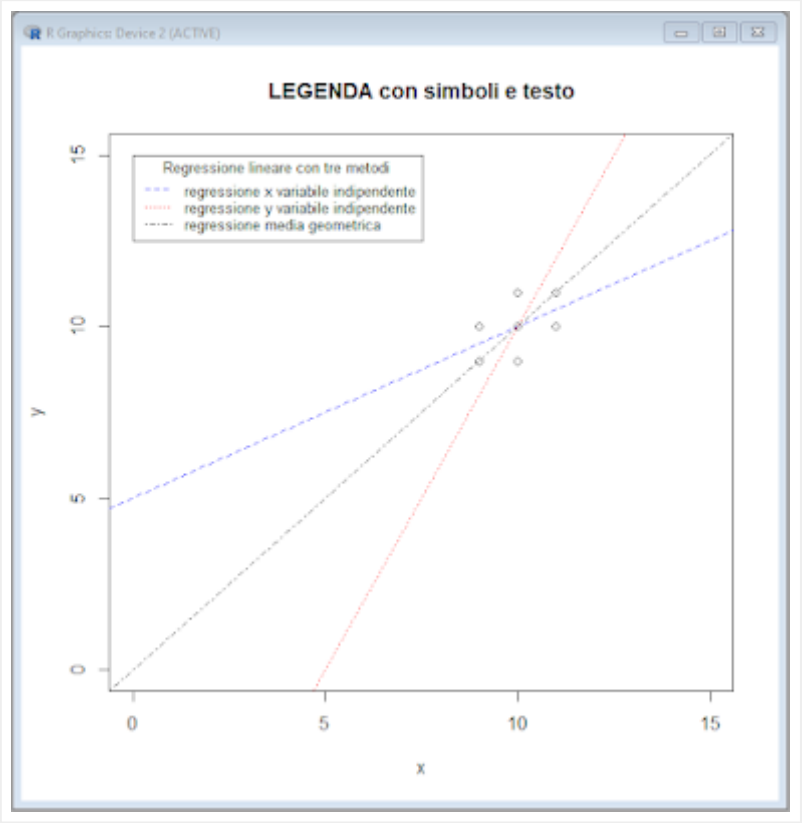
Aggiungere la legenda a un grafico

I modi per aggiungere una legenda a un grafico sono dispersi qua e la nei post di questo blog. Vale quindi la pena di farne una sintesi riportando gli esempi più significativi, basati sull'impiego della funzione **legend()** e realizzati senza la necessità di installare pacchetti aggiuntivi ma impiegando solamente le *funzioni statistiche e grafiche di base* di **R**.

Nel primo esempio realizziamo la legenda che combina simboli e testo per identificare le tre rette di regressione riportate nel grafico, calcolate con altrettanti modelli basati su differenti assunti [1].

Copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# LEGENDA con simboli e testo
#
x <- c(9,10,11,9,10,10,11) # variabile in ascisse
y <- c(9,10,11,10,9,11,10) # variabile in ordinate
windows() # apre e inizializza una nuova finestra grafica
#
plot(x, y, xlim=c(0,15), ylim=c(0,15), main="LEGENDA con simboli e testo") # grafico dei dati
abline(5, 0.5, lty=2, col="blue") # retta x variabile indipendente
abline(-10, 2, lty=3, col="red") # retta y variabile indipendente
abline(0, 1, lty=4, col="black") # retta media geometrica
#
legend(0, 15, legend=c("regressione x variabile indipendente", "regressione y variabile indipendente", "regressione media geometrica"), lty=c(2,3,4), col=c("blue", "red", "black"), cex=0.8, title="Regressione lineare con tre metodi") # aggiunge al grafico la legenda
#
```



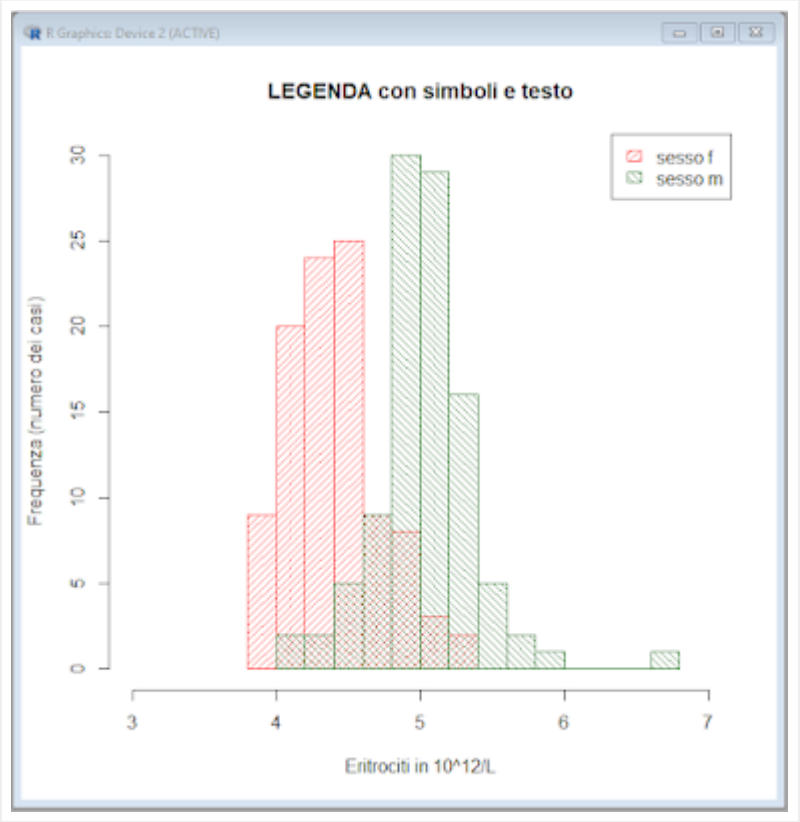
La legenda necessaria per identificare le tre rette riportate nel grafico viene realizzata mediante la funzione **legend()** impiegando come argomenti:
→ **0** che specifica il valore della `x` al quale posizionare l'angolo superiore sinistro della legenda;
→ **15** che specifica il valore della `y` al quale posizionare l'angolo superiore sinistro della legenda;
→ **legend=** che specifica con la funzione **c(, ,)** le tre righe di testo, uno per ciascuna retta, che compaiono nella legenda;
→ **lty=** che specifica con la funzione **c(, ,)** i tre stili delle linee corrispondenti a quelli delle rette tracciate nel grafico - da notare che specificando **lty** (o **lwd** cioè lo spessore della linea) la funzione **legend()** traccia automaticamente accanto alle righe di testo della legenda il simbolo rappresentato da un segmento di retta;
→ **col=** che specifica con la funzione **c(, ,)** i tre colori corrispondenti e quelli delle rette tracciate nel grafico [2];
→ **cex=0.8** che gestisce la dimensione della legenda (la rimpicciolisce lievemente rispetto al valore **1** di default);
→ **title=** che mostra sulla prima riga della legenda un titolo (facoltativo).

La posizione della legenda può essere definita anche sostituendo i due primi argomenti con "*bottomright*", "*bottom*", "*bottomleft*", "*left*", "*topleft*", "*top*", "*topright*", "*right*" e "*center*", con una personalizzazione meno spinta di quella possibile con l'impiego delle coordinate cartesiane, ma che può essere adeguata, come vediamo nel prossimo script.

Il secondo esempio illustra come generare una legenda che combina ancora simboli e testo, ma in modo leggermente diverso.

Copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# LEGENDA con simboli e testo
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
windows() # apre e inizializza una nuova finestra grafica
#
hist(f, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), border="red", col="red", density=20, angle=45, main="LEGENDA con simboli e testo", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di f
hist(m, add=TRUE, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(m)), border="green4", col="green4", density=20, angle=-45) # istogramma di m
#
legend("topright", legend=c("sesso f","sesso m"), fill=c("red","green4"), border=c("red","green4"), density=c(20,20), angle=c(45,-45)) # riporta la legenda
#
```



La legenda che identifica i due istogrammi è realizzata con la funzione **legend()** impiegando come argomenti:
→ **"topright"** che posiziona la legenda in alto a destra;
→ **legend=** che specifica le due righe di testo che compaiono nella legenda;
→ **fill=** che determina la comparsa accanto alle righe di testo di due quadrati che avranno all'interno i colori specificati;
→ **border=** che definisce i colori del bordo dei quadrati di cui al punto precedente;
→ **density=** che specifica la densità del colore indicato dall'argomento **fill=** e che con un valore uguale a **20** (linee per pollice) genera in entrambi i quadrati un tratteggio con questa densità, identica a quella dei corrispondenti istogrammi;
→ **angle=...** che indicare l'angolo di pendenza del tratteggio che con un valore positivo dell'angolo (**45**) riprende lo stile del primo istogramma, inclinato da sinistra a destra dal basso in alto, e con un valore negativo dell'angolo (**-45**) riprende lo stile del secondo istogramma, inclinato da sinistra a destra dall'alto in basso.

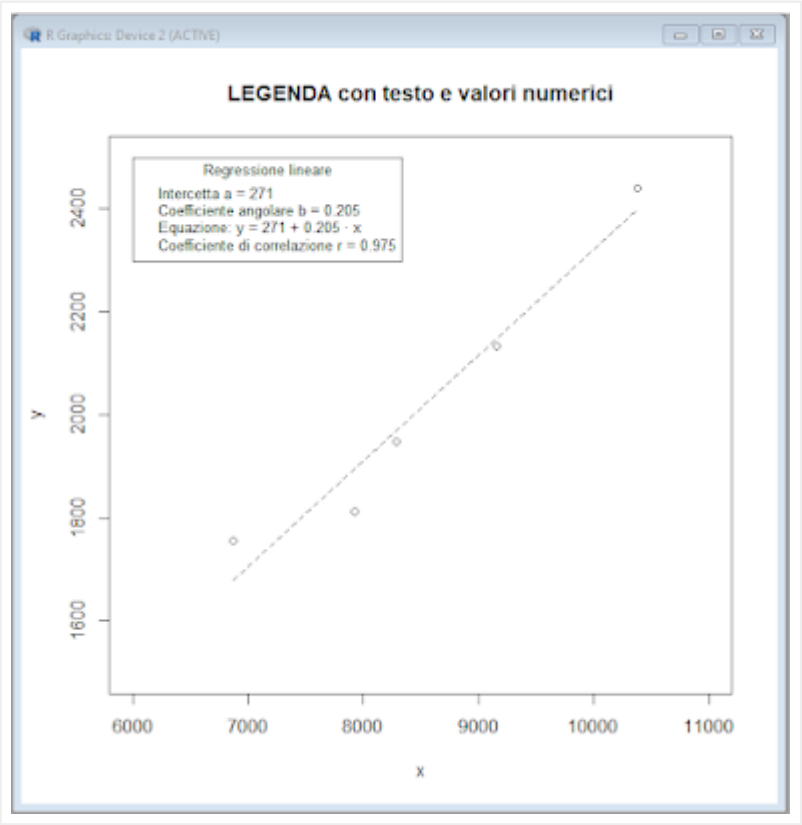
Anche in questo caso la funzione **c()** consente di specificare uno per uno i valori degli argomenti per riportarli ai corrispondenti elementi del grafico.

Il terzo esempio illustra come impiegare la funzione **paste()** per riportare all'interno di una legenda righe che combinano testo e numeri.

Copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# LEGENDA con testo e valori numerici
#
x <- c(10376,9159,8290,7924,6872) # valori in ascisse
y <- c(2441,2135,1948,1812,1756) # valori in ordinate
windows() # apre e inizializza una nuova finestra grafica
#
reglin <- lm(y ~ x) # calcola la regressione lineare
a <- reglin$coefficients[1] # intercetta a
b <- reglin$coefficients[2] # coefficiente angolare b
r <- cor(x, y, method="pearson") # coefficiente di correlazione r
```

#
plot(x, y, xlim=c(6000,11000), ylim=c(1500,2500), main="LEGENDA con testo e valori numerici") # grafico dei dati
lines(c(min(x), max(x)), c(a+b*min(x), a+b*max(x)), col="black", lty=2, lwd=1) # retta di regressione
#
legend(6000, 2500, legend=c(paste("Intercetta a =", round(a, digits=0)), paste("Coefficiente angolare b =", round(b, digits=3)), paste("Equazione: y =", round(a, digits=0), ifelse(sign(b) == 1, "+", "-"), round(b, digits=3), "· x"), paste("Coefficiente di correlazione r =", round(r, digits=3))), cex=0.8, title="Regressione lineare") # aggiunge al grafico la legenda
#



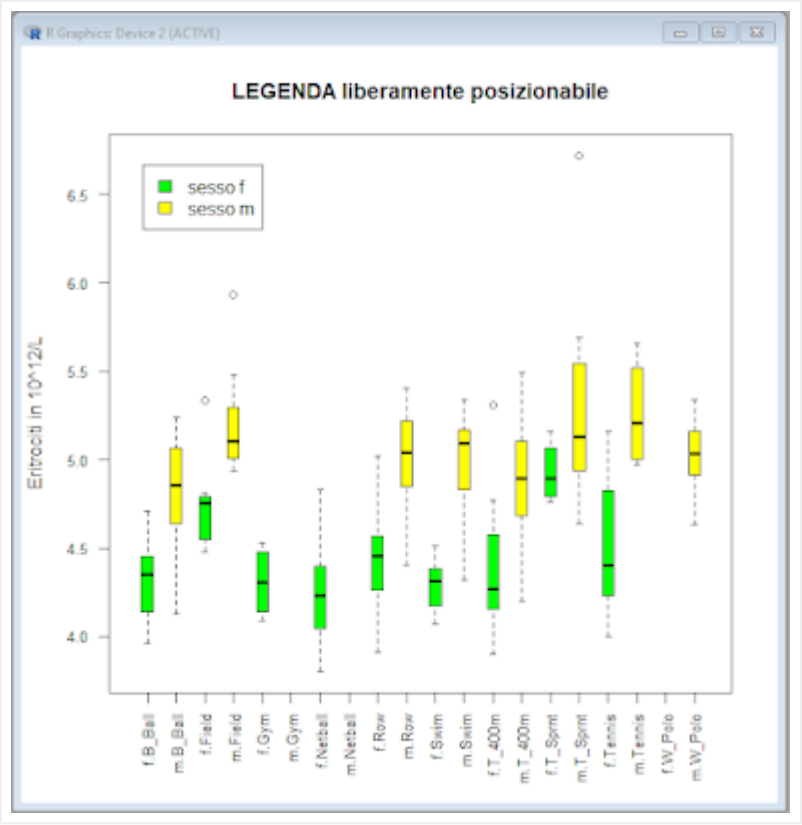
In questo caso la legenda, che riporta nel grafico l'equazione della retta di regressione e il coefficiente di correlazione lineare, viene realizzata impiegando come argomenti della funzione **legend()**:
→ **6000** che specifica il valore della `x` al quale posizionare l'angolo superiore sinistro della legenda;
→ **2500** che specifica il valore della `y` al quale posizionare l'angolo superiore sinistro della legenda;
→ **legend=** che specifica mediante la funzione **c()** le quattro righe che compaiono nella legenda, ciascuna delle quali è generata mediante la funzione **paste()** che combina uno o più testi inclusi nelle virgolette **"...."** con uno o più valori numerici arrotondati mediante la funzione **round()**, mentre la funzione **ifelse()** riporta il segno corretto nell'equazione della retta di regressione;
→ **cex=0.8** che gestisce la dimensione della legenda (la rimpicciolisce lievemente rispetto al valore **1** di default);
→ **title=** che mostra sulla prima riga della legenda un titolo (facoltativo).

Il quarto esempio impiega i dati della tabella **ais** inclusa nel pacchetto **DAAG**. Potete scaricare il pacchetto dal **CRAN** oppure acquisire la tabella seguendo le indicazioni alternative riportate in [3].

L'esempio illustra come generare una legenda liberamente posizionabile: *dopo avere tracciato i grafici separati per sport praticato e per sesso, lo script rimane in attesa. A questo punto bisogna posizionare il mouse dove si vuole che compaia la legenda e fare* `click` *con il tasto sinistro del mouse per farla comparire e terminare lo script*.

Copiate e incollate questo script nella `Console di R` e premete ↵ `Invio`:

```
# LEGENDA liberamente posizionabile
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
attach(ais) # impiega direttamente i nomi delle variabili senza specificare la tabella
windows() # apre e inizializza una nuova finestra grafica
#
boxplot(rcc~sex+sport, horizontal=FALSE, boxwex = 0.4, cex.axis = 0.8, las = 2, data=ais,
main="LEGENDA liberamente posizionabile", xlab="", ylab="Eritrociti in 10^12/L",
notch=FALSE, col=c("green", "yellow")) # eritrociti per ciascuno sport praticato
#
legend(locator(1), legend=c("sesso f","sesso m"), fill=c("green", "yellow")) # posiziona la
legenda
#
detach(ais) # termina l'impiego diretto dei nomi delle variabili
#
```



La legenda viene realizzata mediante la funzione **legend()** impiegando come argomenti:
→ la funzione **locator()** che resta in attesa, legge la posizione del cursore grafico quando viene fatto `click` con il tasto sinistro del mouse, e posiziona la legenda;
→ **legend=** che specifica le due righe di testo che compaiono nella legenda;
→ **fill=** che determina la comparsa accanto alle righe di testo di due quadrati dei colori specificati.

Il codice riportato non consente di spostare la legenda. Se non si è soddisfatti della sua posizione, è necessario rieseguire l'intero script e fare nuovamente `click` con il tasto sinistro del mouse nel punto in cui si vuole posizionare la legenda.

Per le semplici legende che si possono inserire nei grafici a torta e nei grafici a barre si rimanda ai relativi post [4, 5].

Gli esempi forniti consentono di realizzare facilmente il codice necessario per inserire le legende nei propri grafici. Si consiglia di consultare il Manuale di Riferimento di R ovvero di

digitare **help**(*nomedellafunzione*) nella `Console di R` per la documentazione degli argomenti che consentono di adattare e personalizzare le funzioni impiegate.

Infine se al vostro grafico avete aggiunto una legenda probabilmente volete non solo visualizzarlo, ma anche salvarlo sotto forma di file per archiviarlo o inserirlo in una pubblicazione, in un post o in un sito web. Trovate nel post Salvare i grafici di R in un file come farlo nei formati più diffusi.

---

[1] L'argomento è trattato nel post La regressione lineare: assunti e modelli.

[2] Vedere il post Visualizzare i colori disponibili in R.

[3] Vedere nel post Il set di dati ais come acquisire i dati della tabella **ais** anche senza istallare il pacchetto **DAAG**.

[4] Vedere il post Grafici a torta.

[5] Vedere il post Grafici a barre [1] e il post Grafici a barre [2].

venerdì 28 dicembre 2018

Salvare i grafici di R in un file

Una rappresentazione grafica dei dati è importante non solo visualizzarla, ma anche salvarla sotto forma di file per archiviarla o inserirla in una pubblicazione, in un post o in un sito web.

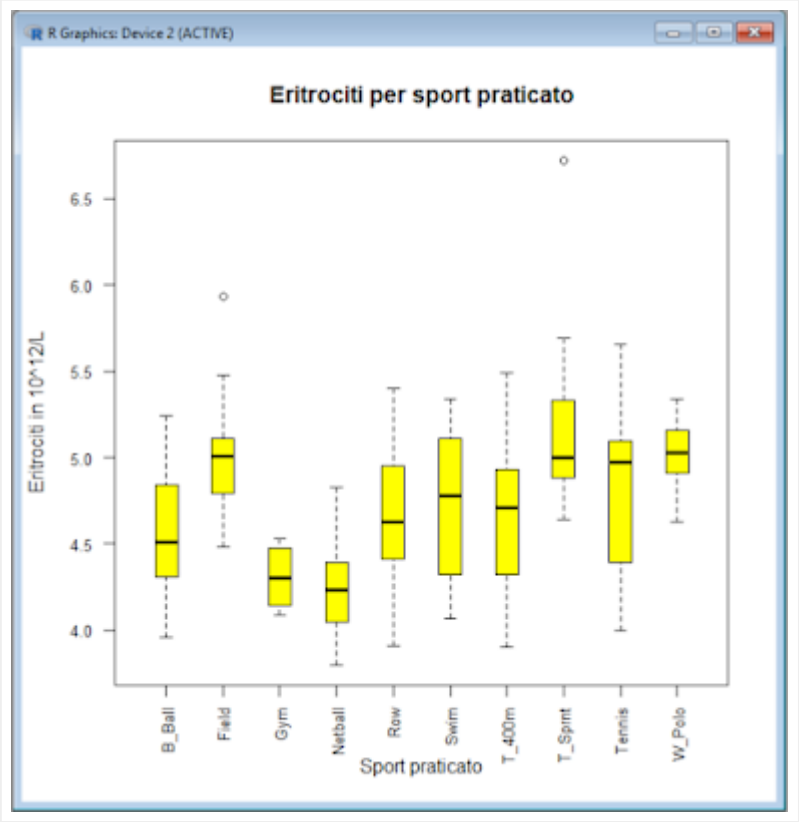
Tecnicamente vi sono due modi per salvare un'immagine sotto forma di file:
→ salvarla in **formato raster** (detto anche **bitmap**);
→ salvarla in **formato vettoriale**.

Un'immagine in formato raster/bitmap è costituita da una matrice di punti (*pixel*) opportunamente colorati. Quando viene ingrandita la qualità dell'immagine peggiora. In compenso è possibile agire sui pixel modificando luminosità, contrasto, saturazione dei colori dell'immagine. Il formato raster/bitmap è tipico della fotografia.

Un'immagine in formato vettoriale è invece costruita con una tecnica che consente, mediante trasformazioni matematiche (*vettori*, onde il nome, e *matrici*), di ingrandirla senza perdita della qualità.

In questo esempio nel quale i caratteri tipografici sono stati ingranditi di circa sei volte è evidente la perdita di qualità del formato raster/bitmap (a sinistra) rispetto al formato vettoriale (a destra):

Nel primo script viene salvato sotto forma di file nei tre formati grafici raster/bitmap più comuni e cioè
→ `.bmp` (Windows bitmap)
→ `.jpeg` (Joint Photographic Experts Group)
→ `.png` (Portable Network Graphics)
questo grafico a scatola con i baffi (boxplot) che illustra la distribuzione della concentrazione nel sangue degli eritrociti (globuli rossi) **rcc** per **sport** praticato ricavata dal set di dati **ais** [1]:



Prima di eseguire lo script è necessario creare (se non l'avete ancora fatto) sul vostro PC o notebook la cartella `C:\Rdati\` nella quale verranno salvati i tre file, denominati rispettivamente `boxplot.bmp`, `boxplot.jpg` e `boxplot.png`. I dati sono contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [1].

Copiate per intero lo script riportato qui sotto quindi incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# SALVA I GRAFICI SU DISCO SOTTO FORMA DI FILE come immagini in formato raster (bitmap)
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
#
# salva l'immagine in formato .bmp (Windows bitmap)
#
bmp("C:/Rdati/boxplot.bmp", units = "px", width = 1000, height = 1000, pointsize = 24,
bg = "white") # predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, boxwex = 0.4, cex.axis = 0.8, las = 2, data=ais,
main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in 10^12/L",
notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
# salva l'immagine in formato .jpeg (Joint Photographic Experts Group)
#
jpeg("C:/Rdati/boxplot.jpg", units = "px", width = 1000, height = 1000, pointsize = 24, bg
= "white") # predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, boxwex = 0.4, cex.axis = 0.8, las = 2, data=ais,
main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in 10^12/L",
notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
# salva l'immagine in formato .png (Portable Network Graphics)
```

```
#
png("C:/Rdati/boxplot.png", units = "px", width = 1000, height = 1000, pointsize = 24, bg = "white") # predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, boxwex = 0.4, cex.axis = 0.8, las = 2, data=ais,
main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in 10^12/L",
notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
```

Gli argomenti **width=1000** e **height=1000** specificano larghezza e altezza dell'immagine in pixel. Questa unità di misura può essere cambiata con l'argomento **units** che qui è posto uguale a **"px"** ma che può essere espresso in alternativa come **"in"** (pollici), **"cm"** (centimetri) o **"mm"** (millimetri).

L'argomento **pointsize = 24** specifica la dimensione dei caratteri, mentre l'argomento **bg = "white"** specifica il colore dello sfondo.

Da notare che nella rappresentazione del grafico a scatola con i baffi nella funzione **boxplot()** compaiono come argomenti:
→ l'argomento **boxwex = 0.4** che gestisce la larghezza dei boxplot;
→ l'argomento **cex.axis = 0.8** che gestisce la dimensione dei caratteri;
→ l'argomento **las = 2** che ruota verticalmente le etichette.

In questo secondo script lo stesso grafico viene salvato nei formati vettoriali
→ `.pdf` (Portable Document Format)
→ `.ps` (Postscript)
nella cartella `C:\Rdati\` nei file denominati rispettivamente `boxplot.pdf` e `boxplot.ps`.

Copiate per intero lo script riportato qui sotto quindi incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# SALVA I GRAFICI SU DISCO SOTTO FORMA DI FILE come immagini in formato vettoriale
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
#
# salva l'immagine in formato .pdf (Portable Document Format)
#
pdf("C:/Rdati/boxplot.pdf", width = 7, height = 7) # predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, boxwex = 0.4, cex.axis = 0.8, las = 2, data=ais,
main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in 10^12/L",
notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
# salva l'immagine in formato .ps (postscript)
#
postscript("C:/Rdati/boxplot.ps", width = 7, height = 7) # predispone nome e formato del file
boxplot(rcc~sport, horizontal=FALSE, boxwex = 0.4, cex.axis = 0.8, las = 2, data=ais,
main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in 10^12/L",
notch=FALSE, col="yellow") # traccia boxplot
dev.off() # salva il file
#
```

Da notare che i formati `.pdf` e `.ps` in quanto formati vettoriali prevedono che le dimensioni dell'immagine (**width = 7, height = 7**) siano espresse in pollici (non avrebbe senso esprimerle in pixel). Inoltre quando estraete dal file `.ps` il file `boxplot.pdf` il file preesistente con lo stesso nome generato dalla funzione **pdf()** verrà sovrascritto.

In entrambi gli script la funzione **dev(off)** salva il file e chiude il dispositivo (virtuale) che ha generato la stampa, e a questo punto il dispositivo al quale sono inviati i grafici torna ad essere la finestra che viene aperta con la funzione **windows()** sul monitor/schermo del PC o notebook.

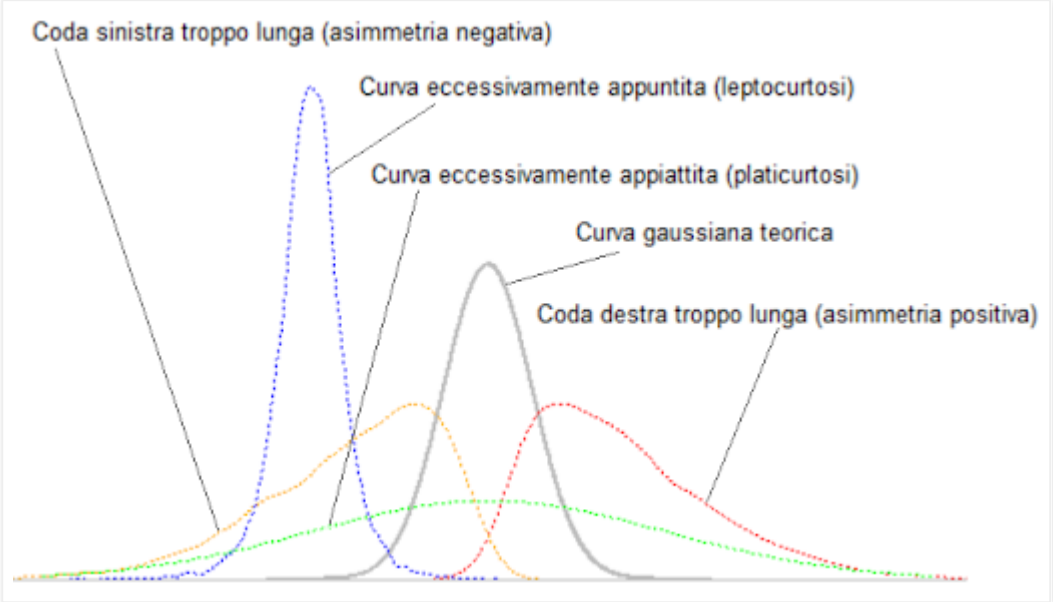
---

[1] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

venerdì 4 gennaio 2019

Valutare asimmetria e curtosi

I due test di normalità (gaussianità) più classici e tradizionali sono:
→ il **test di asimmetria** (test di D'Agostino) che valuta il grado di scostamento della simmetria della distribuzione campionaria (quella effettivamente osservata) da quella della distribuzione gaussiana teorica, che è per definizione perfettamente simmetrica;
→ il **test di curtosi** (test di Anscombe-Glynn) che ci dice se la distribuzione campionaria è troppo appiattita (bassa e larga) o troppo appuntita (alta e stretta) rispetto alla distribuzione gaussiana teorica, in ciascun punto della quale vale un preciso rapporto tra larghezza e altezza.



Prima di eseguire lo script è necessario scaricare e installare il pacchetto aggiuntivo **moments**. Trovate il manuale di riferimento, che include i riferimenti bibliografici, sul repository della documentazione di **R** [1]. Per il test di D'Agostino vedere anche [2].

I dati sono contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [3].

Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
#  TEST DI ASIMMETRIA E TEST DI CURTOSI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(moments) # carica il pacchetto per l'analisi statistica
#
skewness(ais$ferr) # coefficiente di asimmetria
agostino.test(ais$ferr) # test di D'Agostino per il coefficiente di asimmetria
kurtosis(ais$ferr) # coefficiente di curtosi
anscombe.test(ais$ferr) # test di Anscombe-Glynn per il coefficiente di curtosi
#
```

Dopo avere caricato il pacchetto **DAAG** che contiene i dati nella tabella **ais** e il pacchetto **moments** che contiene le funzioni che ci interessano, nella terza riga di codice con la funzione **skewness()** viene calcolato il coefficiente di asimmetria della

variabile **ais$ferr** (concentrazione della `ferritina` nel siero espressa in `µg/L`) la cui significatività viene valutata nella riga successiva con la funzione **agostino.test()**.

Il tutto viene ripetuto per il coefficiente di curtosi, calcolato con la funzione **kurtosis()** e la cui significatività viene poi valutata nella riga successiva mediante la funzione **anscombe.test()**.

Il coefficiente di asimmetria è significativo, con un valore osservato di asimmetria `skew=1.2806` che dista `z=6.1376` deviazioni standard dal valore `0` previsto per una distribuzione gaussiana, e una probabilità di osservare per caso tale distanza molto bassa (`p-value=8.377e-10`):

```
        D'Agostino skewness test

data:  ais$ferr
skew = 1.2806, z = 6.1376, p-value = 8.377e-10
alternative hypothesis: data have a skewness
```

Il coefficiente di curtosi è anch'esso significativo, con un valore osservato di curtosi `kurt=4.4202` che dista `z=2.9449` deviazioni standard dal valore `3` previsto per una distribuzione gaussiana, e una probabilità di osservare per caso tale distanza bassa (`p-value=0.003231`):

```
        Anscombe-Glynn kurtosis test

data:  ais$ferr
kurt = 4.4202, z = 2.9449, p-value = 0.003231
alternative hypothesis: kurtosis is not equal to 3
```

I criteri per valutare la significatività statistica sono ovviamente i soliti:
→ si considera il risultato non significativo quando la probabilità di osservare per caso il valore (effettivamente ottenuto nel caso specifico) della statistica è "elevato";
→ si considera il risultato significativo quando la probabilità di osservare per caso il valore (effettivamente ottenuto nel caso specifico) della statistica è "sufficientemente basso";
→ come valore soglia tra "elevato" e "sufficientemente basso" viene tradizionalmente impiegato il valore di probabilità `p=0.05` (ma nulla vieta, al fine di ridurre ulteriormente la possibilità di trarre dal test statistico conclusioni errate, di impiegare un valore soglia inferiore, come per esempio `p=0.01` e quindi considerare significativo un risultato statistico solamente quando la probabilità di osservarlo per caso è dell'1% o meno).

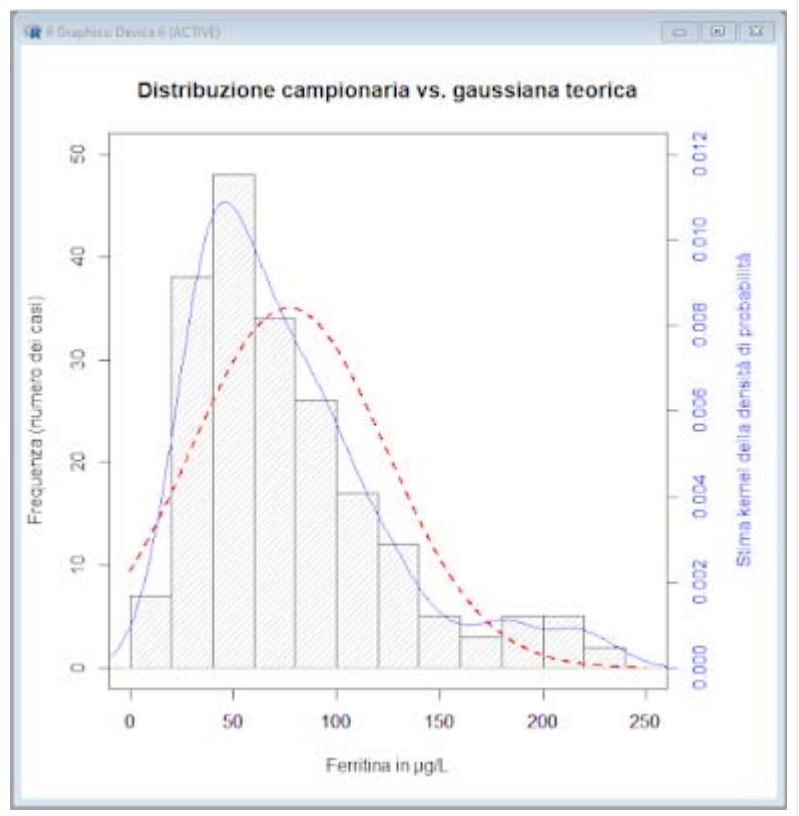
Integriamo ora l'analisi statistica con una rappresentazione grafica dei dati che riporta, sovrapposti, l'istogramma e il kernel density plot della distribuzione campionaria e per confronto la distribuzione gaussiana teorica, quella che gli stessi dati avrebbero se fossero distribuiti in modo gaussiano.

Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# DISTRIBUZIONE CAMPIONARIA vs. GAUSSIANA TEORICA
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mar=c(5.1,4.1,4.1,5.1)) # aumenta il margine destro da 2.1 a 5.1
#
hist(ais$ferr, xlim=c(0,250), ylim=c(0,50), freq=TRUE, breaks="FD", density=20,
angle=45, border="black", main="Distribuzione campionaria vs. gaussiana teorica",
xlab="Ferritina in µg/L", ylab="Frequenza (numero dei casi)") # traccia l'istogramma
#
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
```

plot(density(ais$ferr), yaxt="n", xlim=c(0,250), ylim=c(0,0.012), col="blue", lwd=1) # sovrappone all'istogramma il kernel density plot
axis(4, col.ticks="blue", col.axis="blue") # riporta sulla destra l'asse delle y del kernel density plot
mtext("Stima kernel della densità di probabilità", side=4, line=3, cex=1, col="blue") # aggiunge la legenda all'asse delle y sulla destra
#
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
x <- seq(0, 250, length.out=1000) # calcola i valori in ascisse della gaussiana teorica
y <- dnorm(x, mean=mean(ais$ferr), sd=sd(ais$ferr)) # calcola i valori in ordinate della gaussiana teorica
plot(x, y, xlim=c(0, 250), ylim=c(0, 0.012), yaxt="n", col="red", type="l", lty=2, lwd=2) # sovrappone la distribuzione gaussiana teorica in colore rosso
#
par(mar=c(5.1,4.1,4.1,2.1)) # ripristina i margini di default
#

Questo è il grafico risultante:



La conclusione dei test statistici trova una conferma empirica nella rappresentazione grafica dei dati: la concentrazione della ferritina nel siero, rispetto alla corrispondente distribuzione gaussiana teorica (in tratteggio di colore rosso) – cioè rispetto a quella che i dati dovrebbero avere se fossero distribuiti in modo gaussiano – è eccessivamente appuntita (leptocurtica) e asimmetrica (asimmetria positiva) pertanto la distribuzione dei dati non è gaussiana.

Nel codice riportato si possono notare alcune cose:
→ il numero delle classi **breaks="FD"** è calcolato con la regola di Friedman-Diaconis [4];
→ le barre dell'istogramma sono profilate in nero (**border="black"**) riempite di colore (il colore grigio di default) con un tratteggio che copre il 20% della superficie (**density=20**) ed è inclinato a 45 gradi (**angle=45**);
→ la scala delle ordinate di sinistra riporta, per l'istogramma, il numero dei casi osservati in ciascuna classe;

→ la scala delle ordinate di destra riporta, per il kernel density plot, la stima kernel della densità di probabilità [5];
→ per la distribuzione gaussiana teorica vale la stessa scala di densità di probabilità riportata sulla destra;
→ le ordinate della distribuzione gaussiana corrispondente ai dati sono calcolate con la funzione **dnorm()** impiegando la media (**mean(ais$ferr)**) e la deviazione standard (**sd(ais$ferr)**) campionarie della ferritina;
→ la distribuzione gaussiana teorica, quella che i dati avrebbero avere, è riportata con una linea tratteggiata (**lty=2**) di spessore aumentato (**lwd=2**) in colore rosso (**col="red"**).

Nel post Analizzare graficamente la distribuzione di una variabile potete trovare, riunite in un'unica vista, quattro grafici per una rappresentazione più completa delle differenze tra la vostra distribuzione campionaria e la distribuzione gaussiana teorica.

I test di asimmetria e di curtosi sono molto importanti, ma più importante ancora è che facciano parte di un approccio complessivo e organico al calcolo delle **statistiche elementari** di una variabile che, si ricorda, prevede di:
→ effettuare una analisi esplorativa dei dati
→ effettuare uno o più test di normalità (gaussianità).
→ impiegare le statistiche elementari parametriche se i dati sono distribuiti in modo gaussiano;
→ impiegare le statistiche elementari non parametriche se i dati non sono distribuiti in modo gaussiano.

---

[1] *Available CRAN Packages By Name*. URL consultato il 04/01/2019: https://goo.gl/hLC9BB

[2] D'Agostino RB. *Transformation to Normality of the Null Distribution of g1*. Biometrika 1970;57(3);679-681. URL consultato il 05/01/2019: https://goo.gl/xMVUEx

[3] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto DAAG.

[4] Vedere Wikipedia, the free encyclopedia. Freedman–Diaconis rule. URL consultato il 22/02/2023.

[5] Per i dettagli della rappresentazione grafica vedere il post Istogrammi e il post Kernel density plot.

lunedì 7 gennaio 2019

Analisi esplorativa dei dati

Con l'espressione "analisi esplorativa dei dati" non si fa riferimento a una tecnica statistica specifica, bensì all'insieme delle valutazioni preliminari che è sempre necessario effettuare nell'ambito di un percorso logico che prevede, per il calcolo delle statistiche elementari di una singola variabile (analisi univariata), i seguenti passi:
→ **analisi esplorativa dei dati**;
→ esecuzione dei test di normalità (gaussianità), per valutare se i dati seguono una distribuzione gaussiana;
→ calcolo delle statistiche elementari parametriche (media, deviazione standard, varianza, quantili parametrici) se i dati seguono una distribuzione gaussiana [1];
→ calcolo delle statistiche elementari non parametriche (mediana, deviazione assoluta mediana o MAD, quartili e altri quantili non parametrici) se i dati non seguono una distribuzione gaussiana.

In quanto si tratta di una prima fase, l'analisi esplorativa dei dati non prevede test di significatività, che sono riservati alla fasi successive, ma rappresenta piuttosto il momento di valutazione critica preliminare e globale dei dati raccolti, che deve includere quantomeno:
→ l'identificazione di eventuali *dati mancanti*;
→ un primo confronto orientativo tra i risultati di *statistiche parametriche* e di *statistiche non parametriche*;
→ l'individuazione di possibili *dati anomali* (outliers) cioè di dati che si discostano in modo inatteso dagli altri;
→ l'identificazione della possibile origine degli outliers (errori di digitazione? casi inclusi erroneamente? problemi strumentali? altro?) e la valutazione degli interventi correttivi (laddove possibili).

Vediamo ora alcune funzioni che possono aiutarci per le attività previste nei primi tre punti (ovviamente non per il quarto) utilizzando come esempio i dati ematologici e biometrici rilevati in 202 atleti australiani contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [2] dove trovate anche illustrati i dati contenuti nella tabella.

Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ANALISI ESPLORATIVA DEI DATI funzioni base
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
#
ais[!complete.cases(ais)] # verifica la presenza di NA (dati mancanti)
colSums(is.na(ais)) # conteggia gli eventuale dati mancanti per colonna
#
summary(ais) # statistiche elementari per tutte le variabili (numeriche e non) di ais
#
media <- mean(ais$ferr) # calcola la media della ferritina
mediana <- median(ais$ferr) # calcola la mediana
data.frame(media, mediana) # le mette a confronto
#
ds <- sd(ais$ferr) # calcola la deviazione standard della ferritina
mad <- mad(ais$ferr) # calcola la Median Absolute Deviation (about the median) o MAD
data.frame(ds, mad) # le mette a confronto
#
qpar <- round(qnorm(c(seq(0.025, 0.975, 0.025)), mean=mean(ais$ferr),
sd=sd(ais$ferr)), digits=2) # calcola i quantili parametrici della ferritina
```

qnon <- round(quantile(ais$ferr, probs=seq (0.025, 0.975, 0.025)), digits=2) # calcola i quantili non parametrici
data.frame(qpar, qnon) # li mette a confronto
#
windows() # apre e inizializza una nuova finestra grafica
boxplot(ais$ferr, range=1.5, horizontal=FALSE, main="Valori oltre la mediana ± 1.5 · IQR", ylab="Ferritina in µg/L", notch=FALSE, col="yellow") # rappresenta graficamente i valori della ferritina
boxplot.stats(ais$ferr, coef=1.5)$stats # mostra i 5 punti notevoli dei baffi e della scatola
boxplot.stats(ais$ferr, coef=1.5)$out # mostra i valori che si trovano oltre la mediana ± 1.5 volte il range interquartile (IQR)
#

Con la prima riga di codice viene caricato il pacchetto che contiene la tabella **ais** con i dati che ci servono.

Quindi con la funzione **complete.cases()** viene fatta la ricerca dei casi con dati non (**!**) completi, nei quali dovremmo avere i dati mancanti sostituiti con la sigla NA (Not Available).

```
> ais[!complete.cases(ais)] # verifica la presenza di NA (dati mancanti)
data frame con 0 colonne e 202 righe
```

Con la funzione **colSums()** sono ricercati i dati mancanti **is.na()** per ciascuna delle colonne/variabili della tabella:

```
> colSums(is.na(ais)) # conteggia gli eventuale dati mancanti per colonna
   rcc    wcc     hc     hg   ferr    bmi    ssf pcBfat    lbm     ht     wt
     0      0      0      0      0      0      0      0      0      0      0
   sex  sport
     0      0
```

Ora sappiamo che nella tabella i dati sono completi, non abbiamo dati mancanti.

Segue la funzione **summary()** che per ciascuna variabile della tabella riporta valore minimo (`Min.`), primo quartile (`1st Qu.`), mediana o secondo quartile (`Median`), media (`Mean`), terzo quartile (`3rd Qu.`), valore massimo (`Max.`).

```
> summary(ais) # statistiche elementari per tutte le variabili (numeriche e non) di ais
      rcc             wcc              hc              hg
 Min.   :3.800   Min.   : 3.300   Min.   :35.90   Min.   :11.60
 1st Qu.:4.372   1st Qu.: 5.900   1st Qu.:40.60   1st Qu.:13.50
 Median :4.755   Median : 6.850   Median :43.50   Median :14.70
 Mean   :4.719   Mean   : 7.109   Mean   :43.09   Mean   :14.57
 3rd Qu.:5.030   3rd Qu.: 8.275   3rd Qu.:45.58   3rd Qu.:15.57
 Max.   :6.720   Max.   :14.300   Max.   :59.70   Max.   :19.20

      ferr             bmi             ssf            pcBfat
 Min.   :  8.00   Min.   :16.75   Min.   : 28.00   Min.   : 5.630
 1st Qu.: 41.25   1st Qu.:21.08   1st Qu.: 43.85   1st Qu.: 8.545
 Median : 65.50   Median :22.72   Median : 58.60   Median :11.650
 Mean   : 76.88   Mean   :22.96   Mean   : 69.02   Mean   :13.507
 3rd Qu.: 97.00   3rd Qu.:24.46   3rd Qu.: 90.35   3rd Qu.:18.080
 Max.   :234.00   Max.   :34.42   Max.   :200.80   Max.   :35.520

      lbm              ht              wt         sex         sport
```

```
 Min.   : 34.36   Min.   :148.9   Min.   : 37.80   f:100   Row    :37
 1st Qu.: 54.67   1st Qu.:174.0   1st Qu.: 66.53   m:102   T_400m :29
 Median : 63.03   Median :179.7   Median : 74.40           B_Ball :25
 Mean   : 64.87   Mean   :180.1   Mean   : 75.01           Netball:23
 3rd Qu.: 74.75   3rd Qu.:186.2   3rd Qu.: 84.12           Swim   :22
 Max.   :106.00   Max.   :209.4   Max.   :123.20           Field  :19
                                                           (Other):47
```

In una distribuzione gaussiana media e mediana sono identiche [1]. Qui si osservano tra le due in genere valori abbastanza simili, tranne che nel caso della ferritina (`ferr`), della somma dello spessore delle pliche cutanee (`ssf`) e della percentuale di grasso corporeo (`pcBfat`).

Continuiamo quindi con l'analisi esplorativa dei dati, che per semplicità è qui limitata alla ferritina, calcolando e confrontando media e mediana

```
> media <- mean(ais$ferr) # calcola la media della ferritina
> mediana <- median(ais$ferr) # calcola la mediana
> data.frame(media, mediana) # le mette a confronto
     media mediana
1 76.87624    65.5
```

calcolando e confrontando deviazione standard e MAD

```
> ds <- sd(ais$ferr) # calcola la deviazione standard della ferritina
> mad <- mad(ais$ferr) # calcola la Median Absolute Deviation (about the median) o
MAD
> data.frame(ds, mad) # le mette a confronto
        ds     mad
1 47.50124 37.8063
```

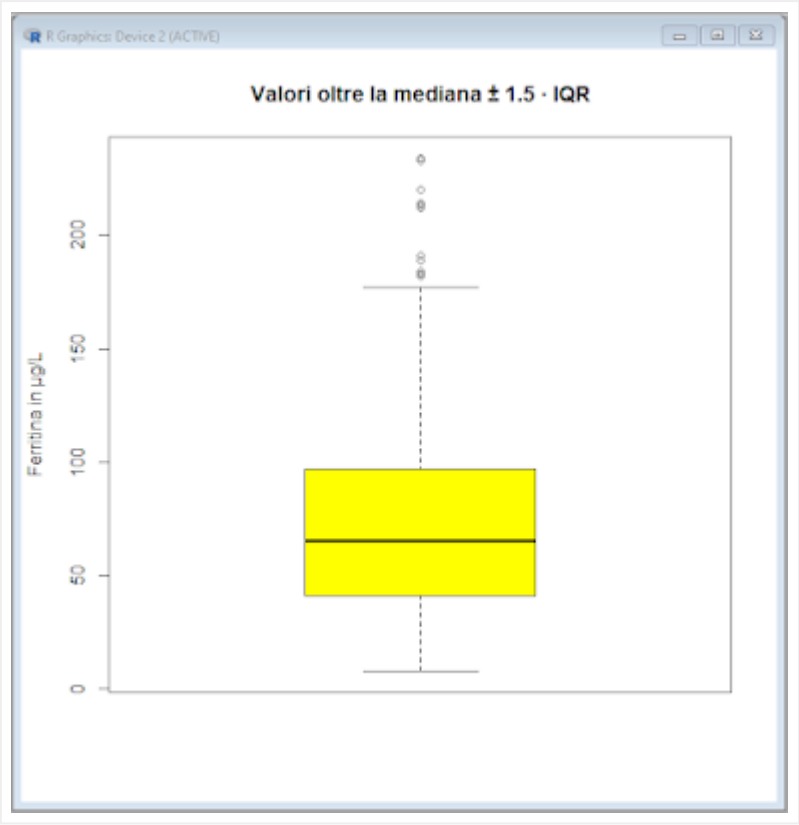
e infine calcolando e confrontando quantili parametrici e quantili non parametrici

```
> qpar <- round(qnorm(c(seq (0.025, 0.975, 0.025)), mean = mean(ais$ferr), sd =
sd(ais$ferr)), digits=2) # calcola i quantili parametrici della ferritina
> qnon <- round(quantile(ais$ferr, probs = seq (0.025, 0.975, 0.025)), digits=2) #
calcola i quantili non parametrici
> data.frame(qpar, qnon) # li mette a confronto
        qpar   qnon
2.5%  -16.22  20.00
5%     -1.26  22.00
7.5%    8.50  27.15
10%    16.00  30.00
12.5%  22.23  33.12
15%    27.64  35.15
17.5%  32.48  37.18
20%    36.90  39.20
22.5%  40.99  41.00
25%    44.84  41.25
27.5%  48.48  43.00
30%    51.97  44.00
32.5%  55.32  48.00
35%    58.57  50.00
37.5%  61.74  53.00
40%    64.84  55.00
42.5%  67.89  58.00
45%    70.91  59.45
```

```
47.5%  73.90  62.48
50%    76.88  65.50
52.5%  79.85  68.53
55%    82.85  71.00
57.5%  85.86  73.00
60%    88.91  76.00
62.5%  92.01  80.00
65%    95.18  85.65
67.5%  98.43  87.68
70%   101.79  90.70
72.5% 105.27  93.73
75%   108.92  97.00
77.5% 112.76 102.00
80%   116.85 107.00
82.5% 121.27 114.13
85%   126.11 122.00
87.5% 131.52 125.88
90%   137.75 138.40
92.5% 145.26 155.93
95%   155.01 182.95
97.5% 169.98 212.00
```

Nel caso di una distribuzione gaussiana tutti questi valori devono essere identici [1] e qui non lo sono.

Per l'identificazione degli outliers impieghiamo la funzione **boxplot()** realizzando un grafico a scatola con i baffi (boxplot) nel quale sono riportati come singoli punti separati i dati che si trovano oltre la mediana ± 1.5 volte il range interquartile (argomento **range=1.5**) [4].



Con le due ultime righe dello script sono quindi presentati due risultati della funzione **boxplot.stats()**.

Con **$stats** sono riportati i valori delle ferritina corrispondenti:

→ al baffo inferiore (8.0)
→ al margine inferiore della scatola (41.0)
→ alla mediana (65.5)
→ al margine superiore della scatola (97.0)
→ al baffo superiore (177.0)

```
> boxplot.stats(ais$ferr, coef=1.5)$stats # mostra i 5 punti notevoli dei baffi e della scatola
[1]   8.0  41.0  65.5  97.0 177.0
```

Con **$out** sono riportati i dati anomali (182 183 212 213 184 220 191 189 212 234 214 233), o outliers.

```
> boxplot.stats(ais$ferr, coef=1.5)$out # mostra i valori che si trovano oltre la mediana ± 1.5 volte il range interquartile (IQR)
 [1] 182 183 212 213 184 220 191 189 212 234 214 233
```

Si tratta dei dati che nel grafico si trovano oltre la mediana ± 1.5 volte il range interquartile (argomento **coef=1.5**), che non per questo devono essere esclusi dalle statistiche, ma sui quali sarebbe opportuno, dato il loro eccessivo scostamento dai dati rimanenti, effettuare una rivalutazione (cosa qui ovviamente impraticabile non avendo accesso alla complessa catena di eventi che ha portato alla produzione dei dati).

Oltre alla funzione **summary()** inclusa nel pacchetto base di **R** [5] altre funzioni [6] per l'analisi esplorativa dei dati sono disponibili nei pacchetti **Hmisc**, **pastecs**, **psych**.

Potete scaricare e installare i pacchetti aggiuntivi dal **CRAN** e trovate la loro documentazione completa, incluso il manuale di riferimento, sul repository della documentazione di **R** [7].

Copiate questo script, incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# ANALISI ESPLORATIVA DEI DATI funzioni per una analisi globale
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
mydata <- ais[c(1,2,3,4,5,6,7,8,9,10,11)] # salva le colonne con le sole variabili numeriche in mydata
#
summary(mydata) # statistiche elementari per tutte le variabili
#
library(Hmisc) # carica il pacchetto
describe(mydata) # statistiche del pacchetto Hmisc
#
library(pastecs) # carica il pacchetto
stat.desc(mydata) # statistiche del pacchetto pastecs
#
library(psych) # carica il pacchetto
describe(mydata) # statistiche del pacchetto psych
describeBy(mydata, ais$sex) # statistiche del pacchetto psych separate per sesso
describeBy(mydata, ais$sport) # statistiche del pacchetto psych separate per sport
#
```

Le funzioni espandono, ciascuna a modo suo, il quadro dei dati fornito dalla funzione **summary()** del *pacchetto base* di **R**, e sono state riportate perché ciascuno ne possa valutare i risultati e l'utilità per gli scopi che si propone.

L'aspetto forse più interessante è rappresentato dalla possibilità offerta dalla funzione **describeBy()** del pacchetto **psych** di riportare statistiche riepilogative riaggregando i dati in base ai fattori presenti nei record, e quindi, nel caso specifico, di riportare statistiche separate per sesso (m,f), come pure statistiche separate per ciascuno degli sport praticati dagli atleti (B_Ball, Field, Gym, Netball, Row, Swim, T_400m, T_Sprnt, Tennis, W_Polo).

Conclusione: l'analisi esplorativa dei dati fornisce informazioni utili a evidenziare per la ferritina alcuni problemi che meritano di essere approfonditi, valutati e opportunamente gestiti continuando a seguire le fasi del percorso logico riportato all'inizio.

Potete riutilizzare facilmente lo script sostituendo all'oggetto **ais** l'oggetto contenente i vostri dati, opportunamente strutturati. Per una guida rapida all'importazione dei dati potete consultare i link:
→ importare i dati di un file .csv
→ importare i dati di un file .xls o .xlsx
→ gestione dei dati mancanti

---

[1] Una tipica distribuzione gaussiana è riportata nel post: La distribuzione gaussiana.

[2] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**. Per manuale di riferimento del pacchetto vedere nel repository della documentazione: *Package 'DAAG'*. URL consultato il 20/01/2023: https://goo.gl/gzKabK

[3] La "*Median Absolute Deviation (about median)*" o MAD ovvero la deviazione assoluta mediana (dalla mediana) è l'equivalente non parametrico della deviazione standard e in una distribuzione gaussiana ha lo stesso valore di questa. Vedere: Rousseeuw PJ, Croux C. *Alternatives to the Median Absolute Deviation*. Journal of the American Statistical Association 88 (424), 1273-1283, 1993. URL consultato il 04/01/2019: https://goo.gl/4Rh53b

[4] Per i dettagli sulla funzione **boxplot()** digitare **help(***boxplot***)** nella `Console di R` e vedere il post Grafici a scatola con i baffi.

[5] Vedere il manuale di riferimento del pacchetto base *R: A Language and Environment for Statistical Computing, Reference Index*. URL consultato il 28/11/2018: https://goo.gl/kes91X

[6] Per la loro documentazione digitate come al solito **help(***nomedellafunzione***)** nella `Console di R`.

[7] *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

martedì 8 gennaio 2019

Test di normalità (gaussianità)

Siamo al secondo passo delle valutazioni preliminari che è sempre necessario effettuare nell'ambito di un percorso logico che prevede, per il calcolo delle statistiche elementari di una singola variabile (analisi univariata), i seguenti passi:
→ analisi esplorativa dei dati;
→ esecuzione dei **test di normalità (gaussianità)** per valutare se i dati seguono una distribuzione gaussiana [1];
→ calcolo delle statistiche elementari parametriche (media, deviazione standard, varianza, quantili parametrici) se i dati seguono una distribuzione gaussiana [2];
→ calcolo delle statistiche elementari non parametriche (mediana, deviazione assoluta mediana o MAD, quartili e altri quantili non parametrici) se i dati non seguono una distribuzione gaussiana.

I dati sono contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [3]. Dovete scaricare e installare anche i pacchetti **moments** e **nortest**. Trovate la loro documentazione completa, incluso il manuale di riferimento, sul repository della documentazione di **R** [4].

Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# TEST DI NORMALITA' (GAUSSIANITA')
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
library(moments) # carica il pacchetto per asimmetria e curtosi
library(nortest) # carica il pacchetto per ulteriori test di normalità
#
mydata <- as.matrix(ais[c(5)]) # ci interessa la sola colonna con la ferritina
#
options(scipen=999) # esprime i numeri in formato fisso
#
agostino.test(mydata) # test di D'Agostino per il coefficiente di asimmetria
anscombe.test(mydata) # test di Anscombe-Glynn per il coefficiente di curtosi
#
ad.test(mydata) # test di Anderson-Darling per la normalità
cvm.test(mydata) # test di Cramer-von Mises per la normalità
pearson.test(mydata) # test chi-quadrato di Pearson per la normalità
sf.test(mydata) # test di Shapiro-Francia per la normalità
#
options(scipen=0) # ripristina la notazione scientifica
#
```

Dopo avere installato i tre pacchetti necessari, selezionato la colonna **5** contenente i dati che ci interessano, imposto di riportare i numeri in formato fisso con **scipen=999**, sono eseguiti il test di asimmetria, il test di curtosi e altri quattro test di normalità per la variabile **ferritina** [5]:

```
> agostino.test(mydata) # test di D'Agostino per il coefficiente di asimmetria

        D'Agostino skewness test

data:  mydata
skew = 1.2806, z = 6.1376, p-value = 0.0000000008377
alternative hypothesis: data have a skewness

> anscombe.test(mydata) # test di Anscombe-Glynn per il coefficiente di curtosi
```

```
        Anscombe-Glynn kurtosis test

data:  mydata
kurt = 4.4202, z = 2.9449, p-value = 0.003231
alternative hypothesis: kurtosis is not equal to 3

> ad.test(mydata) # test di Anderson-Darling per la normalità

        Anderson-Darling normality test

data:  mydata
A = 6.097, p-value = 0.000000000000004904

> cvm.test(mydata) # test di Cramer-von Mises per la normalità

        Cramer-von Mises normality test

data:  mydata
W = 0.94473, p-value = 0.000000002495

> pearson.test(mydata) # test chi-quadrato di Pearson per la normalità

        Pearson chi-square normality test

data:  mydata
P = 63.772, p-value = 0.00000002531

> sf.test(mydata) # test di Shapiro-Francia per la normalità

        Shapiro-Francia normality test

data:  mydata
W = 0.89138, p-value = 0.000000001251
```

Tutti i test di normalità concordano - con valori di `p` sempre di molto inferiori al valore soglia tradizionalmente impiegato di `p=0.05` - nell'attribuire ai valori della ferritina una distribuzione non gaussiana [6].

Se nello script sostituite **ais[c(5)]** con **ais[c(10)]** potete eseguire i test di normalità sulla variabile **altezza** (espressa in centimetri, cm), ottenendo questi risultati:

```
> agostino.test(mydata) # test di D'Agostino per il coefficiente di asimmetria

        D'Agostino skewness test

data:  mydata
skew = -0.1993, z = -1.1856, p-value = 0.2358
alternative hypothesis: data have a skewness

> anscombe.test(mydata) # test di Anscombe-Glynn per il coefficiente di curtosi

        Anscombe-Glynn kurtosis test

data:  mydata
kurt = 3.5281, z = 1.5412, p-value = 0.1233
```

```
alternative hypothesis: kurtosis is not equal to 3

> ad.test(mydata) # test di Anderson-Darling per la normalità

        Anderson-Darling normality test

data:  mydata
A = 0.4462, p-value = 0.2793

> cvm.test(mydata) # test di Cramer-von Mises per la normalità

        Cramer-von Mises normality test

data:  mydata
W = 0.058974, p-value = 0.3886

> pearson.test(mydata) # test chi-quadrato di Pearson per la normalità

        Pearson chi-square normality test

data:  mydata
P = 17.653, p-value = 0.223

> sf.test(mydata) # test di Shapiro-Francia per la normalità

        Shapiro-Francia normality test

data:  mydata
W = 0.98919, p-value = 0.1174
```

I risultati, con valori sempre abbondantemente superiori al `p=0.05`, ci dicono che questa volta siamo di fronte a dati distribuiti in modo gaussiano.

A conferma dei valori numerici forniti dai test di normalità aggiungiamo ora quattro grafici, partendo da quello della **ferritina**. Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

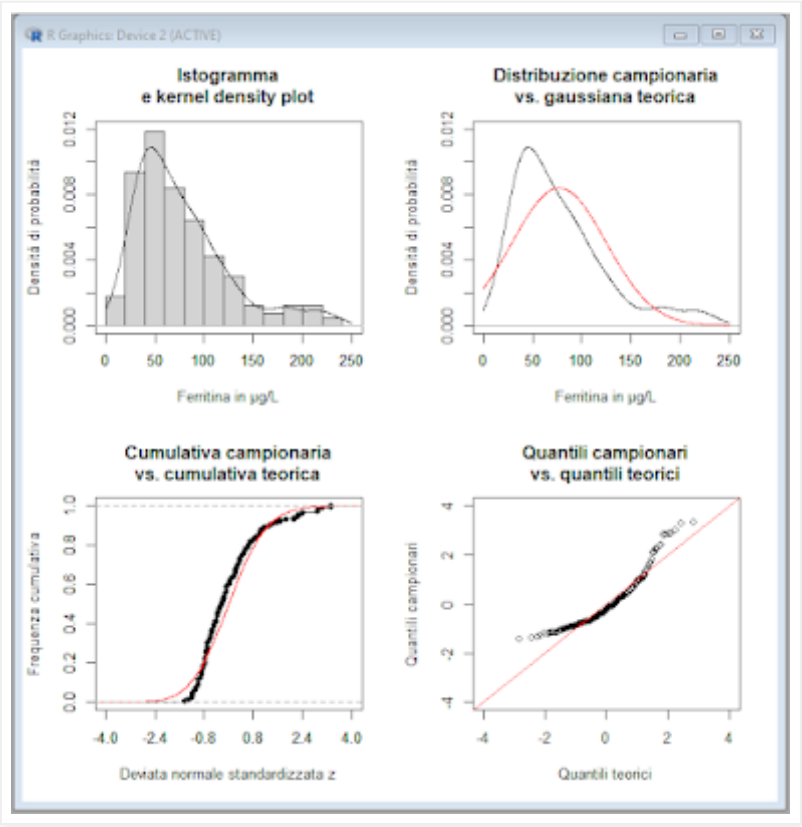
```
# VALUTAZIONE GRAFICA DELLA NORMALITA' (GAUSSIANITA')
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
#
mydata <- as.matrix(ais[c(5)]) # ci interessa la sola colonna con la ferritina
#
windows() # apre una nuova finestra
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
# istogramma e kernel density plot
#
hist(mydata, xlim=c(0, 250), ylim=c(0, 0.020), freq=FALSE, breaks=25,
main="Istogramma\ne kernel density plot", xlab="Ferritina in µg/L", ylab="Densità di
probabilità") # traccia l'istogramma dei dati
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
plot(density(mydata, n=1000, from=0, to=250), xlim=c(0,250), ylim=c(0,
0.020), yaxt="n", col="black") # sovrappone il kernel density plot della distribuzione campionaria
#
# distribuzione campionaria vs. gaussiana teorica
#
```

plot(density(mydata, n=1024, from=0, to=250), xlim=c(0, 250), ylim=c(0, 0.012)) # traccia il kernel density plot della distribuzione campionaria
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
x <- seq(0, 250, length.out=1000) # calcola i valori in ascisse della gaussiana teorica
y <- dnorm(x, mean=mean(mydata), sd=sd(mydata)) # calcola i valori in ordinate della gaussiana teorica
plot(x, y, xlim=c(0, 250), ylim=c(0, 0.012), yaxt="n", col="red", type="l") # sovrappone la distribuzione gaussiana teorica in colore rosso
title(main="Distribuzione campionaria\nvs. gaussiana teorica", xlab="Ferritina in µg/L", ylab="Densità di probabilità") # aggiunge titolo e legende degli assi
#
distribuzione cumulativa campionaria vs. distribuzione cumulativa teorica
#
par(new=FALSE, ann=TRUE) # per mostrare titolo e legende degli assi
plot(ecdf(scale(mydata)), main="Cumulativa campionaria\nvs. cumulativa teorica", xlab="Deviata normale standardizzata z", ylab="Frequenza cumulativa", xlog = FALSE, ylog = FALSE, xlim=c(-4, 4), ylim=c(0, 1), xaxp=c(-4, 4, 5), yaxp=c(0, 1, 5)) # traccia il grafico della distribuzione cumulativa campionaria
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
plot(ecdf(rnorm(10000, mean=0, sd=1)), col="red", xlog=FALSE, ylog=FALSE, xlim=c(-4, 4), ylim=c(0, 1), xaxp=c(-4, 4, 5), yaxp=c(0, 1, 5)) # sovrappone la distribuzione cumulativa teorica in colore rosso
#
quantili campionari vs. quantili teorici
#
par(new=FALSE, ann=TRUE) # per mostrare titolo e legende degli assi
qqnorm(scale(mydata), main="Quantili campionari\nvs. quantili teorici", xlab="Quantili teorici", ylab="Quantili campionari", xlim=c(-4, 4), ylim=c(-4, 4)) # traccia il grafico della distribuzione dei quantili campionari
abline (0, 1, col="red") # sovrappone la distribuzione dei quantili teorica in colore rosso
#

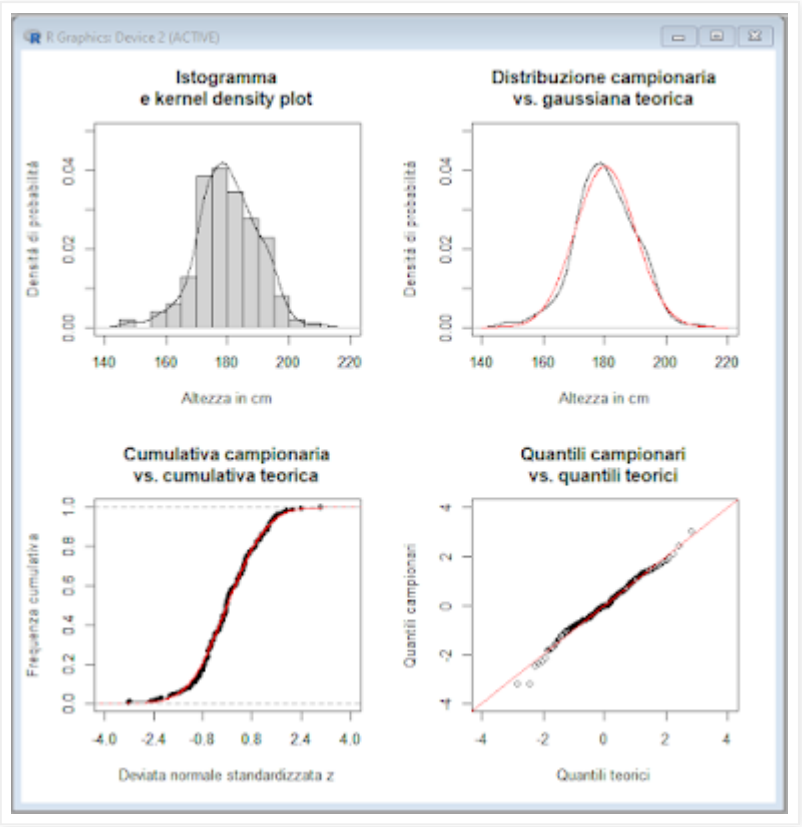
Con la funzione **par(mfrow=c(2,2))** la finestra grafica viene suddivisa in quattro quadranti, che verranno riempiti da sinistra a destra e dall'alto verso il basso dai quattro grafici generati con i successivi blocchi di codice, migliorando la sintesi dei risultati.

Per le funzioni qui riportate [7] sono previsti molti altri possibili argomenti, per i quali si rimanda alla documentazione di **R**. Quelli che vi capiterà di utilizzare più frequentemente, a parte ovviamente la variabile da analizzare, che è sempre il primo argomento in ogni funzione, sono:
→ **main**, il titolo del grafico, che compare in alto;
→ **xlab**, l'etichetta dell'asse delle ascisse x;
→ **ylab**, l'etichetta dell'asse delle ordinate y;
→ **xlim**, che indica limite inferiore e limite superiore della scala da applicare all'asse delle ascisse x;
→ **ylim**, che indica limite inferiore e limite superiore della scala da applicare all'asse delle ordinate y;
→ **xaxp**, che specifica il limite inferiore, il limite superiore e il numero di intervalli da applicare alla scala dell'asse delle ascisse x;
→ **yaxp**, che specifica il limite inferiore, il limite superiore e il numero di intervalli da applicare alla scala dell'asse delle ordinate y.

Questi sono i quattro grafici risultanti, con i dati campionari che confermano che i valori di concentrazione delle ferritina nei 202 atleti australiani non sono distribuiti in modo gaussiano.



Se nello script sostituite **ais[c(5)]** con **ais[c(10)]** e aggiustate le scale degli assi potete visualizzare il grafico per i valori per la variabile **altezza**.



I grafici confermano quanto avevamo già visto con i test di normalità: i valori di altezza nei 202 atleti australiani sono distribuiti in modo gaussiano.

Conclusione:
→ nel caso della ferritina i dati non sono distribuiti in modo gaussiano, pertanto dovremo impiegare metodi non parametrici e calcoleremo la mediana, la deviazione assoluta mediana o MAD e i quantili non parametrici;

→ nel caso dell'altezza i dati sono distribuiti in modo gaussiano, pertanto potremo impiegare metodi parametrici e calcoleremo la media, la deviazione standard e i quantili parametrici.

---

[1] Ghasemi A, Zahediasl S. *Normality Tests for Statistical Analysis: A Guide for Non-Statisticians*. Int J Endocrinol Metab. 2012;10(2);486–489. URL consultato il 05/01/2019: https://goo.gl/AkoUdn

[2] Per il quadro tipico di una distribuzione gaussiana vedere il post: La distribuzione gaussiana

[3] Vedere i dati ematologici e i dati biometrici rilevati in 202 atleti australiani nel post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[4] *Available CRAN Packages By Name*. URL consultato il 05/01/2019: https://goo.gl/hLC9BB

[5] La ferritina è una proteina che, pur con alcune limitazioni, fornisce una misura dei depositi di ferro presenti nell'organismo, necessari per garantire una adeguata produzione di emoglobina. La concentrazione è espressa in µg/L.

[6] Quei stessi stessi test di normalità sono riportati, in una tabella più compatta, nel post Tabulare una serie di test di normalità (gaussianità)

[7] Per la documentazione delle funzioni digitare **help(***nomedellafunzione***)** nella `Console di R`. Per gli argomenti **xlog**, **ylog**, **xlim**, **ylim**, **xaxp**, **yaxp** vedere anche la funzione **par()** con **help(par)**.

mercoledì 9 gennaio 2019

Normale o gaussiana?

In un sito di statistica il riferimento è evidente: distribuzione normale o distribuzione gaussiana? E di conseguenza: test di normalità o test di gaussianità?

Per basare la risposta su qualche dato oggettivo, anche se di per sé non dirimente, ho effettuato, sul motore di ricerca che rende disponibile anche lo spazio web di questo blog, queste query [1], mettendo il testo all'interno dei doppi apici al fine di effettuare una ricerca "esatta" delle espressioni:

Espressione	**Numero di risultati**
"distribuzione normale"	112 000
"distribuzione gaussiana"	26 000
"test di normalità"	44 700
"test di gaussianità"	319

Il rapporto tra il numero di risultati dell'espressione "*distribuzione normale*" e dell'espressione "*distribuzione gaussiana*" è all'incirca di 4 a 1. Il divario è importante, ma non ancora tale da far sì che il numero dei risultati possa rappresentare il razionale per fare optare decisamente per l'una o per l'altra delle due espressioni.

Ho quindi cercato "*normal*e" su un vocabolario. Per rendere riproducibile e consultabile da chiunque il risultato ho effettuato la ricerca online sul "*Vocabolario*" della italianissima "*Treccani*".

Ebbene, nel vocabolario l'espressione "*normale*" è come prima cosa riferita al punto **1** come "*perpendicolare*", al punto **2** come "*che segue la norma*".

Solamente dopo questi due significati principali sono ripresi al punto **3** ed elencati in quanto "*... sign. specifici in varie scienze e discipline*" gli altri che il termine può avere: da quelli più generali a quelli che in campo scientifico vanno dalla botanica alla chimica, dalla fisica alla matematica e (finalmente) - alla lettera **h** (cioè ben all'ottavo posto del punto **3**) - alla statistica.

L'inflazione di significati attribuiti al termine "*normale*" e la sua conseguente svalutazione per l'impiego in campo statistico forniscono il razionale per passare all'impiego sistematico dell'espressione "**distribuzione gaussiana**", che io qui ho ritenuto pertanto di adottare.

Molto diversa invece è la situazione che riguarda i test per verificare il tipo di distribuzione dei dati in esame. Perché se, per analogia con l'aggettivo riferito alla distribuzione, non dovremmo più parlare di "*test di normalità*" ma dovremmo parlare di "*test di gaussianità*", questa soluzione pare essere stata adottata in meno dell'1% dei casi. A causa di questo e nonostante vada in senso opposto alla scelta precedente, ho finito per adottare in questo sito l'espressione "**test di normalità (gaussianità)**" affiancando salomonicamente le due espressioni. Anche se spero che prima o poi la "*gaussianità*" possa uscire dalla parentesi in cui l'ho dovuta relegare per la necessità di adeguarmi (spero temporaneamente) alla "*normalità*" [2].

---

[1] Dati aggiornati al gennaio 2023.

[2] Quanto sia considerato normale, ancora attualmente, nella letteratura scientifica, l'impiego dell'espressione "test di normalità", lo si può desumere anche dai titoli di due recenti lavori sull'argomento:

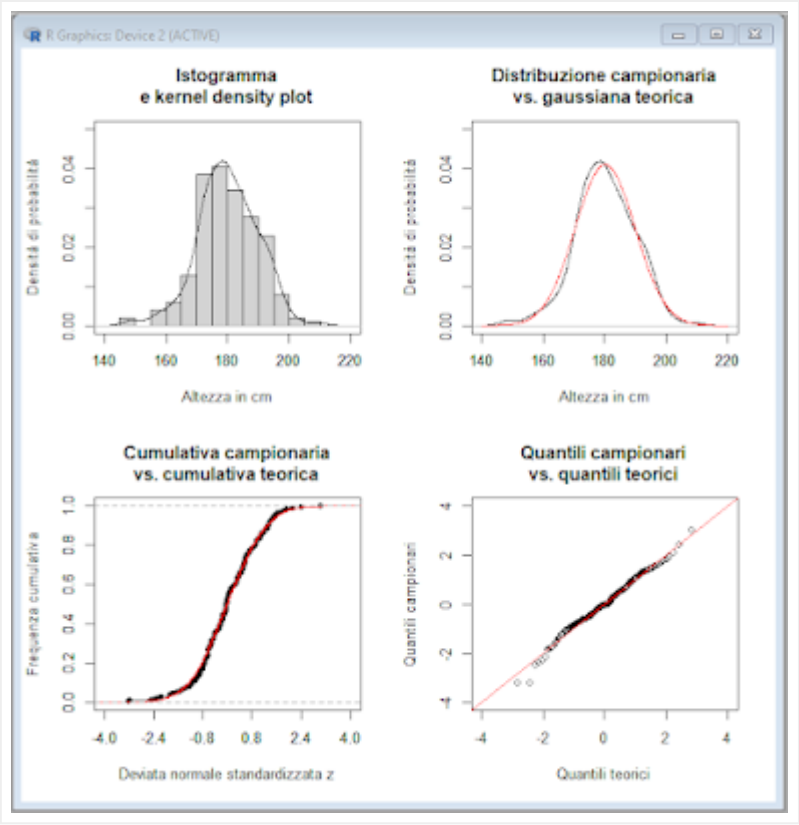
→ Székely, G. J. and Rizzo, M. L. *A new test for multivariate normality*. Journal of Multivariate Analysis 2005;93;58-80. URL consultato il 05/01/2019: https://doi.org/10.1016/j.jmva.2003.12.002
→ Ghasemi A, Zahediasl S. *Normality Tests for Statistical Analysis: A Guide for Non-Statisticians*. Int J Endocrinol Metab. 2012;10(2);486–489. URL consultato il 05/01/2019: https://goo.gl/AkoUdn

giovedì 10 gennaio 2019

Statistiche elementari parametriche

Siamo al terzo passo delle valutazioni che è sempre necessario effettuare nell'ambito di un percorso logico che prevede, per il calcolo delle statistiche elementari di una singola variabile (analisi univariata):
→ analisi esplorativa dei dati;
→ esecuzione dei test di normalità (gaussianità) per valutare se i dati seguono una distribuzione gaussiana;
→ calcolo delle **statistiche elementari parametriche** (media, deviazione standard, varianza, quantili parametrici) se i dati seguono una distribuzione gaussiana;
→ calcolo delle statistiche elementari non parametriche (mediana, deviazione assoluta mediana o MAD, quartili e altri quantili non parametrici) se i dati non seguono una distribuzione gaussiana.

Si tratta del passo conclusivo per la variabile **altezza** (espressa in cm) rilevata in 202 atleti australiani, in quanto con i test di normalità abbiamo stabilito che la variabile è distribuita in modo gaussiano: questa era la sintesi grafica dei risultati.



Ora copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# STATISTICHE ELEMENTARI PARAMETRICHE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
mydata <- as.matrix(ais[c(10)]) # ci interessa la sola colonna con l'altezza
#
# statistiche elementari parametriche
#
min(mydata) # valore minimo
mean(mydata) # media
max(mydata) # valore massimo
```

range(mydata) # range
max(mydata)-min(mydata) # campo di variazione
var(mydata) # varianza
sd(mydata) # deviazione standard
(sd(mydata)/mean(mydata))*100 # coefficiente di variazione CV %
#
confronta media e mediana
#
media <- mean(mydata) # calcola la media
mediana <- median(mydata) # calcola la mediana
data.frame(media, mediana) # le mette a confronto
#
confronta deviazione standard e MAD
#
ds <- sd(mydata) # calcola la deviazione standard
mad <- mad(mydata) # calcola la Median Absolute Deviation (about the median) o MAD
data.frame(ds, mad) # le mette a confronto
#
confronta quantili parametrici e non parametrici
#
qpar <- round(qnorm(c(seq(0.025, 0.975, 0.025)), mean=mean(mydata), sd=sd(mydata)), digits=2) # calcola i quantili parametrici
qnon <- round(quantile(mydata, probs=seq(0.025, 0.975, 0.025)), digits=2) # calcola i quantili non parametrici
dper <- abs(round(100*(qpar-qnon)/((qpar+qnon)/2), digits=2)) # calcola la differenza in percentuale
data.frame(qpar, qnon, dper) # li mette a confronto
#

Non avrebbe senso ripetere l'analisi esplorativa dei dati, i test di normalità e la sintesi grafica dei risultati - per questi si rimanda agli script riportati nel percorso logico indicato all'inizio. Per cui qui ci limitiamo a calcolare le usuali statistiche elementari parametriche, riportando per confronto i risultati delle statistiche elementari non parametriche corrispondenti.

Dopo avere caricato il pacchetto **DAAG** [1] con la tabella **ais**, la variabile che ci interessa, contenuta nella colonna **10** della tabella **ais** viene assegnata (**<-**) all'oggetto **mydata**, sul quale sono calcolate le statistiche elementari parametriche con le funzioni:
→ **min()** per riportare il valore minimo osservato;
→ **mean()** per calcolare la media;
→ **max()** per riportare il valore massimo osservato;
→ **range()** per calcolare il range, che qui è correttamente;
→ **max(mydata)-min(mydata)** per calcolare il campo di variazione, talora indicato (impropriamente) come "range";
→ **var()** per calcolare la varianza,
→ **sd()** per calcolare la deviazione standard;
→ e con l'espressione **(sd(mydata)/mean(mydata))*100** che calcola il coefficiente di variazione percentuale.

```
> min(mydata) # valore minimo
[1] 148.9
> mean(mydata) # media
[1] 180.104
> max(mydata) # valore massimo
[1] 209.4
> range(mydata) # range
[1] 148.9 209.4
```

```
> max(mydata)-min(mydata) # campo di variazione
[1] 60.5
> var(mydata) # varianza
         ht
ht 94.76038
> sd(mydata) # deviazione standard
[1] 9.734494
> (sd(mydata)/mean(mydata))*100 # coefficiente di variazione CV %
[1] 5.404931
```

Sono poi messe a confronto la media con la mediana

```
> media <- mean(mydata) # calcola la media
> mediana <- median(mydata) # calcola la mediana
> data.frame(media, mediana) # le mette a confronto
    media mediana
1 180.104   179.7
```

la deviazione standard con la MAD [2]

```
> ds <- sd(mydata) # calcola la deviazione standard
> mad <- mad(mydata) # calcola la Median Absolute Deviation (about the median) o MAD
> data.frame(ds, mad) # le mette a confronto
        ds     mad
1 9.734494 8.96973
```

e i quantili parametrici con i quantili non parametrici, riportando nell'ultima colonna la differenza percentuale tra i due

```
> qpar <- round(qnorm(c(seq(0.025, 0.975, 0.025)), mean=mean(mydata), sd=sd(mydata)),
digits=2) # calcola i quantili parametrici
> qnon <- round(quantile(mydata, probs=seq(0.025, 0.975, 0.025)), digits=2) # calcola
i quantili non parametrici
>  dper  <-  abs(round(100*(qpar-qnon)/((qpar+qnon)/2),  digits=2))  #  calcola  la
differenza in percentuale
> data.frame(qpar, qnon, dper) # li mette a confronto
        qpar   qnon dper
2.5%  161.02 158.98 1.28
5%    164.09 163.96 0.08
7.5%  166.09 167.35 0.76
10%   167.63 169.17 0.91
12.5% 168.91 170.36 0.85
15%   170.01 171.40 0.81
17.5% 171.01 172.22 0.71
20%   171.91 172.76 0.49
22.5% 172.75 173.52 0.44
25%   173.54 174.00 0.26
27.5% 174.29 174.18 0.06
30%   175.00 175.00 0.00
32.5% 175.69 175.73 0.02
35%   176.35 176.07 0.16
37.5% 177.00 177.30 0.17
40%   177.64 177.94 0.17
42.5% 178.26 178.44 0.10
45%   178.88 178.99 0.06
47.5% 179.49 179.60 0.06
```

```
50%   180.10 179.70 0.22
52.5% 180.71 180.10 0.34
55%   181.33 180.36 0.54
57.5% 181.94 181.00 0.52
60%   182.57 182.66 0.05
62.5% 183.21 183.06 0.08
65%   183.85 183.90 0.03
67.5% 184.52 184.67 0.08
70%   185.21 185.17 0.02
72.5% 185.92 185.60 0.17
75%   186.67 186.18 0.26
77.5% 187.46 187.28 0.10
80%   188.30 188.62 0.17
82.5% 189.20 189.18 0.01
85%   190.19 190.67 0.25
87.5% 191.30 191.94 0.33
90%   192.58 192.79 0.11
92.5% 194.12 193.86 0.13
95%   196.12 195.17 0.49
97.5% 199.18 197.48 0.86
```

Le differenze trascurabili tra statistiche elementari parametriche e statistiche elementari non parametriche riflettono il fatto che nel caso di una distribuzione perfettamente gaussiana [3] le due coincidono - è quindi assolutamente lecito impiegare le seconde al posto delle prime. Mentre il contrario è sbagliato - nel caso di una distribuzione non gaussiana le statistiche elementari parametriche non devono essere impiegate.

Lo script può essere riutilizzato molto semplicemente, assegnando (**<-**) all'oggetto **mydata** i propri dati**.**

---

[1] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[2] La Median Absolute Deviation about median o MAD ovvero la deviazione assoluta mediana (dalla mediana) è l'equivalente non parametrico della deviazione standard. Vedere: Rousseeuw PJ, Croux C. *Alternatives to the Median Absolute Deviation*. Journal of the American Statistical Association 88 (424), 1273-1283, 1993. URL consultato il 04/01/2019: https://goo.gl/4Rh53b

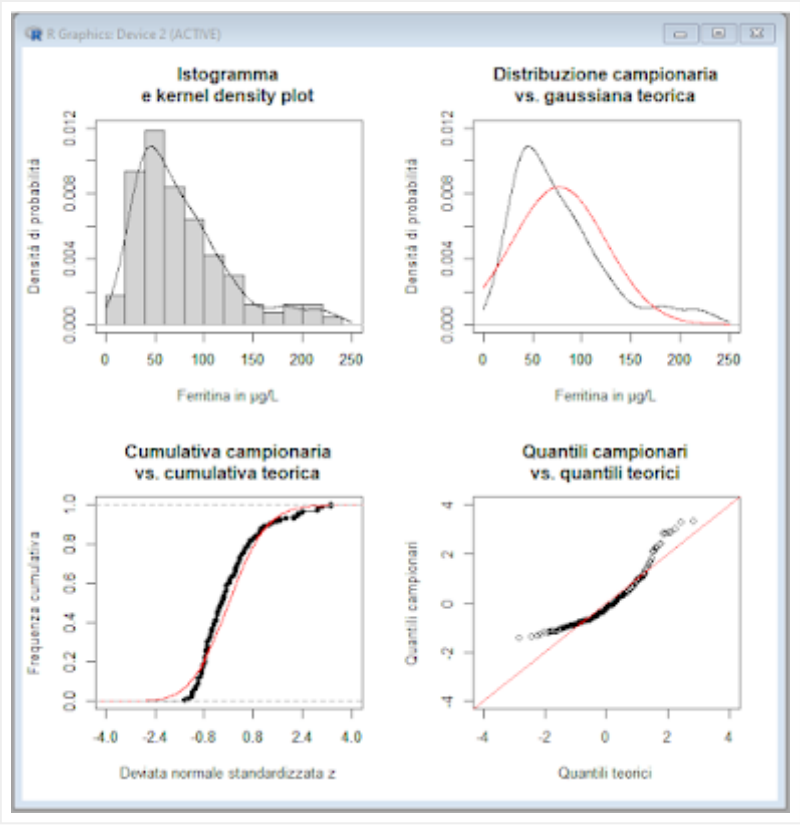
[3] Un esempio di dati distribuiti in modo perfettamente gaussiano è riportato nel post La distribuzione gaussiana.

venerdì 11 gennaio 2019

Statistiche elementari non parametriche

Siamo all'ultimo passo delle valutazioni preliminari che è sempre necessario effettuare nell'ambito di un percorso logico che prevede, per il calcolo delle statistiche elementari di una singola variabile (analisi univariata):
→ analisi esplorativa dei dati;
→ esecuzione dei test di normalità (gaussianità) per valutare se i dati seguono una distribuzione gaussiana;
→ calcolo delle statistiche elementari parametriche (media, deviazione standard, varianza, quantili parametrici) se i dati seguono una distribuzione gaussiana;
→ calcolo delle **statistiche elementari non parametriche** (mediana, deviazione assoluta mediana o MAD, quartili e altri quantili non parametrici) se i dati non seguono una distribuzione gaussiana.

Si tratta del passo conclusivo per la variabile concentrazione della **ferritina** (espressa in µg/L) rilevata in 202 atleti australiani, in quanto con i test di normalità abbiamo stabilito che la variabile non è distribuita in modo gaussiano cone risulta evidente anche dalla sintesi grafica dei risultati.



Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# STATISTICHE ELEMENTARI NON PARAMETRICHE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
mydata <- unlist(ais[c(5)]) # ci interessa la sola colonna con la ferritina
#
# statistiche elementari non parametriche
#
min(mydata) # valore minimo
median(mydata) # mediana
max(mydata) # valore massimo
```

range(mydata) # range
max(mydata)-min(mydata) # campo di variazione
mad(mydata) # deviazione assoluta mediana (MAD)
quantile(mydata) # quartili
#
quartili, decili e percentili della distribuzione campionaria
#
quantile(mydata, probs=seq (0, 1, 0.25)) # con 0.25 calcola i quartili della ferritina
quantile(mydata, probs=seq (0, 1, 0.1)) # con 0.1 calcola i decili
quantile(mydata, probs=seq (0, 1, 0.01)) # con 0.01 calcola i percentili
#

Non avrebbe senso ripetere l'analisi esplorativa dei dati, i test di normalità e la sintesi grafica dei risultati - per questi si rimanda agli script riportati nel percorso logico indicato all'inizio. Per cui qui ci limitiamo a calcolare le usuali statistiche elementari non parametriche.

Dopo avere caricato il pacchetto **DAAG** [1] contenente la tabella **ais**, la variabile `ferr` contenuta nella colonna **5** della tabella **ais** viene assegnata (**<-**) all'oggetto **mydata**, sul quale sono calcolate le statistiche elementari non parametriche con le funzioni:
→ **min()** per riportare il valore minimo osservato;
→ **median()** per calcolare la media;
→ **max()** per riportare il valore massimo osservato;
→ **range()** per calcolare il range qui correttamente inteso come "*intervallo (di valori di una grandezza)*";
→ **max(mydata)-min(mydata)** per calcolare il campo di variazione, talora indicato (impropriamente) con il termine "range";
→ **mad()** per calcolare la deviazione assoluta mediana MAD [2];
→ **quantile()** per calcolare i quartili.

```
> min(mydata) # valore minimo
[1] 8
> median(mydata) # mediana
[1] 65.5
> max(mydata) # valore massimo
[1] 234
> range(mydata) # range
[1]   8 234
> max(mydata)-min(mydata) # campo di variazione
[1] 226
> mad(mydata) # deviazione assoluta mediana (MAD)
[1] 37.8063
> quantile(mydata) # quartili
    0%    25%    50%    75%   100%
  8.00  41.25  65.50  97.00 234.00
```

Da notare che la funzione **quantile()**, se impiegata specificando come unico argomento il nome della variabile, prevede di default il calcolo dei quartili. Se si aggiunge l'argomento **probs** e si impiega la funzione **seq()** per stabilire la sequenza di valori per i quali calcolarli [3] è possibile calcolare i quantili desiderati, che in questo caso sono
→ i quartili impostando **seq (0, 1, 0.25)**
→ i decili impostando **seq (0, 1, 0.1)**
→ i percentili impostando **seq (0, 1, 0.01)**
essendo a loro volta gli argomenti della funzione **seq()** il quantile iniziale (**0** che corrisponde al valore minimo osservato), il quantile finale (**1** che corrisponde al valore massimo osservato) e l'incremento da applicare per il calcolo dei quantili intermedi. Modificando opportunamente questi tre argomenti è possibile calcolare qualsiasi altra sequenza di quantili.

```
> quantile(mydata, probs=seq (0, 1, 0.25)) # con 0.25 calcola i quartili della
ferritina
    0%    25%    50%    75%   100%
  8.00  41.25  65.50  97.00 234.00
> quantile(mydata, probs=seq (0, 1, 0.1)) # con 0.1 calcola i decili
   0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
  8.0  30.0  39.2  44.0  55.0  65.5  76.0  90.7 107.0 138.4 234.0
> quantile(mydata, probs=seq (0, 1, 0.01)) # con 0.01 calcola i percentili
    0%     1%     2%     3%     4%     5%     6%     7%     8%     9%    10%
  8.00  13.03  19.02  20.03  21.04  22.00  25.06  26.07  29.00  29.09  30.00
   11%    12%    13%    14%    15%    16%    17%    18%    19%    20%    21%
 30.22  32.12  34.00  34.14  35.15  36.00  36.17  38.00  39.00  39.20  40.00
   22%    23%    24%    25%    26%    27%    28%    29%    30%    31%    32%
 40.22  41.00  41.00  41.25  43.00  43.00  43.28  44.00  44.00  45.31  46.64
   33%    34%    35%    36%    37%    38%    39%    40%    41%    42%    43%
 48.33  50.00  50.00  51.00  52.37  53.00  53.39  55.00  56.41  58.00  58.00
   44%    45%    46%    47%    48%    49%    50%    51%    52%    53%    54%
 58.44  59.45  60.46  61.47  63.48  64.00  65.50  66.51  68.00  69.00  70.54
   55%    56%    57%    58%    59%    60%    61%    62%    63%    64%    65%
 71.00  72.00  72.57  73.00  73.59  76.00  77.61  79.24  81.26  83.28  85.65
   66%    67%    68%    69%    70%    71%    72%    73%    74%    75%    76%
 86.66  87.00  88.00  89.69  90.70  91.71  93.00  94.00  97.00  97.00 100.52
   77%    78%    79%    80%    81%    82%    83%    84%    85%    86%    87%
101.77 102.00 105.37 107.00 109.00 109.82 115.00 117.84 122.00 124.00 124.87
   88%    89%    90%    91%    92%    93%    94%    95%    96%    97%    98%
126.88 131.78 138.40 142.82 154.60 163.44 176.94 182.95 188.80 211.37 212.98
   99%   100%
219.94 234.00
```

Come al solito lo script può essere riutilizzato molto facilmente, assegnando (**<-**) all'oggetto **mydata** i propri dati**.**

---

[1] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[2] La Median Absolute Deviation about median o MAD ovvero la deviazione assoluta mediana (dalla mediana) è l'equivalente non parametrico della deviazione standard. Vedere: Rousseeuw PJ, Croux C. *Alternatives to the Median Absolute Deviation*. Journal of the American Statistical Association 88 (424), 1273-1283, 1993. URL consultato il 04/01/2019: https://goo.gl/4Rh53b

[3] Per la documentazione della funzione **seq()** digitare **help(seq)** nella `Console di R`.

sabato 12 gennaio 2019

Istogrammi e kernel density plot

Gli **istogrammi** sono lo strumento tradizionale per la rappresentazione grafica dei dati di una distribuzione univariata.

I **kernel density plot**, cioè i grafici basati sulla stima kernel di densità, possono essere utilizzati in alternativa al tradizionale istogramma, del quale rappresentano l'evoluzione. La stima kernel di densità è il metodo non parametrico impiegato per realizzare i kernel density plot. Invece di raccogliere le osservazioni in barre come negli istogrammi, lo stimatore kernel di densità colloca in corrispondenza di ogni osservazione una piccola ”gobba”, determinata da un fattore di forma (*kernel*), quindi somma tutte le gobbe e genera una curva smussata impiegando un parametro di smussamento detto ampiezza di banda (*bandwidth*).

Qui riporto uno script che vuole giusto introdurre il tema, mentre trovate il necessario approfondimento, che include vari script con le rappresentazioni più frequentemente impiegate nella pratica, nei post:
→ Istogrammi
→ Kernel density plot
→ Kernel density plot sovrapposti
→ Istogrammi affiancati, sovrapposti e contrapposti

Vediamo quindi una breve sintesi della rappresentazione di istogramma e kernel density plot di una variabile numerica continua, impiegando come esempio il set di dati **ais**. Accertatevi di avere installato il pacchetto **DAAG** che contiene il set di dati, o in alternativa procedete come indicato [1].

Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ISTOGRAMMI E KERNEL DENSITY PLOT
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
hist(ais$rcc, xlim=c(3,7), main = "Istogramma semplice", xlab = "Eritrociti in 10^12/L",
ylab = "Frequenza") # traccia un istogramma semplice dei dati
#
hist(ais$rcc, xlim=c(3,7), breaks=30, col="red", main = "Istogramma colorato", xlab =
"Eritrociti in 10^12/L", ylab = "Frequenza") # traccia un istogramma colorato
#
hist(ais$rcc, xlim=c(3,7), freq=FALSE, breaks="FD", col="red", main = "Istogramma con
curva gaussiana", xlab = "Eritrociti in 10^12/L", ylab = "Densità") # traccia un istogramma
colorato
x <- seq(min(ais$rcc), max(ais$rcc), length.out=40) # curva gaussiana teorica valori in
ascisse
y <- dnorm(x, mean=mean(ais$rcc), sd=sd(ais$rcc)) # curva gaussiana teorica valori in
ordinate
lines(x, y, col="blue", lwd=2) # sovrappone all'istogramma la curva gaussiana
#
plot(density(ais$rcc), xlim=c(3,7), main = "Kernel density plot", xlab="Eritrociti in
10^12/L", ylab = "Stima kernel di densità") # traccia il kernel density plot
polygon(density(ais$rcc), col="gray82", border="black") # colora il kernel density plot
#
```

Lo script traccia tre differenti istogrammi e il kernel density plot per la variabile **rcc**, la concentrazione dei globuli rossi del sangue (eritrociti) [2].

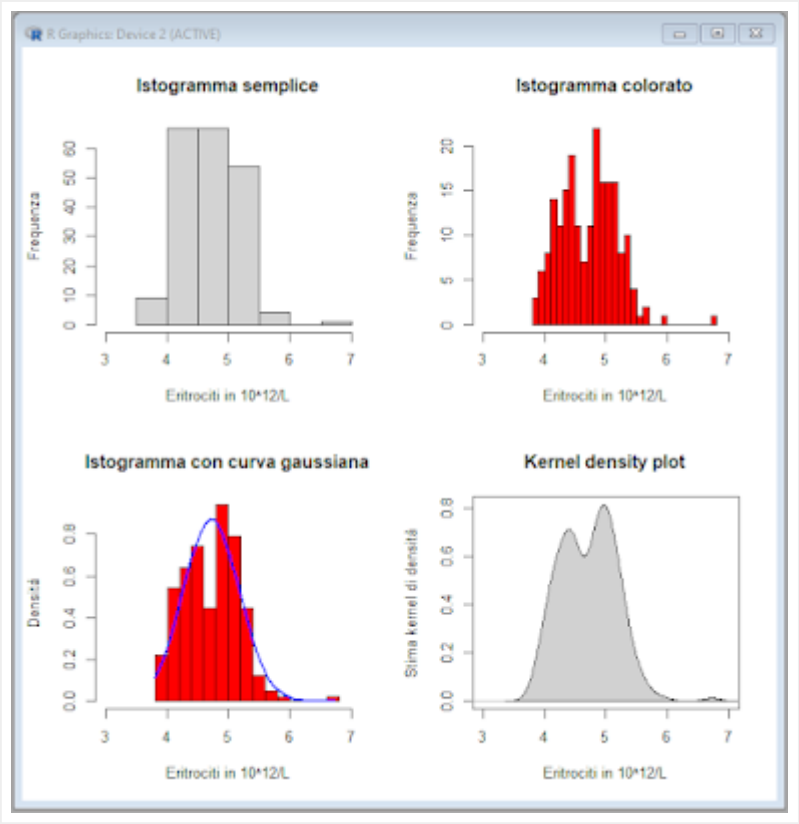
Per favorire il confronto tra le diverse rappresentazioni, la finestra grafica aperta con la funzione **windows()** viene divisa con **par(mfrow=c(2,2))** in quattro quadranti, uno per grafico, che sono riempiti nell'ordine da sinistra verso destra e dall'alto verso il basso. Ma con un semplice adattamento, e cioè eliminando la riga

par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico

e aggiungendo prima di ciascuno dei quattro grafici la riga

windows() # apre e inizializza una nuova finestra grafica

è possibile rappresentare i grafici separatamente.



Il primo dei tre **istogrammi** è il più semplice che si possa realizzare, specifica come argomenti solamente i dati dei quali tracciare l'istogramma (**ais$rcc**), limite inferiore e superiore dell'asse delle x (**xlim=c(3,7)**) per rendere la rappresentazione omogenea con gli altri istogrammi, più il titolo (**main = "..."**) e le etichette dell'asse delle x (**xlab = "..."**) e dell'asse delle y (**ylab = "..."**). Per tutti gli altri argomenti sono impiegati i valori di default della funzione **hist()** [3].

Il secondo istogramma prevede in aggiunta rispetto al primo solamente l'argomento che specifica il numero di classi (**breaks=30**), ovviamente adattabile secondo le esigenze [4], e l'argomento che specifica il colore dell'istogramma (**col="red"**).

Il terzo istogramma viene realizzato impiegando l'argomento **breaks="FD"** con il quale il numero delle classi viene calcolato impiegando la regola di Freedman-Diaconis [5], che consente di minimizzare la differenza tra l'area dell'istogramma e l'area della gaussiana corrispondente. Ponendo in ordinate, invece della frequenza, la densità (di probabilità) con l'argomento **freq=FALSE** è possibile sovrapporre all'istogramma la curva gaussiana teorica corrispondente, avente cioè la media **mean=mean(ais$rcc)** e la deviazione standard **sd=sd(ais$rcc)** della variabile indagata. Da

notare come in questo caso la curva gaussiana mal si adatta alla effettiva distribuzione dei dati, che sembra essere bimodale.

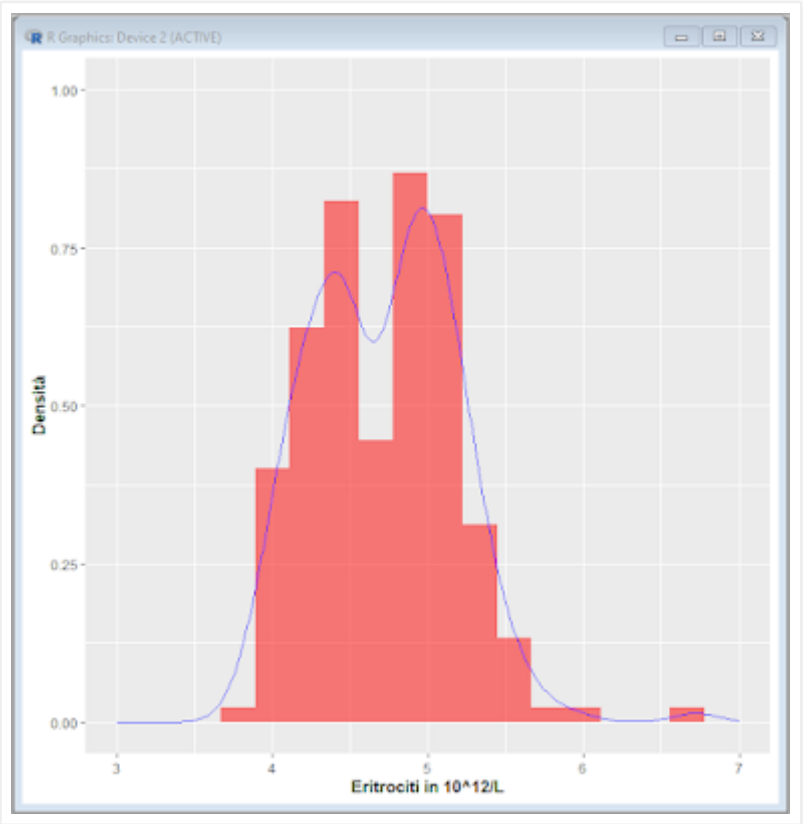
Il quarto grafico è il **kernel density plot**. Per tracciarlo viene impiegata la funzione **plot()** il cui argomento principale non è più costituito come nel caso degli istogrammi dalla variabile **ais$rcc** bensì dalla sua densità kernel calcolata con la funzione **density()**. La successiva funzione **polygon()** colora il kernel density plot in una delle numerose possibili tonalità di colore grigio (**"gray82"**) e ne traccia il profilo in colore nero (**"black"**). Da notare che il kernel density plot, contrariamente all'istogramma semplice in alto a sinistra, rende immediatamente evidente la distribuzione bimodale dei dati.

Vediamo ora come realizzare istogramma e kernel density plot sovrapposti con il pacchetto **ggplot2**, che deve essere scaricato e installato dal **CRAN**).

Copiate lo script, incollatelo nella `Console di R` e premete `↲ Invio`:

```
# ISTOGRAMMA E KERNEL DENSITY PLOT SOVRAPPOSTI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(ggplot2) # carica il pacchetto per la grafica
#
windows() # apre e inizializza una nuova finestra grafica
#
ggplot(data.frame(ais$rcc), aes(x = ais$rcc)) + geom_histogram(aes(y =
after_stat(density)), fill = 'red', alpha = 0.5, bins = 19) + geom_density(colour = 'blue') +
xlab(expression(bold('Eritrociti in 10^12/L'))) + ylab(expression(bold('Densità'))) +
xlim(3,7) + ylim(0,1) # traccia istogramma e kernel density plot sovrapposti
#
```

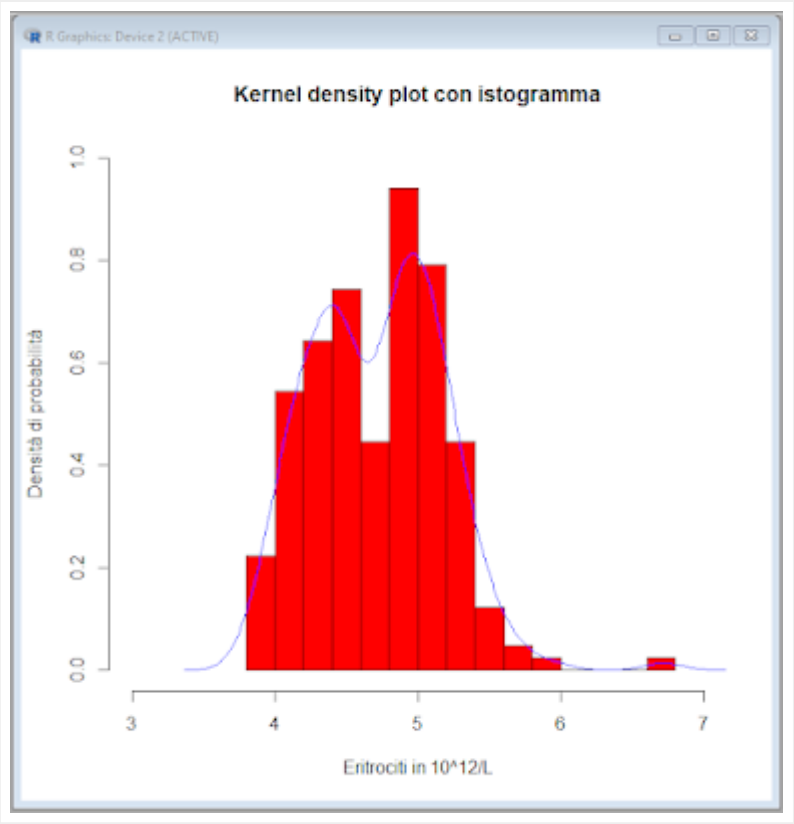
Come vedete nello script il grafico viene realizzato mediante la funzione **ggplot()** [6] concatenando (**+**) più funzioni in una sola riga di codice:



Ma potete anche semplificare il codice impiegando questa variante - basata su due sole funzioni, **hist()** e **lines()** - che non richiede l'installazione di un pacchetto aggiuntivo per la grafica:

```
# KERNEL DENSITY PLOT CON L'ISTOGRAMMA CORRISPONDENTE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
hist(ais$rcc, xlim=c(3,7), ylim=c(0,1), freq=FALSE, breaks="FD", col="red",
main="Kernel density plot con istogramma", xlab="Eritrociti in 10^12/L", ylab="Densità
di probabilità") # traccia l'istogramma
#
lines(density(ais$rcc), xlim=c(3,7), ylim=c(0,1), col="blue", lwd=1) # sovrappone
all'istogramma il kernel density plot
#
```

E questo è il risultato:



L'istogramma e il kernel density plot sono un ottimo esempio di come senza una adeguata rappresentazione grafica potrebbe passare inosservato che nel set di dati ais la concentrazione degli eritrociti nel sangue segue una distribuzione bimodale, dovuta alla coesistenza di due sottoinsiemi di dati aventi differente moda (e media) e correlati al sesso.

Ricordo anche la possibilità di realizzare kernel density plot sovrapposti [6], separati per ciascuno dei fattori impiegati per la classificazione dei casi (che nel set di dati ais sono sesso e sport praticato).

Come al solito il riutilizzo del codice di entrambi gli script è molto semplice in quanto richiede solamente di sostituire alla variabile **ais$rcc** la variabile che volete analizzare, adattando poi opportunamente titolo e legende del grafico.

---

[1] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[2] Concentrazione espressa in migliaia di miliardi di globuli rossi per litro di sangue (10^{12}/L), il che è numericamente identico ad esprimerla in milioni di globuli rossi per microlitro (milionesimo di litro) di sangue (10^{6}/µL), come può essere riportata in alternativa.

[3] Per gli altri argomenti che è possibile impiegare con la funzione **hist()** digitare **help(hist)** nella Console di R. Da notare che a partire da **R** versione **4.0** le barre dell'istogramma sono di default riempite in colore grigio.

[4] In ogni caso il numero delle classi viene poi arrotondato automaticamente da **R**, per la regola utilizzata vedere la funzione **pretty()**.

[5] Vedere *Wikipedia*. Freedman–Diaconis rule. URL consultato il 05/01/2023.

[6] Per i dettagli relativi alle funzioni e agli argomenti impiegati vedere sul **CRAN** il manuale del pacchetto: *Package 'ggplot2'*. URL aggiornato il 14/11/2022: Package ‘ggplot2’

domenica 13 gennaio 2019

Kernel density plot sovrapposti

Abbiamo una distribuzione bimodale quando, all'interno della stessa variabile, coesistono due sottoinsiemi differenti, con differente moda (e media).

In effetti è noto che la concentrazione degli eritrociti (globuli rossi) nel sangue nella donna è mediamente inferiore a quella presente nell'uomo. Impiegando la concentrazione degli eritrociti nel sangue rilevata in 202 atleti australiani riportata nella colonna/variabile `rcc` della tabella **ais** inclusa nel pacchetto **DAAG**, verifichiamo la loro distribuzione impiegando uno script che genera due kernel density plot separati - uno per il sesso/fattore `m` e uno per il sesso/fattore `f` - e vediamo se si sovrappongono. Accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [1].

Lo script richiede inoltre il pacchetto aggiuntivo **sm** che deve essere anch'esso preventivamente scaricato e installato dal **CRAN.**

Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT SOVRAPPOSTI separati per sesso
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
str(ais) # mostra la struttura di ais
#
library(sm) # carica il pacchetto
attach(ais) # funzione per impiegare direttamente i nomi delle variabili di ais
#
# al termine fare click con il tasto sinistro del mouse nel punto in cui si desidera posizionare la legenda
#
windows() # apre una nuova finestra
sm.density.compare(rcc, sex, xlab="Eritrociti in 10^12/L", ylab="Stima kernel di densità") # traccia kernel density plot separati per sesso
title(main="Concentrazione degli eritrociti per sesso") # aggiunge il titolo
sesso <- factor(sex, levels= c("f","m"), labels = c("Donna", "Uomo")) # identifica i casi per sesso
colfill<-c(2:(2+length(levels(sesso)))) # predispone i colori
legend(locator(1), levels(sesso), fill=colfill) # posiziona la legenda
#
```

Dopo avere caricato il pacchetto **DAAG** con il set di dati **ais**, viene mostrata la struttura dei dati:

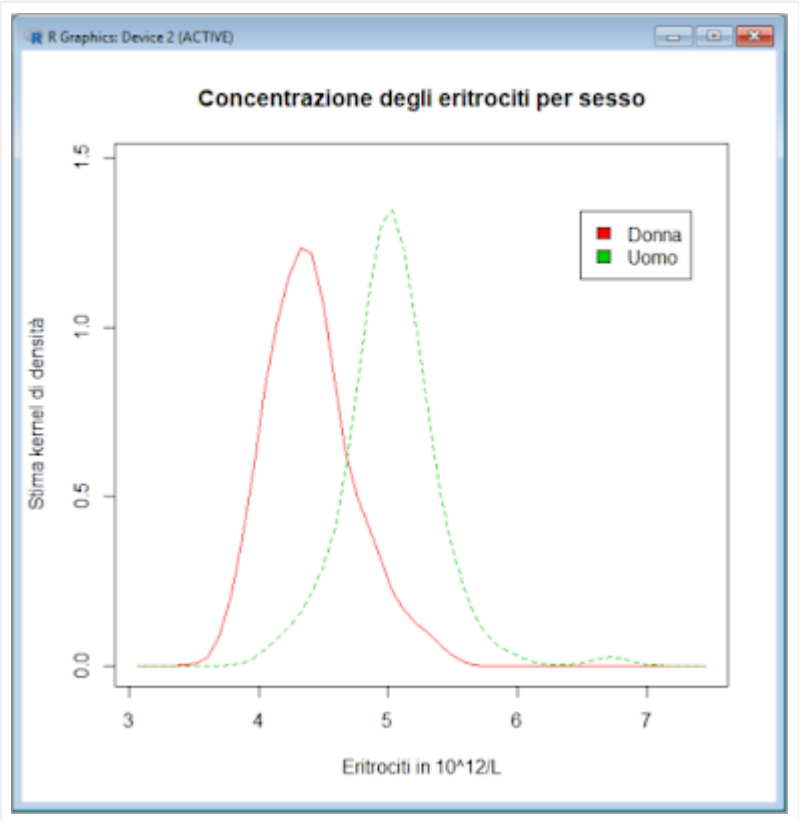
```
> library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
Carico il pacchetto richiesto: lattice
> str(ais) # mostra la struttura di ais
'data.frame':   202 obs. of  13 variables:
 $ rcc   : num  3.96 4.41 4.14 4.11 4.45 4.1 4.31 4.42 4.3 4.51 ...
 $ wcc   : num  7.5 8.3 5 5.3 6.8 4.4 5.3 5.7 8.9 4.4 ...
 $ hc    : num  37.5 38.2 36.4 37.3 41.5 37.4 39.6 39.9 41.1 41.6 ...
 $ hg    : num  12.3 12.7 11.6 12.6 14 12.5 12.8 13.2 13.5 12.7 ...
 $ ferr  : num  60 68 21 69 29 42 73 44 41 44 ...
 $ bmi   : num  20.6 20.7 21.9 21.9 19 ...
 $ ssf   : num  109.1 102.8 104.6 126.4 80.3 ...
 $ pcBfat: num  19.8 21.3 19.9 23.7 17.6 ...
```

```
$ lbm    : num  63.3 58.5 55.4 57.2 53.2 ...
$ ht     : num  196 190 178 185 185 ...
$ wt     : num  78.9 74.4 69.1 74.9 64.6 63.7 75.2 62.3 66.5 62.9 ...
$ sex    : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
$ sport  : Factor w/ 10 levels "B_Ball","Field",..: 1 1 1 1 1 1 1 1 1 1 …
```

La variabile `sex` e la variabile `sport`, le due ultime elencate, sono i **fattori** che possono essere impiegati per la classificazione dei casi. La prima prevede due livelli di classificazione dei casi (`m`, `f`), la seconda ne prevede dieci (`B_Ball`, `Field`, `Gym`, `Netball`, `Row`, `Swim`, `T_400m`, `T_Sprnt`, `Tennis`, `W_Polo`).

Nello script per tracciare i kernel density plot separati per i due sessi viene impiegata la funzione **sm.density.compare()** del pacchetto **sm**, che prevede molto semplicemente come primo argomento la variabile (**rcc**) da analizzare e come secondo argomento il fattore (**sex**) da impiegare per la classificazione dei casi.

Il codice fa una cosa piuttosto interessante: dopo avere tracciato due kernel density plot indipendenti e sovrapposti per i soggetti di sesso maschile (**m**) e di sesso femminile (**f**) rimane in attesa. A questo punto, posizionate il mouse dove volete che compaia la legenda [2] e fate `click` con il tasto sinistro del mouse per farla comparire (e terminare lo script).



Il codice riportato non consente di spostare la legenda. Se non si è soddisfatti della sua posizione, è necessario rieseguire l'intero script e fare nuovamente `click` con il tasto sinistro del mouse nel punto in cui si vuole posizionare la legenda.

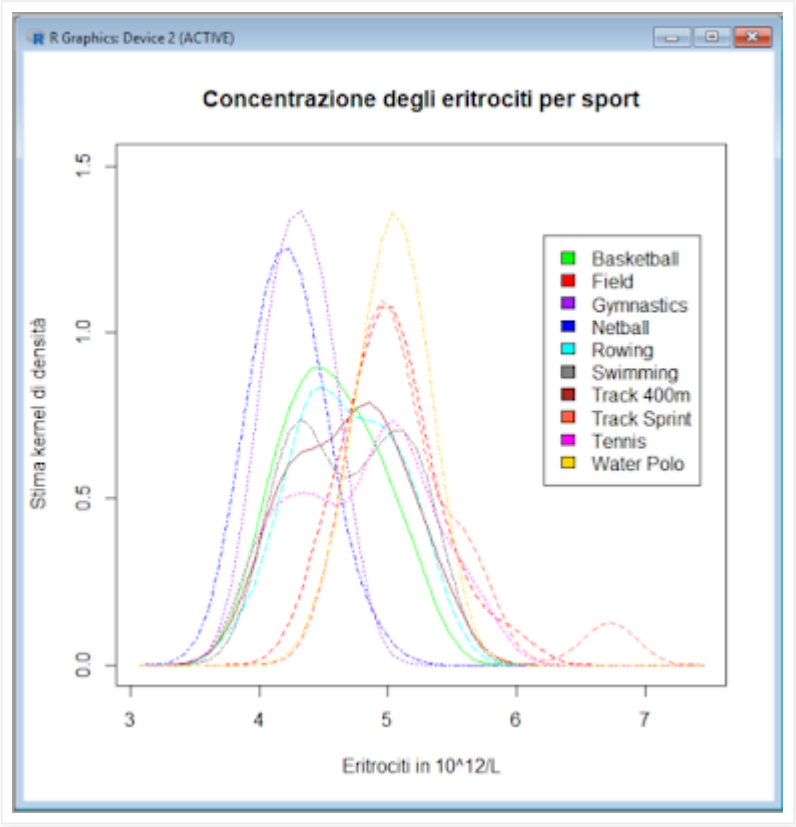
Come risulta evidente dal grafico i kernel density plot separati per sesso mostrano per la concentrazione degli eritrociti nel siero degli atleti australiani la presenza di due sottopopolazioni, una dovuta alle donne, con valori mediamente inferiori, e una dovuta agli uomini, con valori mediamente superiori. Quando i dati sono rappresentati senza essere differenziati per sesso, queste due sottopopolazioni si sommano, determinando la distribuzione bimodale degli eritrociti che si osserva negli istogrammi e nei kernel density plot [3].

Possiamo effettuare un ulteriore approfondimento grafico sugli eritrociti mediante kernel density plot separati per ciascuno degli sport. Copiate quest'altro script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT SOVRAPPOSTI separati per sport
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
#
library(sm) # carica il pacchetto
attach(ais) # funzione per impiegare direttamente i nomi delle variabili di ais
#
# al termine fare click con il tasto sinistro del mouse nel punto in cui si desidera posizionare la legenda
#
windows() # apre una nuova finestra
mycol <- c("green", "red", "purple", "blue", "cyan", "grey50", "brown", "tomato", "magenta", "gold") # predispone i colori per ciascuno sport
sm.density.compare(rcc, sport, col=mycol, xlab="Eritrociti in 10^12/L", ylab="Stima kernel di densità") # traccia kernel density plot separati per sport
title(main="Concentrazione degli eritrociti per sport") # aggiunge il titolo
sprt <- factor(sport, levels=c("B_Ball", "Field", "Gym", "Netball", "Row", "Swim", "T_400m", "T_Sprnt", "Tennis", "W_Polo"), labels=c("Basketball", "Field", "Gymnastics", "Netball", "Rowing", "Swimming", "Track 400m", "Track Sprint", "Tennis", "Water Polo")) # identifica i casi per sport
legend(locator(1), levels(sprt), fill=mycol) # posiziona la legenda
#
```

In questo caso i colori sono scelti e assegnati a ciascuno sport mediante il vettore **mycol** che viene poi impiegato sia nella funzione **sm.density.compare()** sia nella funzione **legend()**.

Di nuovo posizionate il mouse dove volete che compaia la legenda, e fate `click` con il tasto sinistro del mouse per farla comparire (e terminare lo script).



Questo grafico fornisce una prova tangibile delle notevoli rappresentazioni grafiche che possono essere realizzate con **R** quando i dati in ingresso sono adeguatamente raccolti e organizzati.

---

[1] Vedere il post Il set di dati ais. La concentrazione degli eritrociti viene espressa in 10^{12}/L o in 10^{6}/μL, le due unità di misura impiegate nella pratica, che sono numericamente identiche. Nel post trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[2] Più precisamente nel punto in cui viene fatto il click verrà posizionato l'angolo superiore sinistro della legenda.

[3] Potete osservare la distribuzione bimodale nel post Istogrammi e nel post Kernel density plot.

martedì 15 gennaio 2019

Test parametrici e non parametrici per due campioni indipendenti

Attenzione: nel caso di confronti multipli, cioè se dovete effettuare il confronto tra più campioni indipendenti come ad esempio, nel caso più semplice di tre campioni, confrontare il primo campione con il secondo campione, il primo campione con il terzo e il secondo campione con il terzo, dovete impiegare uno dei test riportati nel post Test parametrici e non parametrici per più campioni indipendenti.

Qui ci occupiamo invece del **confronto tra due campioni indipendenti** [1] che viene effettuato verificando se le medie (la tradizionale misura di posizione, parametrica) o le mediane (la misura di posizione non parametrica alternativa alla media) dei due campioni differiscono significativamente tra loro.

Come nel caso delle statistiche elementari anche in questo caso è necessario effettuare una analisi esplorativa dei dati e i test di normalità (gaussianità) [2] per decidere quale sia il test appropriato da impiegare:
→ il tradizionale **test parametrico** per il confronto tra medie, rappresentata dal test t di Student, se i dati sono distribuiti in modo gaussiano;
→ uno dei **test non parametrici** per il confronto tra mediane, che devono essere impiegati quando i dati non sono distribuiti in modo gaussiano, come il test di Wilcoxon per campioni indipendenti (in genere meglio noto come test U di Mann-Whitney) e il test di Kruskal-Wallis.

Lo script è diviso in cinque parti al fine di sviluppare in modo analitico il percorso logico da seguire, che prevede:
→ verifica della distribuzione dei dati mediante i *test di normalità (gaussianità)*;
→ **se** *la distribuzione* **è** *gaussiana* in entrambi i campioni messi a confronto, esecuzione del test F di Fisher per verificare se le varianze dei due campioni sono omogenee;
→ **se** la distribuzione **è** gaussiana **e** con il test di Fisher *le varianze sono omogenee* (non differiscono significativamente) esecuzione dei test t di Student per varianze omogenee;
→ **se** la distribuzione **è** gaussiana **e** con il test di Fisher *le varianze non sono omogenee* (differiscono significativamente) esecuzione dei test t di Student per varianze non omogenee;
→ **se** *la distribuzione* **non è** *gaussiana*, esecuzione dei test non parametrici (test di Wilcoxon per campioni indipendenti e/o test di Kruskall-Wallis).

Vediamo un esempio con i dati rilevati in atleti di sesso femminile e di sesso maschile e la domanda: altezza, peso, percentuale di grasso corporeo, emoglobina e gli altri dati rilevati, differiscono significativamente nei due sessi, o sono simili tra loro? I dati sono contenuti nella tabella **ais** del pacchetto **DAAG**, accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [3].

Copiate questa prima parte dello script, incollatela nella `Console di R` e premete `↲ Invio`:

```
# CONFRONTO TRA DUE CAMPIONI INDIPENDENTI (1/5)
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
# test di normalità
#
library(nortest) # carica il pacchetto
lillie.test(wt[sex == "f"]) # test di Lilliefors (Kolmogorov-Smirnov) su peso corporeo donne
```

lillie.test(wt[sex == "m"]) # idem su peso corporeo uomini
lillie.test(pcBfat[sex == "f"]) # test di Lilliefors (Kolmogorov-Smirnov) su percentuale di grasso corporeo donne
lillie.test(pcBfat[sex == "m"]) # idem su percentuale di grasso corporeo uomini
#

Dopo avere caricato i dati con il pacchetto **DAAG**, avere mostrato la struttura di **ais** e avere consentito con **attach(ais)** di impiegare direttamente nel codice che segue i nomi delle variabili del set di dati **ais**, viene caricato il pacchetto **nortest** che consente di effettuare i test di normalità.

Scegliamo uno dei più classici e collaudati, il test di Lilliefors (Kolmogorov-Smirnov) e lo applichiamo al **peso corporeo** (variabile **wt**) e alla **percentuale di grasso corporeo** (variabile **pcBfat**) di uomini (**m**) e donne (**f**).

Nel caso del peso corporeo **wt** il test di Lilliefors conferma, sia per donne sia per uomini, che i dati non si discostano significativamente da una distribuzione gaussiana (`p = 0.6532` e `p = 0.6384` rispettivamente).

```
> lillie.test(wt[sex == "f"]) # test di Lilliefors (Kolmogorov-Smirnov) su peso corporeo donne

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  wt[sex == "f"]
D = 0.05469, p-value = 0.6532

> lillie.test(wt[sex == "m"]) # idem su peso corporeo uomini

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  wt[sex == "m"]
D = 0.054681, p-value = 0.6384
```

Nel caso della percentuale di grasso corporeo **pcBfat** invece ci troviamo di fronte a dati che se nelle donne non si discostano significativamente da una distribuzione gaussiana (`p = 0.3291`), negli uomini invece non sono distribuiti in modo gaussiano (`p = 2.254e-08`).

```
> lillie.test(pcBfat[sex == "f"]) # test di Lilliefors (Kolmogorov-Smirnov) su percentuale di grasso corporeo donne

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  pcBfat[sex == "f"]
D = 0.066987, p-value = 0.3291

> lillie.test(pcBfat[sex == "m"]) # idem su percentuale di grasso corporeo uomini

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  pcBfat[sex == "m"]
D = 0.17702, p-value = 2.254e-08
```

Per poter confrontare due campioni impiegando un test parametrico, è necessario che i dati siano distribuiti in modo gaussiano in entrambi, pertanto:
→ per il confronto del peso corporeo **wt** in uomini e donne possiamo impiegare un test parametrico, il test t di Student;

→ per il confronto della percentuale di grasso corporeo **pcBfat** in uomini e donne dobbiamo impiegare un test non parametrico (in realtà ne vediamo due, il test di Wilcoxon per campioni indipendenti e il test di Kruskal-Wallis).

Al peso corporeo, distribuito in modo gaussiano, applichiamo innanzitutto mediante la funzione **var.test()** il test F di Fisher per il controllo dell'omogeneità tra le varianze dei due campioni da mettere a confronto, che sono ricavati aggregando i valori della variabile peso corporeo (**wt**) in base al valore della variabile sesso (**sex**) mediante l'argomento **wt~sex**:

```
# test parametrico (confronto tra medie) per il peso corporeo, verifica preliminare (2/5)
#
var.test(wt~sex) # controllo dell'omogeneità delle varianze con il test F di Fisher
#
```

Il test F di Fisher indica che le varianze osservate in uomini e donne non sono significativamente diverse (la probabilità di osservare per caso una differenza tra le medie pari a quella effettivamente osservata è `p = 0.2029`) e anche l'ipotesi alternativa conferma questo:

```
> var.test(wt~sex) # controllo dell'omogeneità delle varianze con il test F di Fisher

        F test to compare two variances

data:  wt by sex
F = 0.77423, num df = 99, denom df = 101, p-value = 0.2029
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.5221795 1.1488587
sample estimates:
ratio of variances
         0.7742343
```

Qualora le varianze dei due campioni fossero state significativamente diverse, sarebbe stato necessario procedere con il test t di Student per varianze non omogenee, impostando nella funzione **t.test()** l'argomento **var.equal=FALSE**.

Nel nostro caso invece procediamo con il test t di Student per varianze omogenee impostando nella funzione **t.test()** l'argomento **var.equal=TRUE** e aggregando di nuovo i valori della variabile peso corporeo (**wt**) in base ai valori della variabile sesso (**sex**) mediante l'argomento **wt~sex**.

```
# test parametrico (confronto tra medie) per il peso corporeo (3/5)
#
t.test(wt~sex, var.equal=TRUE) # test t di Student per varianze omogenee
#
```

Questi sono i risultati:

```
> t.test(wt~sex, var.equal=TRUE) # test t di Student per varianze omogenee

        Two Sample t-test

data:  wt by sex
t = -9.2272, df = 200, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -18.42589 -11.93717
sample estimates:
```

```
mean in group f mean in group m
       67.34200        82.52353
```

La media del peso corporeo nelle donne è `67.342`, negli uomini è `82.52353` e le due medie sono significativamente diverse, in quanto la probabilità di osservare per caso una tale differenza è inferiore a 0.00000000000000022 (`p < 2.2e-16`) [1].

Soggiacente al test t di Student è l'ipotesi che le due medie siano uguali ovvero che la differenza tra le due medie sia zero (ipotesi H_0 o ipotesi nulla). Ma il test eseguito con **R** verifica anche l'ipotesi alternativa, cioè che la differenza tra le due medie non sia uguale a 0 (zero). L'intervallo di confidenza al 95% della differenza tra medie va da `-18.42589` a `-11.93717` e non include lo 0 indicando che la differenza tra le due medie, pari a -15.18153 (`67.342 - 82.52353`) è significativamente diversa da 0 (la differenza non sarebbe significativamente diversa se l'intervallo di confidenza della differenza tra le due medie includesse lo 0). Le due soluzioni si confermano l'una con l'altra e ci consentono di affermare che le donne pesano in media significativamente meno degli uomini.

Alla percentuale di grasso corporeo, che non è distribuita in modo gaussiano, applichiamo i test non parametrici per il confronto tra mediane con le funzioni **wilcox.test()** e **kruskal.test()**, in ciascuna delle quali i due campioni da mettere a confronto sono ottenuti aggregando i valori della variabile percentuale di grasso corporeo (**pcBfat**) in base ai valori della variabile sesso (**sex**) mediante l'argomento **pcBfat~sex**:

```
# test non parametrici (confronto tra mediane) per la percentuale di grasso corporeo (4/5)
#
wilcox.test(pcBfat~sex) # test di Wilcoxon per campioni indipendenti
#
kruskal.test(pcBfat~sex) # test di Kruskal-Wallis
#
median(pcBfat[sex == "f"]) # mediana della percentuale di grasso corporeo per sesso = f
median(pcBfat[sex == "m"]) # mediana della percentuale di grasso corporeo per sesso = m
#
```

Diversamente dalla funzione **t.test()**, che riporta al termine dei calcoli il valore della media dei due campioni, le funzioni **wilcox.test()** e **kruskal.test()** non forniscono la mediana, per cui allo script sono state aggiunte due righe di codice per calcolarla sia nelle donne sia negli uomini.

La mediana della percentuale di grasso corporeo nelle donne è `17.94`, negli uomini è `8.625` ed entrambi i test confermano che le due mediane sono significativamente diverse, la probabilità di osservare per caso una tale differenza è per entrambi i test inferiore a 0.00000000000000022 (`p < 2.2e-16`) [4].

```
> wilcox.test(pcBfat~sex) # test di Wilcoxon per campioni indipendenti

        Wilcoxon rank sum test with continuity correction

data:  pcBfat by sex
W = 9417.5, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0

> kruskal.test(pcBfat~sex) # test di Kruskal-Wallis

        Kruskal-Wallis rank sum test

data:  pcBfat by sex
Kruskal-Wallis chi-squared = 108.03, df = 1, p-value < 2.2e-16
```

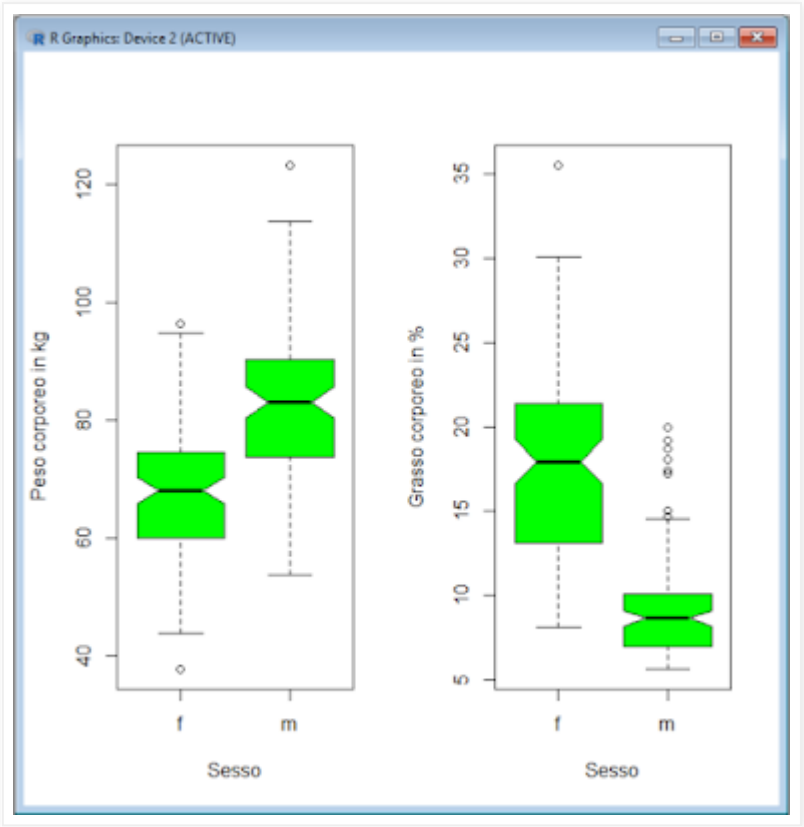
```
> median(pcBfat[sex == "f"]) # mediana della percentuale di grasso corporeo per sesso
= f
[1] 17.94
> median(pcBfat[sex == "m"]) # mediana della percentuale di grasso corporeo per sesso
= m
[1] 8.625
```

Quindi possiamo affermare che nelle donne la mediana della percentuale di grasso corporeo è significativamente maggiore di quella rilevata uomini.

Un interessante integrazione ai risultati numerici ci viene ancora una volta dalla grafica, e in particolare dai grafici a scatola con i baffi:

```
# analisi grafica non parametrica delle distribuzioni (5/5)
#
windows() # apre una nuova finestra
par(mfrow=c(1,2)) # suddivide la finestra in due quadranti, uno per grafico
#
boxplot(wt~sex, data=ais, xlab="Sesso", ylab="Peso corporeo in kg", notch=TRUE,
col="green") # traccia i boxplot per sesso del peso corporeo
#
boxplot(pcBfat~sex, data=ais, xlab="Sesso", ylab="Grasso corporeo in %", notch=TRUE,
col="green") # traccia i boxplot per sesso della percentuale di grasso corporeo
#
```

Questo è il grafico



nel quale i boxplot per sesso del peso corporeo e della percentuale di grasso corporeo sono tracciati con una incisura (**notch=TRUE**) che rappresenta i limiti di confidenza al 95% della mediana. Se i limiti di confidenza delle mediane confrontate non si sovrappongono, le mediane sono significativamente diverse.

Qui il fatto interessante è che andiamo ben oltre l'impiego della grafica come complemento dei risultati dell'analisi numerica: abbiamo un esempio di come sia possibile realizzare un test statistico (non parametrico) mediante una rappresentazione grafica dei dati. La conclusione è che sia per il peso corporeo sia per la percentuale di grasso corporeo le incisure dei boxplot di donne e uomini non si sovrappongono, e pertanto le corrispondenti mediane risultano significativamente diverse in entrambi i casi.

---

[1] Se i due campioni non sono indipendenti impiegare uno dei test riportati nel post Test parametrici e non parametrici per dati appaiati.

[2] Vedere il post Analisi esplorativa dei dati e i post a questo collegati.

[3] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[4] Viene riportato questo valore quando **R** non è più in grado di approssimare numericamente il risultato.

mercoledì 16 gennaio 2019

Test parametrici e non parametrici per dati appaiati

Il confronto tra **dati appaiati** si applica quando la stessa variabile viene misurata nello stesso caso [1] in due occasioni diverse. In campo medico la situazione tipica è quella di una misura che viene effettuata sullo stesso soggetto prima e dopo uno specifico trattamento.

Il Volume Espiratorio Massimo nel primo Secondo (VEMS), impiegato nella diagnostica della capacità respiratoria, è il volume di aria espirata nel corso del primo secondo di una espirazione massima forzata e indica il grado di pervietà delle grandi vie aeree. Viene anche denominato FEV1 dall'acronimo inglese di Forced Expiratory Volume in the 1st second.

I seguenti dati, ricavati da Campbell [2], riportano i valori di VEMS (FEV1) misurati in 5 soggetti asmatici prima (al tempo t0) e dopo (al tempo t1) l'assunzione di un broncodilatatore, e la loro differenza (t0 - t1).

t0	t1	Differenza
1.5	1.7	-0.2
1.7	1.9	-0.2
2.1	2.2	-0.1
1.6	1.9	-0.3
2.4	2.4	0

Per proseguire ora è necessario:
→ effettuare il download del file `FEV1.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati vedere istruzioni e link riportati alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `FEV1.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

t0;t1;Differenza
1.5;1.7;-0.2
1.7;1.9;-0.2
2.1;2.2;-0.1
1.6;1.9;-0.3
2.4;2.4;0

Copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

```
# CONFRONTO TRA DUE CAMPIONI PER DATI APPAIATI
#
mydata <- read.table("c:/Rdati/FEV1.csv", header=TRUE, sep=";") # importa i dati
attach(mydata) # consente di impiegare direttamente i nomi delle variabili
mydata # mostra i dati
#
library(nortest) # carica il pacchetto
lillie.test(Differenza) # test di normalità di Lilliefors (Kolmogorov-Smironv) sulla Differenza (t0 - t1)
#
t.test(t0, t1, paired=TRUE) # test t di Student per dati appaiati (test parametrico)
#
```

wilcox.test(t0, t1, paired=TRUE, exact=FALSE) # test di Wilcoxon per dati appaiati (test non parametrico)
#
detach(mydata) # termina l'impiego diretto dei nomi delle variabili
#

Dopo avere importato i dati del file nell'oggetto **mydata** e dopo avere eseguito la funzione **attach()** che consente nel codice successivo di impiegare direttamente i nomi delle variabili, con **mydata** sono mostrati i dati importati:

```
> mydata # mostra i dati
     t0   t1 Differenza
1   1.5  1.7       -0.2
2   1.7  1.9       -0.2
3   2.1  2.2       -0.1
4   1.6  1.9       -0.3
5   2.4  2.4        0.0
```

La scelta tra il test parametrico, il test t di Student, e il test non parametrico, il test di Wilcoxon (o Wilcoxon Signed Rank Test) viene effettuata come al solito sulla base dei risultati dei test di normalità (gaussianità).

Qui viene impiegato come test di normalità (gaussianità) il test di Lilliefors (Kolmogorov-Smirnov) che viene applicato alla variabile `Differenza` (t0 - t1) presa con il segno:

```
> lillie.test(mydata$Differenza) # test di normalità di Lilliefors (Kolmogorov-
Smironv) sulla Differenza (t0 - t1)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  Differenza
D = 0.23714, p-value = 0.4686
```

Il test conferma una distribuzione gaussiana dei dati (`p = 0.4686`). Si procede quindi a impiegare le conclusioni tratte dal test t di Student

```
> t.test(t0, t1, paired=TRUE) # test t di Student per dati appaiati

        Paired t-test

data:  t0 and t1
t = -3.1379, df = 4, p-value = 0.03492
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.30157148 -0.01842852
sample estimates:
mean of the differences
                  -0.16
```

che con un valore di `p = 0.03492` indica significatività (anche se abbastanza risicata, di poco inferiore a `0.05`) della differenza tra VEMS misurato prima e VEMS misurato dopo la somministrazione del farmaco.

Soggiacente al test t di Student è l'ipotesi che la media delle differenze t0 – t1 sia zero (ipotesi H_0 o ipotesi nulla). Ma il test eseguito con **R** verifica anche l'ipotesi alternativa, cioè che la media delle differenze non sia uguale a 0 (zero).

L'intervallo di confidenza al 95% calcolato per questa ipotesi va da `-0.30157148` a `-0.01842852` e indica che la media delle differenze, uguale a -0.16, è significativamente diversa da 0 (non sarebbe significativamente diversa da 0 se l'intervallo di confidenza della media delle differenze includesse lo 0). Le due soluzioni si confermano l'una con l'altra.

Lo script prevede comunque di eseguire anche il test di Wilcoxon, questo è il risultato:

```
> wilcox.test(t0, t1, paired=TRUE, exact=FALSE) # test di Wilcoxon  per dati appaiati (test non parametrico)

        Wilcoxon signed rank test with continuity correction

data:  t0 and t1
V = 0, p-value = 0.09751
alternative hypothesis: true location shift is not equal to 0
```

Dato il ridotto numero di casi l'ipotesi alternativa non viene sviluppata.

Da notare che assumendo come soglia di significatività il classico `p = 0.05` il test di Wilcoxon con un valore di `p = 0.09751` indica una differenza non significativa. Una cosa che non deve meravigliare, in quanto è noto che le statistiche non parametriche aumentano il valore di `p` ovvero rendono la differenza meno significativa rispetto a quella ottenuta con le statistiche parametriche (che hanno evidenziato una significatività risicata).

Il fatto che approcci statistici differenti possono portare a conclusioni opposte, e il fatto che una significatività statistica (risicata) non implica necessariamente una rilevanza pratica, meritano sempre attenzione nella fase di valutazione critica dei risultati di un'analisi statistica.

Una interessante rappresentazione grafica dei dati appaiati qui riportati può essere effettuata mediante un grafico prima-dopo (slopegraph) al quale si rimanda.

---

[1] Un caso è per definizione un soggetto/oggetto univocamente identificato, come illustrato anche nel post Importazione dei dati da un file .csv.

[2] Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 142.

giovedì 17 gennaio 2019

Grafici a scatola con i baffi (boxplot)

Il modo più semplice per generare i **grafici a scatola con i baffi** (box-and-whiskers plot, denominazione in genere abbreviata in **boxplot**) è impiegare la funzione **boxplot()** inclusa nel pacchetto base di **R** che utilizza i seguenti criteri di rappresentazione [1]:
→ le **scatole** (box) includono il 50% delle osservazioni;
→ il bordo inferiore delle scatole corrisponde al 25° percentile o primo quartile (Q_1);
→ la linea interna alle scatole corrisponde al 50° percentile o secondo quartile (Q_2) ovvero alla mediana;
→ il bordo superiore delle scatole corrisponde al 75° percentile o terzo quartile (Q_3);
→ i **baffi** (whiskers) corrispondono al valore minimo (baffo inferiore) e al valore massimo (baffo superiore) osservati dopo avere escluso gli outliers (vedi sotto);
→ la differenza interquartile [2] viene definita come IQR = $Q_3 - Q_1$ ovvero come differenza tra il valore corrispondente al terzo quartile (Q_3) e il valore corrispondente al primi quartile (Q_1);
→ i valori inferiori a Q_2 - 1.5 · IQR e i valori superiori a Q_2 + 1.5 · IQR sono considerati **outliers** (dati anomali o dati aberranti), sono esclusi dal computo dei valori minimo e massimo (vedi sopra), e sono riportati come punti singoli separati.

I boxplot forniscono una analisi non parametrica dei dati complementare a quella numerica. Qui li impieghiamo per effettuare l'analisi della concentrazione degli eritrociti (globuli rossi) nel sangue rilevata in 202 atleti australiani riportata nella colonna/variabile **rcc** della tabella **ais** inclusa nel pacchetto **DAAG**. Accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [3]. La conclusione?

Con **R** è possibile realizzare grafici a scatola con i baffi con una sola e semplicissima riga di codice!

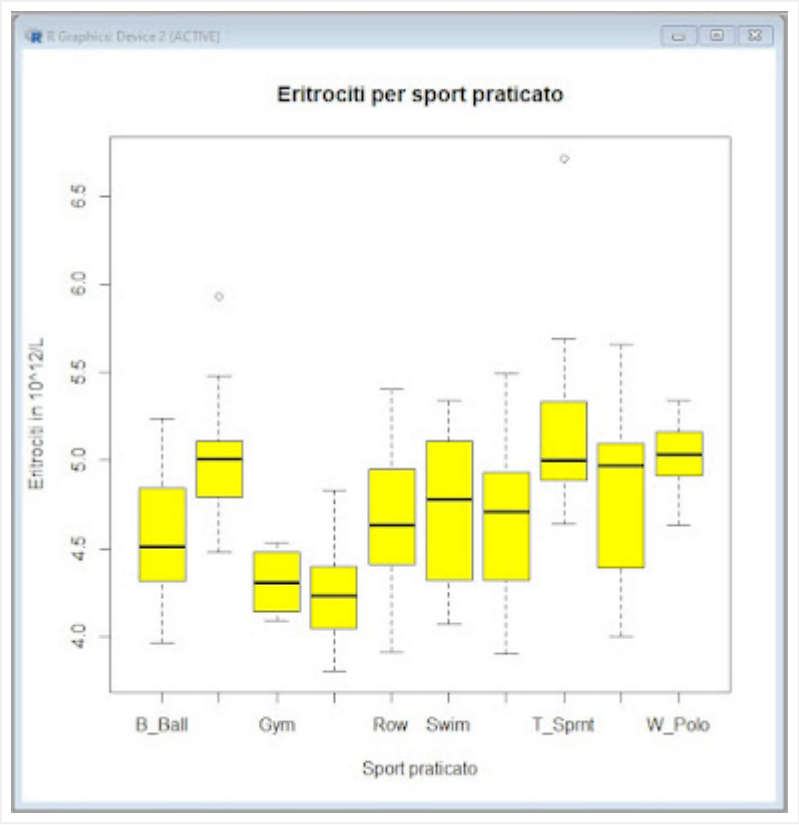
Iniziamo con queste quattro righe di codice, copiatele e incollatele nella `Console di R` e premete ↲ `Invio`:

```
# GRAFICI A SCATOLA CON I BAFFI suddivisi in base a un fattore
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
boxplot(rcc~sport, data=ais, horizontal=FALSE, main="Eritrociti per sport praticato",
xlab="Sport praticato", ylab="Eritrociti in 10^12/L", notch=FALSE, col="yellow") #
eritrociti per ciascuno sport praticato
#
```

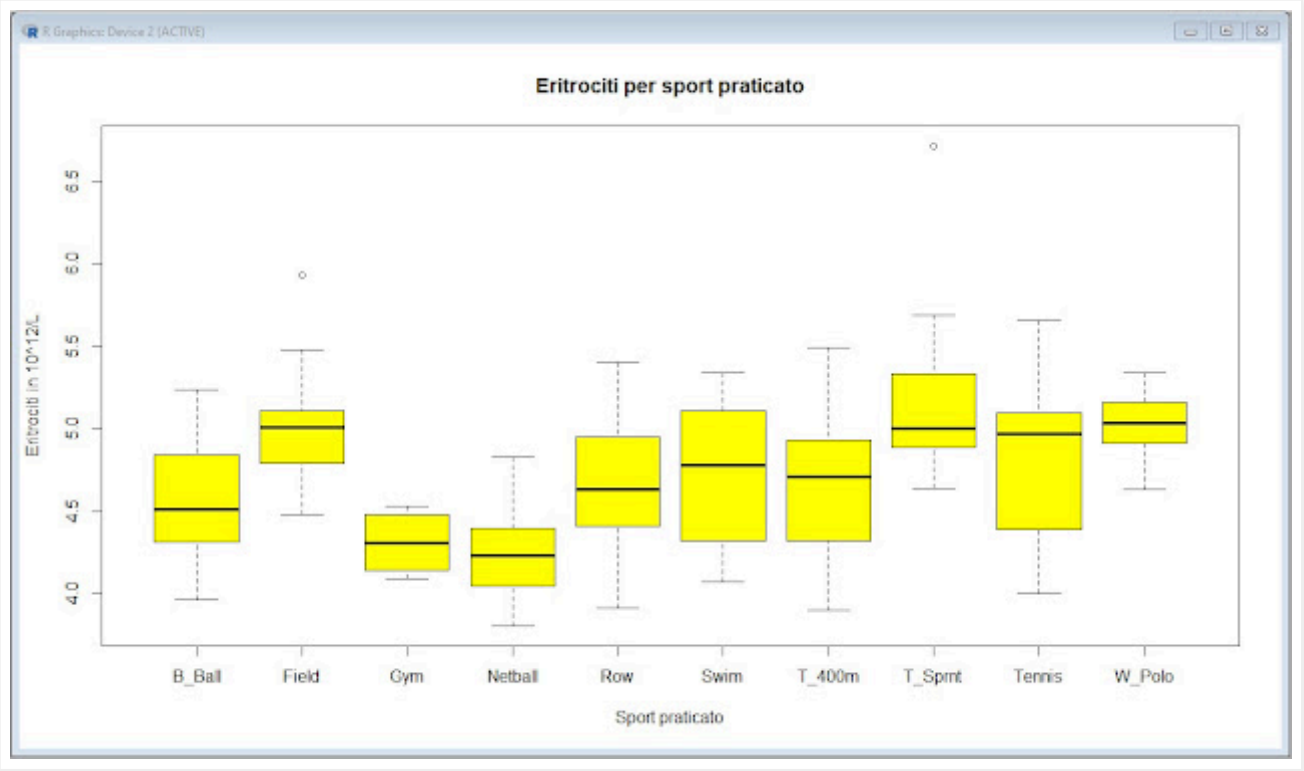
Le prime due righe si limitano a:
→ caricare con la funzione **library()** il pacchetto **DDAG** che include il set di dati **ais**;
→ aprire una finestra grafica con la funzione **windows()**.

Per tracciare il boxplot viene impiegata una sola riga di codice con la funzione **boxplot()** e con questi argomenti:
→ **rcc~sport** che indica che la rappresentazione degli eritrociti sotto forma di boxplot deve essere effettuata aggregando (**~**) i valori della variabile eritrociti (**rcc**) nei sottoinsiemi corrispondenti ai valori del fattore/variabile **sport** (`B_Ball`, `Field`, `Gym`, `Netball`, `Row`, `Swim`, `T_400m`, `T_Sprnt`, `Tennis`, `W_Polo`);
→ **data=ais** che specifica il nome della tabella che contiene la variabile **rcc** e la variabile **sport** - notare che questo argomento è superfluo se vi riferite alle variabili della

tabella **ais** indicandole con il nome completo **ais$***nomedellavariabile*, ovvero nel nostro caso **ais$rcc**, **ais$sex**, **ais$sport**, e così via;
→ **horizontal=FALSE** che indica che i boxplot devono essere orientati verticalmente;
→ **main="..."** che riporta il titolo del grafico;
→ **xlab="..."** che riporta l'etichetta da applicare all'asse `x` delle ascisse;
→ **ylab="..."** che riporta l'etichetta da applicare all'asse `y` delle ordinate:
→ **notch=FALSE** che esclude dai boxplot la rappresentazione dell'incisura che riporta i limiti di confidenza al 95% della mediana dei valori osservati;
→ **col="..."** che definisce il colore di riempimento dei boxplot.



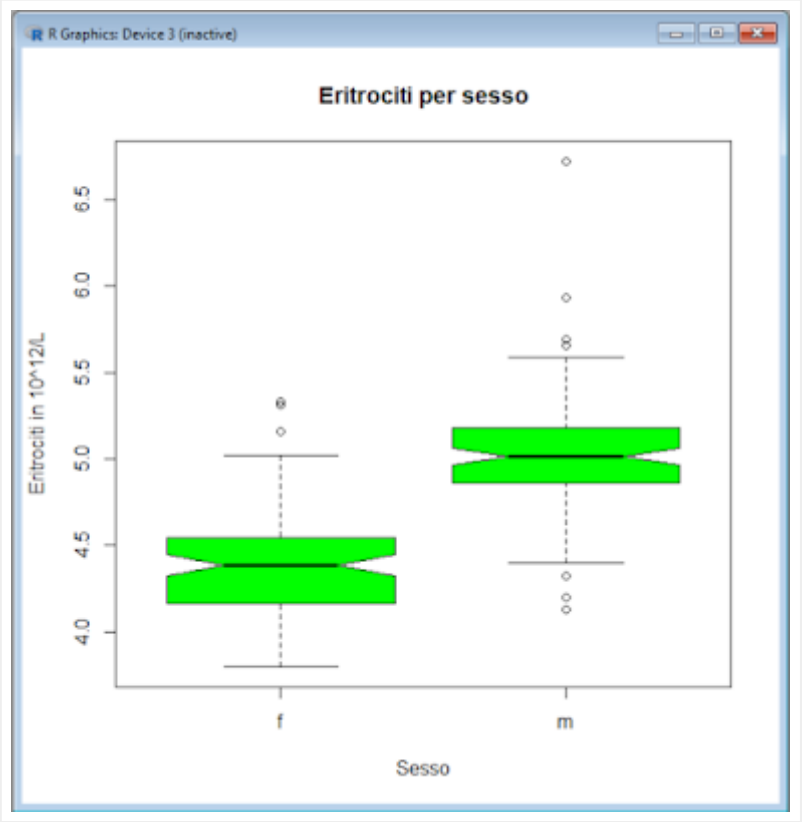
Sull'asse orizzontale apparentemente non c'è spazio sufficiente per riportare le denominazioni di tutti gli sport: ma se "afferrate" il lato sinistro della finestra con il mouse ed estendete la finestra grafica in orizzontale vedete comparire le denominazioni mancanti.



In questo secondo script rielaboriamo i dati aggregando i valori per sesso con l'argomento **rcc~sex**:

```
# GRAFICI A SCATOLA CON I BAFFI con i limiti di confidenza della mediana
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
boxplot(rcc~sex, data=ais, horizontal=FALSE, main="Eritrociti per sesso", xlab="Sesso",
ylab="Eritrociti in 10^12/L", notch=TRUE, col="green") # eritrociti per sesso
#
```

Impiegando l'argomento **notch=TRUE** ora sui lati dei boxplot sono comparse le tacche (**notch**) o se preferite le incisure che rappresentano i limiti di confidenza al 95% della mediana.



Questo corrisponde ad un test per la significatività della differenza tra le mediane: se le tacche o incisure di due boxplot non si sovrappongono, pur sovrapponendosi in parte le distribuzioni dei dati, come in questo caso, la conclusione è che le mediane delle due distribuzioni differiscono significativamente.

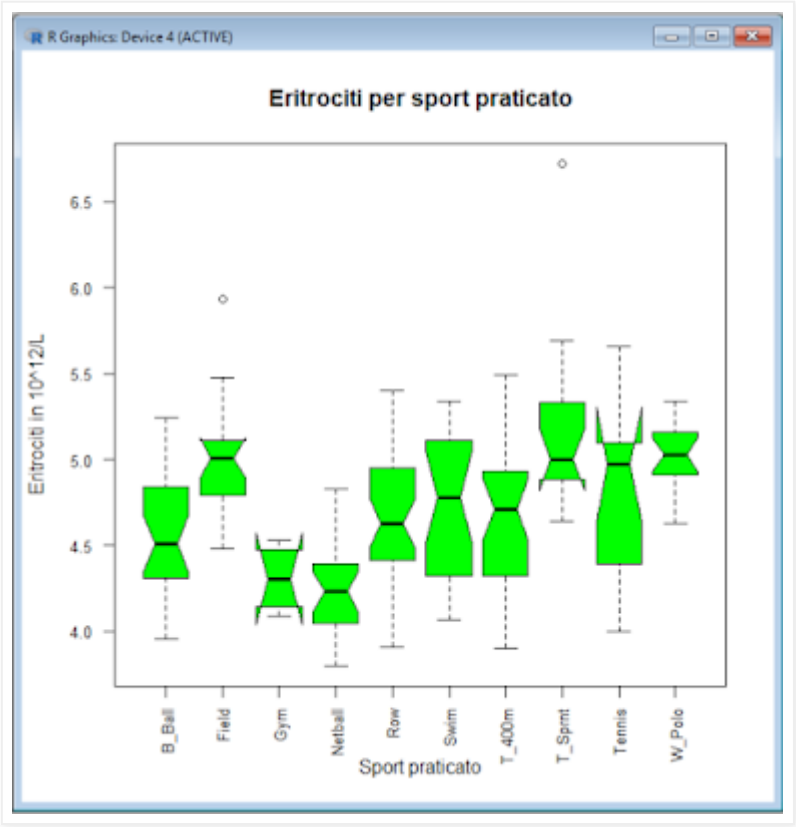
Il fatto interessante è quindi che una rappresentazione grafica può essere impiegata non solo per effettuare una *analisi esplorativa dei dati*, ma addirittura per effettuare un *test statistico* (un confronto tra mediane) che ci conferma il fatto che la mediana della concentrazione degli eritrociti nelle atlete è inferiore a quella dagli atleti (è all'incirca di $4.4 \cdot 10^{12}$/L contro $5.0 \cdot 10^{12}$/L) e che la differenza tra le mediane, all'incirca di $0.6 \cdot 10^{12}$/L, è statisticamente significativa come risulta dal confronto fra i due boxplot.

Da notare che quando nella funzione **boxplot()** si pone l'argomento **notch=TRUE** potrebbe comparire nella `Console di R` un messaggio che avverte che in alcuni casi le incisure sono uscite dai bordi della scatola e che suggerisce di valutare l'opportunità di sostituire l'argomento con **notch=FALSE**.

Questo vi accadrà quando eseguite questo terzo script, che applica l'argomento **notch=TRUE** ai boxplot differenziati per sport:

```
# GRAFICI A SCATOLA CON I BAFFI con i limiti di confidenza della mediana
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
boxplot(rcc~sport, data=ais,  horizontal=FALSE, cex.axis=0.8, las=2, main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in 10^12/L", notch=TRUE, col="green") # eritrociti per ciascuno sport praticato
#
```

In questo caso il grafico



evidenzia la comparsa del problema per gli sport `Field`, `Gym`, `T_sprnt`, `Tennis` e nella `Console di R` viene riportato questo messaggio:

```
> boxplot(rcc~sport, data=ais,  horizontal=FALSE, cex.axis=0.8, las=2,
main="Eritrociti per sport praticato", xlab="Sport praticato", ylab="Eritrociti in
10^12/L", notch=TRUE, col="green") # eritrociti per ciascuno sport praticato
Messaggio di avvertimento:
In (function (z, notch = FALSE, width = NULL, varwidth = FALSE,  :
alcune tacche sono andate fuori dai cardini ('box'): potresti impostare notch=FALSE
> #
```

Il problema è causato dal fatto che per gli sport in questione il numero delle osservazioni è troppo ridotto. In questi casi vi sono solamente due modi per superare il problema:
→ mettere l'argomento **notch=FALSE**, come consigliato nella nota "`potresti impostare notch=FALSE`", e quindi rinunciare al "test grafico" di significatività;
→ aumentare adeguatamente, sempre che sia possibile farlo, il numero delle osservazioni.

Da notare che nella funzione **boxplot()** sono stati aggiunti due argomenti che consentono di ricavare lo spazio necessario per riportare sotto ai boxplot degli eritrociti le denominazioni degli sport senza dover ridimensionare la finestra grafica:
→ l'argomento **cex.axis=0.8** che riduce lievemente la dimensione dei caratteri;
→ l'argomento **las=2** che ruota verticalmente le etichette.

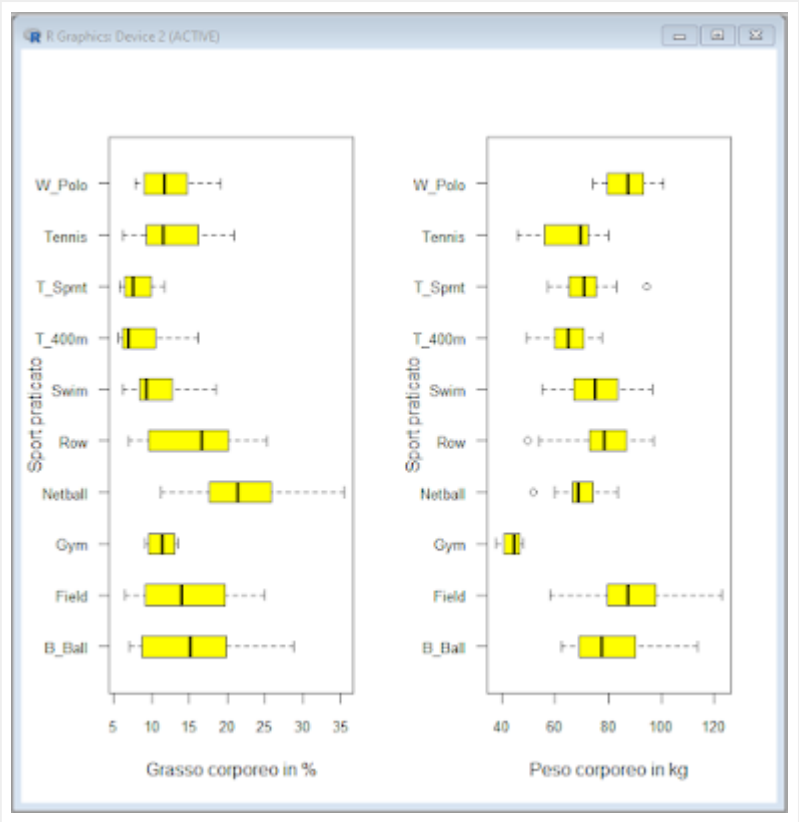
Ora copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GRAFICI A SCATOLA CON I BAFFI con due grafici nella stessa immagine
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(1,2)) # due grafici in una riga e due colonne
#
```

boxplot(ais$pcBfat~ais$sport, horizontal=TRUE, boxwex=0.4, cex.axis=0.8, las=1, xlab="Grasso corporeo in %", ylab="Sport praticato", notch=FALSE, col="yellow") # valori della percentuale di grasso corporeo aggregati per sport
#
boxplot(ais$wt~ais$sport, horizontal=TRUE, boxwex=0.4, cex.axis=0.8, las=1, xlab="Peso corporeo in kg", ylab="Sport praticato", notch=FALSE, col="yellow") # [2] valori del peso corporeo aggregati per sport
#

In questo caso, per riportare due grafici affiancati:
→ con **par(mfrow=c(1,2))** è stata predisposta la suddivisione della finestra in **1** riga e **2** colonne;
→ i boxplot sono riportati in orizzontale (**horizontal=TRUE**);
→ la larghezza dei box è stata ridotta (**boxwex=0.4**);
→ è impiegato l'argomento **cex.axis=0.8** che riduce lievemente la dimensione dei caratteri;
→ le etichette delle ascisse sono riportate in orizzontale (**las=1)**,
→ non vengono riportate le incisure (**notch=FALSE**) che rappresentano i limiti di confidenza al 95% della mediana.

Inoltre qui, in assenza dell'argomento **data=ais**, nella funzione **boxplot()** sono stati riportati per intero – cioè con il prefisso **ais$** che indica la tabella che le contiene – i nomi della variabili.

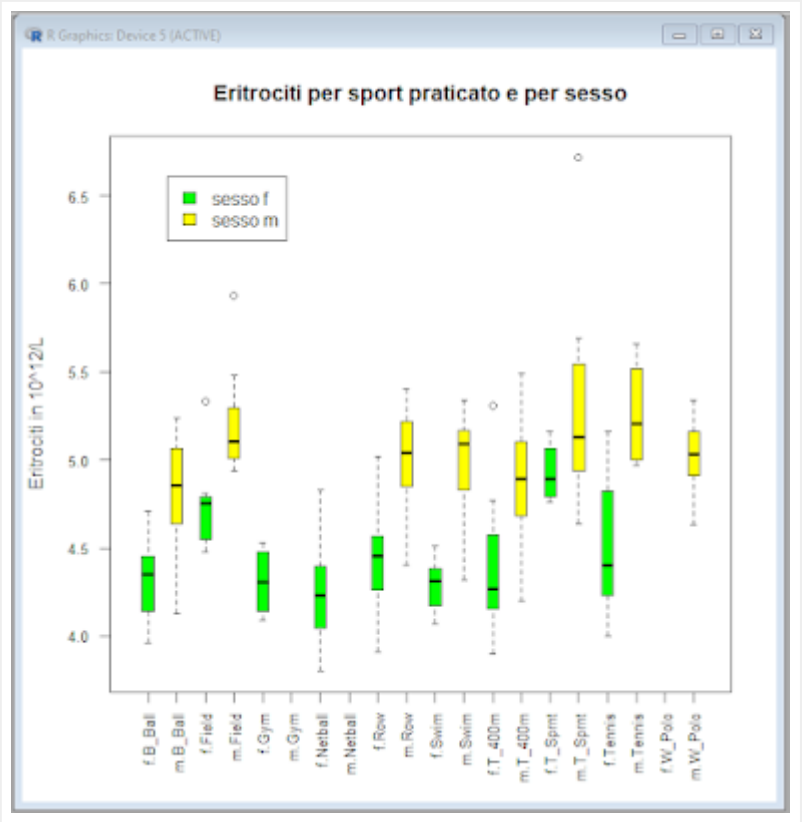


Con quest'ultimo script riportiamo in un unico grafico i boxplot suddivisi in base a due fattori: sport e sesso. Questa volta la novità consiste in una legenda liberamente posizionabile: dopo avere tracciato i grafici separati per sport praticato e per sesso, lo script rimane in attesa. A questo punto bisogna posizionare il mouse dove si vuole che compaia la legenda [4] e fare `click` con il tasto sinistro del mouse per farla comparire e terminare lo script.

GRAFICI A SCATOLA CON I BAFFI suddivisi in base a due fattori
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#

boxplot(rcc~sex+sport, data=ais, horizontal=FALSE, boxwex=0.4, cex.axis=0.8, las=2, main="Eritrociti per sport praticato e per sesso", xlab="", ylab="Eritrociti in 10^12/L", notch=FALSE, col=c("green", "yellow")) # eritrociti per ciascuno sport praticato
#
legend(locator(1), legend=c("sesso f","sesso m"), fill=c("green", "yellow")) # posiziona la legenda
#

Il codice non consente di spostare la legenda. Se non si è soddisfatti della sua posizione, è necessario rieseguire l'intero blocco di codice e fare nuovamente `click` con il tasto sinistro del mouse nel punto in cui si vuole posizionare la legenda.



Nella funzione **boxplot()** sono ora impiegati tre argomenti per risolvere il problema dello spazio necessario per riportare nella finestra grafica i boxplot degli eritrociti per tutti gli sport e per entrambi i sessi:
→ l'argomento **boxwex=0.4** che riduce la larghezza dei boxplot;
→ l'argomento **cex.axis=0.8** che riduce lievemente la dimensione dei caratteri;
→ l'argomento **las=2** che ruota verticalmente le etichette.

La legenda viene realizzata mediante la funzione **legend()** impiegando come argomenti:
→ la funzione **locator()** che resta in attesa, legge la posizione del cursore grafico quando viene fatto click con il tasto sinistro del mouse, e posiziona la legenda;
→ **legend=** che specifica le due righe di testo che compaiono nella legenda;
→ **fill=** che determina la comparsa accanto alle righe di testo di due quadrati dei colori specificati.

Chiudo con tre suggerimenti che potrebbero essere utili:
→ trovate come combinare adeguatamente in un'unica immagine più grafici a scatola con i baffi nel post Inserire più grafici nella stessa immagine;
→ potete combinare i grafici a scatola con i baffi con il kernel density plot dei dati impiegando i grafici a violino (violin plot) [5];
→ quando i dati sono pochi, e preferite conservare il dettaglio dei singoli valori, potete impiegare in alternativa i grafici a punti (dotplot).

---

[1] Digitate **help(boxplot)** nella `Console di R` per la documentazione della funzione **boxplot()**.

[2] In italiano oltre che differenza interquartile si impiega anche scarto o ampiezza o range interquartile, in inglese si impiega Inter Quartile Range, da cui IQR.

[3] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[4] Più precisamente fare `click` con il tasto sinistro del mouse nel punto in cui si vuole posizionare l'angolo superiore sinistro della legenda.

[5] Richiede l'installazione del pacchetto aggiuntivo **ggplot2**.

venerdì 18 gennaio 2019

Grafici prima-dopo (slopegraph)

In un esempio Campbell [1] riporta i dati di VEMS (misura del Volume Espiratorio Massimo nel primo Secondo) effettuate in 5 soggetti asmatici prima (al tempo `t0`) e dopo (al tempo `t1`) l'assunzione di un broncodilatatore, dati riportati in questa tabella:

t0	t1	Differenza
1.5	1.7	-0.2
1.7	1.9	-0.2
2.1	2.2	-0.1
1.6	1.9	-0.3
2.4	2.4	0

In un caso come questo di **dati appaiati** può essere interessante integrare l'analisi statistica con una rappresentazione grafica che, se da un lato è semplicemente la versione minimalista dei grafici a linee, è in grado di fornire un significativo valore aggiunto all'analisi numerica.

La definizione di **grafici prima-dopo** è giustificata dal fatto che nel caso dei dati appaiati il disegno sperimentale tipicamente prevede che sullo stesso caso oggetto di indagine sia effettuata la stessa misura due volte, prima e dopo uno specifico trattamento. In questo modo la differenza tra la prima e la seconda misura è in linea di principio imputabile al solo trattamento deliberatamente previsto dal disegno sperimentale.

Nel pacchetto **CGPfunctions** che impiegheremo, che è necessario scaricare dal **CRAN**, seguendo la concettualizzazione del tema ad opera di Edward Tufte [2] il grafico viene denominato **slopegraph**, ponendo in questo modo l'enfasi sul fatto che la grafica è particolarmente utile a evidenziare la pendenza (*slope*) della linea che unisce i punti, che nel nostro caso prima-dopo sono solamente due, ma che in realtà possono essere molti.

Per effettuare la rappresentazione grafica i dati devono essere strutturati in modo differente da quello della tabella riportata da Campbell sulla pagina di un libro [1], ma devono essere invece strutturati nella forma più generale qui illustrata

Caso	Tempo	Valore
paziente_A	t0	1.5
paziente_A	t1	1.7
paziente_B	t0	1.7
paziente_B	t1	1.9
paziente_C	t0	2.1
paziente_C	t1	2.2
paziente_D	t0	1.6
paziente_D	t1	1.9
paziente_E	t0	2.4
paziente_E	t1	2.4

che corrisponde alla forma nella quale `record` e `campi` sono estratti da un database, dove a ogni dato numerico corrisponde un record, e i record possono essere aggregati tramite un identificatore

univoco [3] qui riportato come variabile `Caso` e rappresentato da uno specifico paziente. Una struttura dati di questo genere segue il concetto di "atomizzazione dell'informazione" e consente non solo di ottenere una generalizzazione della soluzione grafica, ma soprattutto di impiegare direttamente i dati estratti da un database.

Per proseguire ora è necessario:
→ effettuare il download del file `FEV1slopegraph.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il download del file vedere link e istruzioni riportati alla pagina Dati. In alternativa potete copiare le undici righe riportate qui sotto aggiungendo un `↲ Invio` al termine dell'ultima riga e salvarle in `C:\Rdati\` in un file di testo denominato `FEV1slopegraph.csv` (di default i file di testo sono salvati con l'estensione `.txt` ma noi qui salviamo il file con l'estensione `.csv`).

Caso;Tempo;Valore
paziente_A;t0;1.5
paziente_A;t1;1.7
paziente_B;t0;1.7
paziente_B;t1;1.9
paziente_C;t0;2.1
paziente_C;t1;2.2
paziente_D;t0;1.6
paziente_D;t1;1.9
paziente_E;t0;2.4
paziente_E;t1;2.4

Copiate e incollate nella `Console di R` questo script e premete `↲ Invio`:

```
# GRAFICO PRIMA-DOPO (SLOPEGRAPH) dati importati da un file
#
library(CGPfunctions) # carica il pacchetto necessario
mydata <- read.table("c:/Rdati/FEV1slopegraph.csv", header=TRUE, sep=";") # importa i dati
mydata # mostra i dati
windows() # apre una nuova finestra
#
newggslopegraph(dataframe=mydata, Grouping=Caso, Times=Tempo, Measurement=Valore, Title="VEMS prima (t0) e dopo (t1) l'assunzione di un broncodilatatore", SubTitle ="", Caption=NULL, XTextSize=15, YTextSize=4, DataTextSize=4, LineThickness=2) # traccia il graficoprima-dopo (slopegraph)
#
```

Potete anche utilizzare in alternativa al precedente quest'altro script che costruisce manualmente il dataframe, copiatelo e incollatelo nella `Console di R` e premete `↲ Invio`:

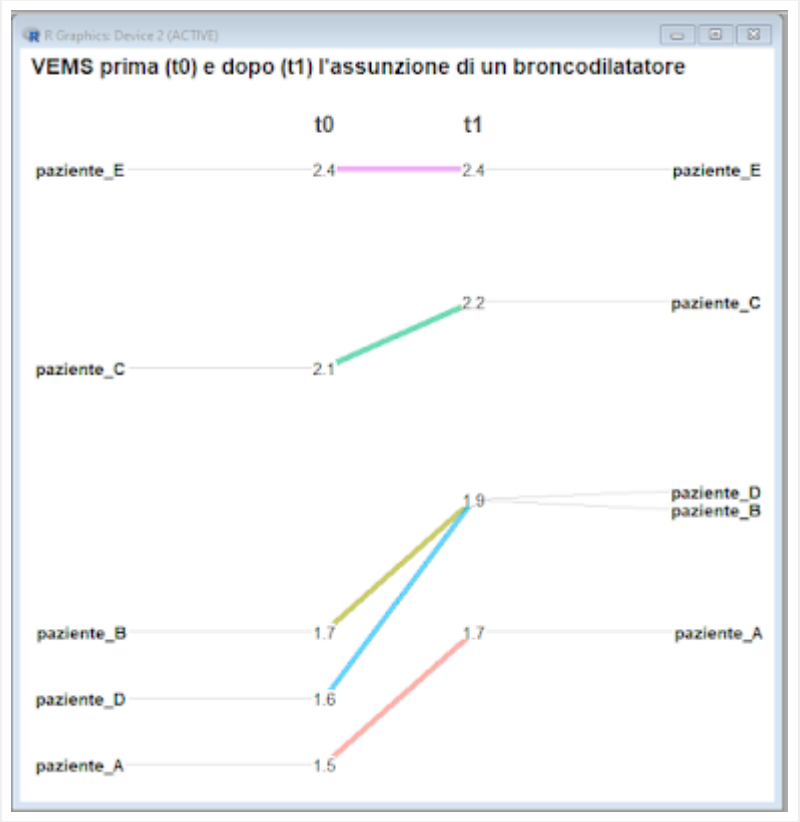
```
# GRAFICO PRIMA-DOPO (SLOPEGRAPH) dataframe costruito manualmente
#
library(CGPfunctions) # carica il pacchetto necessario
Caso <- c("paziente_A", "paziente_A", "paziente_B", "paziente_B", "paziente_C", "paziente_C", "paziente_D", "paziente_D", "paziente_E", "paziente_E") # vettore con i casi
Tempo <- c("t0", "t1", "t0", "t1", "t0", "t1", "t0", "t1", "t0", "t1") # vettore con i tempi
Valore <- c(1.5, 1.7, 1.7, 1.9, 2.1, 2.2, 1.6, 1.9, 2.4, 2.4) # vettore con i valori
mydata <- data.frame(Caso, Tempo, Valore) # genera il dataframe
mydata # mostra i dati
windows() # apre una nuova finestra
```

#
newggslopegraph(dataframe=mydata, Grouping=Caso, Times=Tempo, Measurement=Valore, Title="VEMS prima (t0) e dopo (t1) l'assunzione di un broncodilatatore", SubTitle="", Caption=NULL, XTextSize=15, YTextSize=4, DataTextSize=4, LineThickness=2) # traccia il grafico prima-dopo (slopegraph)
#

Dopo avere importato i dati dal file con il primo script, o avere costruito manualmente il dataframe con il secondo script, e avere mostrato i dati importati

```
> mydata # mostra i dati
         Caso Tempo Valore
1  paziente_A    t0    1.5
2  paziente_A    t1    1.7
3  paziente_B    t0    1.7
4  paziente_B    t1    1.9
5  paziente_C    t0    2.1
6  paziente_C    t1    2.2
7  paziente_D    t0    1.6
8  paziente_D    t1    1.9
9  paziente_E    t0    2.4
10 paziente_E    t1    2.4
```

il grafico è realizzato in entrambi i casi in modo identico mediante la funzione **newggslopegraph()** nella quale gli argomenti sono:
→ **dataframe**, la tabella contenente le tre variabili da elaborare;
→ **Grouping**, la variabile che contiene l'identificativo univoco in base al quale aggregare i dati (il caso);
→ **Times**, la variabile che mette in sequenza, sull'asse orizzontale, i dati di uno stesso caso;
→ **Measurement**, la variabile che contiene i singoli dati di un caso, che vengono riportati in scala sull'asse verticale;
→ **Title**, il titolo del grafico, e un possibile sottotitolo (**SubTitle**), qui lasciato vuoto;
→ **Caption**, una possibile didascalia, qui non impiegata (**= NULL**);
→ **XTextSize**, la dimensione del testo per la variabile **Times**;
→ **YTextSize**, la dimensione del testo per la variabile **Grouping**;
→ **DataTextSize**, la dimensione del testo per la variabile **Measurement**;
→ **LineThickness**, lo spessore della linea che unisce i due valori di ciascun caso.



I colori delle linee sono generati automaticamente dalla funzione **newggslopegraph()**.

Per ulteriori approfondimenti potete digitare **help(newggslopegraph)** nella `Console di R` ovvero, cosa che raccomando sempre, scaricate e consultate il manuale di riferimento del pacchetto [2].

Il grafico è piuttosto interessante in quanto evidenzia una tendenza all'aumento dei valori dopo l'assunzione del farmaco, tendenza che potrà poi essere ulteriormente valutata in termini di significatività statistica impiegando un test per il confronto tra dati appaiati [4].

---

[1] Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 142.

[2] Vedere il manuale di riferimento del pacchetto: *Package 'CGPfunctions'*. URL consultato il 17/01/2019: https://goo.gl/62pmn9

[3] Vedere anche il post Importazione dei dati da un file .csv.

[4] Vedere il post Test parametrici e non parametrici per dati appaiati.

domenica 20 gennaio 2019

Grafici a linee

Vediamo come rappresentare i grafici a linee impiegando una funzione e un set di dati inclusi nella *installazione base* di **R**. Si tratta della funzione **plot()** [1], che viene impiegata per una molteplicità di rappresentazioni, e del set di dati **WorldPhones**.

Se nella `Console di R` digitale **data()** potete vedere l'elenco che include questo e gli altri set di dati che sono caricati automaticamente all'avvio del programma.

Copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

```
# GRAFICI A LINEE set di dati di R
#
mydata <- data.frame(WorldPhones) # trasforma il set di dati WorldPhones in dataframe
mydata # mostra i dati
#
windows() # apre una nuova finestra
#
# traccia il grafico a linee per la prima variabile
plot(mydata$N.Amer, type="o", pch=19, lty=1, col="red", ylim=c(0,80000), axes=FALSE,
ann=FALSE)
# sovrappone i grafici a linee per le variabili successive
lines(mydata$Europe, type="o", pch=20, lty=2, col="blue")
lines(mydata$Asia, type="o", pch=21, lty=3, col="brown")
lines(mydata$S.Amer, type="o", pch=22, lty=4, col="darkolivegreen")
lines(mydata$Oceania, type="o", pch=23, lty=5, col="gold1")
lines(mydata$Africa, type="o", pch=24, lty=6, col="green4")
lines(mydata$Mid.Amer, type="o", pch=25, lty=7, col="cornflowerblue")
#
# traccia gli assi con le rispettive scale e riporta le etichette sull'asse orizzontale
axis(1, at=1:7, labels=c("1951","1956","1957","1958","1959","1960","1961"))
axis(2, las=1, at=20000*0:80000)
#
# riporta titolo ed etichetta dell'asse delle x
title(main="Numero di apparecchi telefonici nel mondo", col.main="black", font.main=1)
title(xlab="Anno della rilevazione", col.lab="black")
#
# riporta una legenda con i nomi delle variabili e i simboli delle linee impiegati
legend(5, 70000, c("Nord America", "Europa", "Asia", "Sud America", "Oceania", "Africa",
"Centro America"), cex=0.8, col=c("red","blue", "brown", "darkolivegreen", "gold1",
"green4", "cornflowerblue"), pch=19:25, lty=1:7, bty="y")
#
```

Nella prima riga di codice dal set di dati **WorldPhones** con la funzione **data.frame()** viene creata una tabella (dataframe) che viene assegnata (**<-**) all'oggetto **mydata**, quindi con **mydata** sono mostrati i dati importati:

```
> mydata # mostra i dati
     N.Amer Europe Asia S.Amer Oceania Africa Mid.Amer
1951  45939  21574 2876   1815    1646     89      555
1956  60423  29990 4708   2568    2366   1411      733
1957  64721  32510 5230   2695    2526   1546      773
1958  68484  35218 6662   2845    2691   1663      836
```

```
1959   71799   37598 6856    3000     2868    1769       911
1960   76036   40341 8220    3145     3054    1905      1008
1961   79831   43173 9053    3338     3224    2005      1076
```

che sono rappresentati dalle rilevazioni, effettuate tra l'anno 1951 e l'anno 1961, del numero di apparecchi telefonici, suddivisi per le principali aree geografiche del mondo.

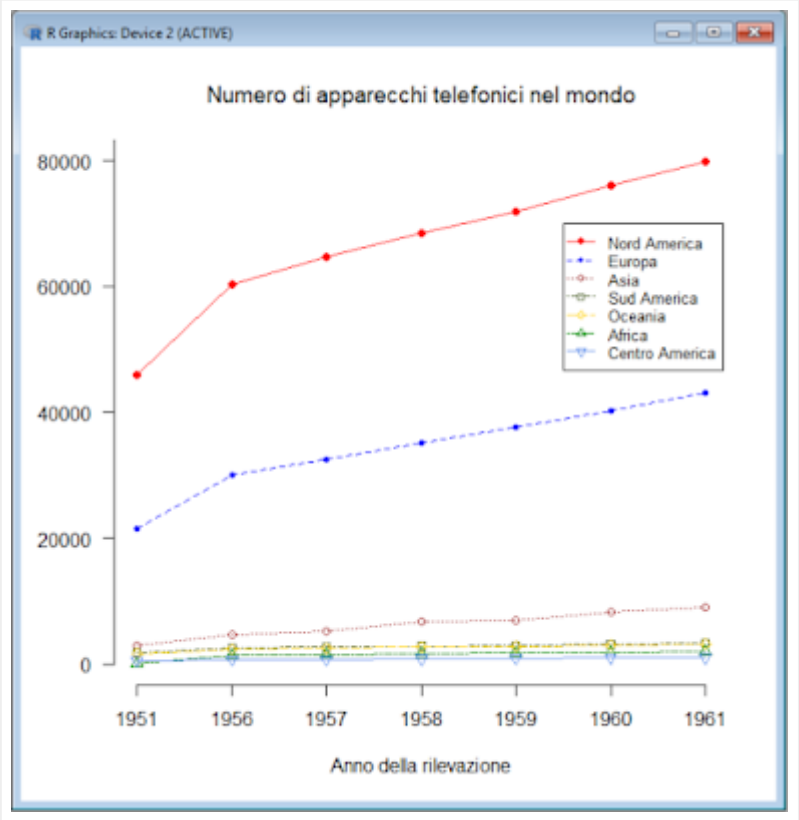
Dopo avere aperto la finestra grafica con **windows()** nel successivo blocco di codice viene generato mediante la funzione **plot()** il grafico a linee che riporta in ascisse (x) i descrittori delle righe (`1951, 1952, … 1961`) e impiega come argomenti:
→ **mydata$N.Amer** la prima variabile del set di dati, posta in ordinate (y);
→ **type="o"** che consente di sovrascrivere i grafici successivi;
→ **pch=19** quale dei simboli dei punti di R deve essere impiegato per i punti da rappresentare;
→ **lty=1** quale degli stili delle linee di R deve essere impiegato per la linea da rappresentare;
→ **col="red"** quale colore impiegare per i punti e le linee da rappresentare;
→ **ylim=c(0,80000)** limiti inferiore e superiore dell'asse delle y;
→ **axes=FALSE** che indica di non rappresentare gli assi del grafico che verranno poi configurati manualmente nelle righe successive;
→ **ann=FALSE** che indica di non rappresentare titolo ed etichette dell'asse delle x e delle y che verranno poi configurati manualmente nelle righe successive.

Al grafico così creato, con le sei righe di codice successive sono sovrapposti i grafici a linee delle altre sei variabili che vogliamo rappresentare: **Europe**, **Asia**, **S.Amer**, **Oceania**, **Africa**, **Mid.Amer**.

Viene poi tracciato l'asse delle x **axis(1….)** ponendo nelle posizioni che vanno dalla 1 alla 7 (**at=1:7**) le etichette **"1951", "1956", "1957", "1958", "1959", "1960", "1961"** riportate nell'argomento **labels**. Per l'asse delle ordinate **axis(2….)** le tacche sono tracciate in orizzontale (**las = 1**) ogni 20000 (**at=20000**) per la scala che va da 0 a 80000 (**0:80000**).

Titolo ed etichetta dell'asse delle x riportati nel blocco di codice successivo non richiedono particolari commenti.

Infine la funzione **legend()** consente di aggiungere una legenda, che prevede i seguenti argomenti:
→ **5, 70000** sono le coordinate x e y alle quali viene posizionato l'angolo superiore sinistro della legenda;
→ **"Nord America", "Europa", "Asia", "Sud America", "Oceania", "Africa", "Centro America"** sono i nomi da riportare;
→ **cex.axis = 0.8** specifica la dimensione dei caratteri da impiegare;
→ **"red","blue", "brown", "darkolivegreen", "gold1", "green4", "cornflowerblue"** sono i colori di R che ovviamente riprendono nell'ordine quelli delle linee riportate nel grafico;
→ **pch=19:25** sono i simboli dei punti di R impiegati nel grafico;
→ **lty=1:7** sono gli stili delle linee di R impiegati nel grafico;
→ **bty="y"** che è l'opzione di default della funzione **plot()** e consente di tracciare il riquadro contenente la legenda, viene qui riportato per ricordare che con **bty="n"** è possibile eliminarlo.



Da notare che l'esempio è stato sviluppato esclusivamente per illustrare le funzioni e gli argomenti per rappresentare un grafico a linee. Ma si fa notare nel set di dati fornito, tra il 1951 e il 1956 intercorrono 5 anni, mentre gli intervalli successivi sono di un anno: questo distorce la rappresentazione, aumentando la pendenza del primo segmento di retta rispetto al reale. Ovviamente si raccomanda di non introdurre mai distorsioni di questo o di altro genere nella rappresentazione dei propri dati.

I grafici a linee vengono spesso impiegati per rappresentare variabili con pochi dati: potrebbe pertanto essere utile disporre di un esempio nel quale i dati sono inseriti manualmente. Copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

```
# GRAFICI A LINEE dataframe costruito manualmente
#
N.Amer <- c(45939, 60423, 64721, 68484, 71799, 76036, 79831) # vettore con la prima variabile
Europe <- c(21574, 29990, 32510, 35218, 37598, 40341, 43173) # vettore con la seconda variabile
Asia <- c(2876, 4708, 5230, 6662, 6856, 8220, 9053) # vettore con la terza variabile
S.Amer <- c(1815, 2568, 2695, 2845, 3000, 3145, 3338) # vettore con la quarta variabile
Oceania <- c(1646, 2366, 2526, 2691, 2868, 3054, 3224) # vettore con la quinta variabile
Africa <- c(89, 1411, 1546, 1663, 1769, 1905, 2005) # vettore con la sesta variabile
Mid.Amer <- c(555, 733, 773, 836, 911, 1008, 1076) # vettore con la settima variabile
mydata <- data.frame(N.Amer, Europe, Asia, S.Amer, Oceania, Africa, Mid.Amer) # combina i vettori nel dataframe mydata
row.names(mydata) <- c(1951, 1956, 1957, 1958, 1959, 1960, 1961) # aggiunge i nomi delle righe
mydata # mostra i dati
#
windows() # apre una nuova finestra
#
# traccia il grafico a linee per la prima variabile
plot(N.Amer, type="o", pch=19, lty=1, col="red", ylim=c(0,80000), axes=FALSE, ann=FALSE)
```

```
# sovrappone i grafici a linee per le variabili successive
lines(Europe, type="o", pch=20, lty=2, col="blue")
lines(Asia, type="o", pch=21, lty=3, col="brown")
lines(S.Amer, type="o", pch=22, lty=4, col="darkolivegreen")
lines(Oceania, type="o", pch=23, lty=5, col="gold1")
lines(Africa, type="o", pch=24, lty=6, col="green4")
lines(Mid.Amer, type="o", pch=25, lty=7, col="cornflowerblue")
#
# traccia gli assi con le rispettive scale e riporta le etichette sull'asse orizzontale
axis(1, at=1:7, labels=c("1951", "1956", "1957", "1958", "1959", "1960", "1961"))
axis(2, las=1, at=20000*0:80000)
#
# riporta titolo ed etichetta dell'asse delle x
title(main="Numero di apparecchi telefonici nel mondo", col.main="black", font.main=1)
title(xlab="Anno della rilevazione", col.lab="black")
#
# riporta una legenda con i nomi delle variabili e i simboli delle linee impiegati
legend(5, 70000, c("Nord America", "Europa", "Asia", "Sud America", "Oceania", "Africa",
"Centro America"), cex=0.8, col=c("red","blue", "brown", "darkolivegreen", "gold1",
"green4", "cornflowerblue"), pch=19:25, lty=1:7, bty="y")
#
```

Dopo avere inserito mediante la funzione **c()** i valori delle singole variabili, questi sono combinati mediante la funzione **data.frame()** nella tabella (dataframe) **mydata**, alla quale sono infine aggiunti con la funzione **row.names()** i nomi delle righe che verranno riportati sull'asse orizzontale del grafico. Il resto del codice è identico a quello dello script precedente.

Da notare come ultima cosa che in quest'ultimo script avendo eseguito la funzione **row.names()** si possono impiegare direttamente i nomi delle variabili (ad esempio **N.Amer**) senza eseguire la funzione **attach()**, mentre nello script precedente, nel quale la funzione **attach()** non è stata eseguita, si è reso necessario specificare per le variabili il nome completo (ad esempio **mydata$N.Amer**).

Entrambi gli script possono essere facilmente riutilizzati, il primo sostituendo al set di dati **WorldPhones** i propri dati, organizzati in modo analogo, il secondo adattando opportunamente numero, nomi e valori delle variabili che confluiscono nell'oggetto **mydata**.

---

[1] Digitate **help(plot)** nella `Console di R` per la documentazione della funzione **plot()** e digitate **help(*nomedellafunzione*)** nella `Console di R` per la documentazione delle altre funzioni impiegate nello script.

mercoledì 23 gennaio 2019

Coefficienti di correlazione parametrici e non parametrici

Come sottolineato altrove, l'assunzione implicita di un rapporto di causa-effetto in seguito ad un coefficiente di correlazione significativo è una delle trappole mentali più ricorrenti e più subdole della statistica: perché non esiste una connessione logica tra correlazione e causazione [1].

Nonostante da questo derivi l'invito ad impiegare la correlazione con parsimonia, con prudenza e con spirito critico nel trarne le conclusioni, risulta difficile ignorare l'esistenza in **R** di due strumenti.

Il primo sono i classici coefficienti di correlazione e precisamente:
→ il **coefficiente di correlazione (lineare) r di Pearson**, il classico test parametrico;
→ il **coefficiente di correlazione per ranghi ρ (rho) di Spearman** (test non parametrico);
→ il **coefficiente di correlazione τ (tau) di Kendall** (test non parametrico).

Il secondo sono i correlogrammi [2], una rappresentazione un po' particolare ma interessante della correlazione tra variabili, utile come complemento grafico ai coefficienti di correlazione di cui sopra.

Va ricordato che il più ampiamente usato (e abbondantemente abusato) **coefficiente di correlazione r** fornisce una misura del grado di **correlazione lineare** ed è pertanto valido solamente se le due variabili a confronto hanno una relazione di tipo lineare, mentre i **coefficienti di correlazione non parametrici** forniscono una misura della correlazione valida anche anche nel caso di **relazioni non lineari**.

In questo primo script viene impiegato il set di dati galton [3], nel quale sono riportate le altezze (in pollici) di 928 coppie padre-figlio. Sono richiesti i pacchetti aggiuntivi **psychTools** e **Hmisc**, che devono essere preventivamente scaricati e installati dal **CRAN**.

Copiate e incollate nella `Console di R` lo script e premete `↵ Invio`:

```
# COEFFICIENTI DI CORRELAZIONE parametrici e non parametrici
#
library(psychTools) # carica il pacchetto che include il set di dati galton
str(galton) # mostra la struttura dei dati
#
# calcola i coefficienti di correlazione con i metodi di pearson (il classico r), spearman, kendall
#
round(cor(galton, use="complete.obs", method="pearson"), digits=2) # r di Pearson
round(cor(galton, use="complete.obs", method="spearman"), digits=2) # rho di Spearman
round(cor(galton, use="complete.obs", method="kendall"), digits=2) # tau di Kendall
#
# calcola i coefficienti di correlazione con i livelli di significatività
#
library(Hmisc) # carica il pacchetto
rcorr(as.matrix(galton), type="pearson") # type può essere solo "pearson" (il classico r) o "spearman"
rcorr(as.matrix(galton), type="spearman")
#
```

Questi sono i risultati ottenuti con i tre metodi per il calcolo della correlazione:

```
> round(cor(galton, use="complete.obs", method="pearson"), digits=2) # r di Pearson
       parent child
parent   1.00  0.46
child    0.46  1.00
```

```
> round(cor(galton, use="complete.obs", method="spearman"), digits=2) # rho di Spearman
       parent child
parent   1.00  0.43
child    0.43  1.00
> round(cor(galton, use="complete.obs", method="kendall"), digits=2) # tau di Kendall
       parent child
parent   1.00  0.33
child    0.33  1.00
```

La funzione **cor()** viene impiegata per calcolare i coefficienti di correlazione con gli argomenti:
→ **galton** che indica i dati da elaborare;
→ **use="complete.obs"** che comporta l'eliminazione automatica dei casi nei quali manchi l'uno o l'altro dato, argomento che qui non servirebbe impiegare, ma che viene riportato per completezza;
→ **method=** che di volta in volta riporta i valori **"pearson"**, **"spearman"**, **"kendall"**, per calcolare le rispettive statistiche.

Da notare che la funzione **cor()** è stata annidata nella funzione **round()** che con l'argomento **digits=2** provvede ad arrotondare a due cifre decimali i risultati: e che i coefficienti di correlazione non parametrici forniscono, come atteso, valori inferiori a quelli del classico coefficiente di correlazione parametrico **r** di Pearson (sono più conservativi).

Infine lo script impiega la funzione **rcorr()** del pacchetto **Hmisc** per calcolare la significatività del coefficiente **r di Pearson** e del coefficiente **ρ (rho) di Spearman**. La probabilità p di osservare per caso i valori di **r** e di **rho** qui ottenuti è molto bassa e, come potrete vedere se eseguite lo script, viene riportata come uguale a 0 (zero). Quindi entrambi questi coefficienti di correlazione risultano essere significativi.

In questo secondo script sono impiegati i dati contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [4].

Copiate e incollate nella `Console di R` lo script e premete `↵ Invio`:

```
# COEFFICIENTI DI CORRELAZIONE parametrici e non parametrici
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
str(ais) # struttura di ais
mydata <- ais[c(1,2,3,4,5,6,7,8,9,10,11)] # salva sole le colonne con le variabili numeriche in mydata
str(mydata) # mostra la struttura di mydata
#
# method può essere pearson (il classico r), spearman, kendall
#
round(cor(mydata, use="complete.obs", method="pearson"), digits=2) # r di Pearson
round(cor(mydata, use="complete.obs", method="spearman"), digits=2) # rho di Spearman
round(cor(mydata, use="complete.obs", method="kendall"), digits=2) # tau di Kendall
#
# calcola i coefficienti di correlazione con i livelli di significatività
#
library(Hmisc) # carica il pacchetto
rcorr(as.matrix(mydata), type="pearson") # type può essere solo "pearson" (il classico r) o "spearman"
rcorr(as.matrix(mydata), type="spearman")
#
```

```
# approfondimenti grafici sulla correlazione lineare
#
library(car) # carica il pacchetto
windows() # apre una finestra grafica
scatterplotMatrix(~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt, regLine =
list(method=lm, lty=1, lwd=2, col="red"), smooth=FALSE,
diagonal=list(method="density", bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE),
col = "black", main="Matrice di dispersione", data=ais) # traccia il grafico di dispersione xy
che incrocia tutte le variabili
#
```

Nella parte iniziale dello script dopo avere caricato il pacchetto **DAAG** con la funzione **str(ais)** viene mostrata la struttura dell'oggetto/set di dati, che contiene 13 variabili, 11 numeriche e 2 non numeriche. Dato che siamo interessati alle sole variabili numeriche, queste, che sono comprese nelle colonne da 1 a 11 della tabella (o dataframe o dataset) **ais**, le salviamo in un nuovo oggetto denominato **mydata** che è quello sul quale lavoreremo. Infine per la conferma dell'avvenuta operazione di selezione delle colonne viene mostrata la struttura del nuovo oggetto con **str(mydata)**.

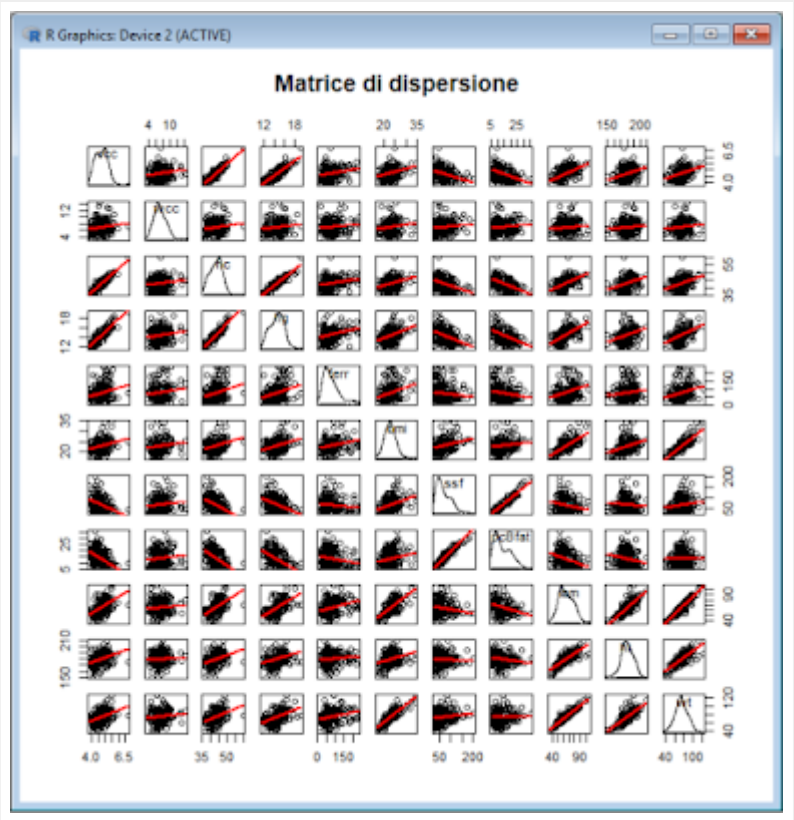
Il resto del codice è praticamente identico allo script precedente. I risultati sono presentati sotto forma di una matrice che contiene i valori dei coefficienti di correlazione che la funzione **cor()** calcola per tutte le coppie di variabili. Questi valori sono simmetrici rispetto alla diagonale, nella quale è riportato sempre un valore uguale a 1 per i coefficienti di correlazione di ciascuna variabile con se stessa. I valori evidenziati (manualmente dal sottoscritto) sono quelli che nel successivo grafico mostrano una relazione tra le variabili che più si avvicina ad una retta.

```
> round(cor(mydata, use="complete.obs", method="pearson"), digits=2) # r di Pearson
         rcc  wcc    hc    hg  ferr  bmi   ssf pcBfat   lbm    ht   wt
rcc     1.00 0.15  0.92  0.89  0.25 0.30 -0.40  -0.49  0.55  0.36 0.40
wcc     0.15 1.00  0.15  0.13  0.13 0.18  0.14   0.11  0.10  0.08 0.16
hc      0.92 0.15  1.00  0.95  0.26 0.32 -0.45  -0.53  0.58  0.37 0.42
hg      0.89 0.13  0.95  1.00  0.31 0.38 -0.44  -0.53  0.61  0.35 0.46
ferr    0.25 0.13  0.26  0.31  1.00 0.30 -0.11  -0.18  0.32  0.12 0.27
bmi     0.30 0.18  0.32  0.38  0.30 1.00  0.32   0.19  0.71  0.34 0.85
ssf    -0.40 0.14 -0.45 -0.44 -0.11 0.32  1.00   0.96 -0.21 -0.07 0.15
pcBfat -0.49 0.11 -0.53 -0.53 -0.18 0.19  0.96   1.00 -0.36 -0.19 0.00
lbm     0.55 0.10  0.58  0.61  0.32 0.71 -0.21  -0.36  1.00  0.80 0.93
ht      0.36 0.08  0.37  0.35  0.12 0.34 -0.07  -0.19  0.80  1.00 0.78
wt      0.40 0.16  0.42  0.46  0.27 0.85  0.15   0.00  0.93  0.78 1.00
> round(cor(mydata, use="complete.obs", method="spearman"), digits=2) # rho di
Spearman
         rcc  wcc    hc    hg  ferr  bmi   ssf pcBfat   lbm    ht    wt
rcc     1.00 0.17  0.91  0.89  0.25 0.29 -0.40  -0.50  0.59  0.40  0.42
wcc     0.17 1.00  0.17  0.14  0.05 0.23  0.15   0.14  0.11  0.04  0.17
hc      0.91 0.17  1.00  0.95  0.25 0.32 -0.44  -0.54  0.63  0.43  0.45
hg      0.89 0.14  0.95  1.00  0.31 0.37 -0.43  -0.54  0.67  0.41  0.49
ferr    0.25 0.05  0.25  0.31  1.00 0.30 -0.11  -0.19  0.34  0.17  0.30
bmi     0.29 0.23  0.32  0.37  0.30 1.00  0.29   0.15  0.70  0.35  0.84
ssf    -0.40 0.15 -0.44 -0.43 -0.11 0.29  1.00   0.96 -0.22 -0.10  0.13
pcBfat -0.50 0.14 -0.54 -0.54 -0.19 0.15  0.96   1.00 -0.41 -0.25 -0.05
lbm     0.59 0.11  0.63  0.67  0.34 0.70 -0.22  -0.41  1.00  0.80  0.91
ht      0.40 0.04  0.43  0.41  0.17 0.35 -0.10  -0.25  0.80  1.00  0.78
wt      0.42 0.17  0.45  0.49  0.30 0.84  0.13  -0.05  0.91  0.78  1.00
> round(cor(mydata, use="complete.obs", method="kendall"), digits=2) # tau di Kendall
         rcc  wcc    hc    hg  ferr  bmi   ssf pcBfat   lbm    ht    wt
rcc     1.00 0.11  0.75  0.71  0.17 0.19 -0.26  -0.33  0.41  0.27  0.28
```

```
wcc      0.11 1.00  0.11  0.09  0.03 0.15  0.10   0.10  0.08  0.03  0.12
hc       0.75 0.11  1.00  0.83  0.17 0.22 -0.29  -0.35  0.44  0.29  0.30
hg       0.71 0.09  0.83  1.00  0.22 0.25 -0.29  -0.36  0.46  0.28  0.33
ferr     0.17 0.03  0.17  0.22  1.00 0.20 -0.07  -0.13  0.22  0.11  0.20
bmi      0.19 0.15  0.22  0.25  0.20 1.00  0.20   0.11  0.52  0.24  0.65
ssf     -0.26 0.10 -0.29 -0.29 -0.07 0.20  1.00   0.82 -0.13 -0.07  0.09
pcBfat -0.33 0.10 -0.35 -0.36 -0.13 0.11  0.82   1.00 -0.24 -0.16 -0.01
lbm      0.41 0.08  0.44  0.46  0.22 0.52 -0.13  -0.24  1.00  0.61  0.77
ht       0.27 0.03  0.29  0.28  0.11 0.24 -0.07  -0.16  0.61  1.00  0.59
wt       0.28 0.12  0.30  0.33  0.20 0.65  0.09  -0.01  0.77  0.59  1.00
```

L'ultima parte dello script realizza un grafico che fornisce la necessaria integrazione grafica all'analisi numerica.

Impiegando la funzione **scatterplotMatrix()** viene realizzato con una sola riga di codice un grafico che riporta sulla diagonale per ciascuna variabile il kernek density plot, e per ciascuna coppia di variabili il grafico di dispersione (xy) con la relativa retta di regressione.



I dettagli delle funzioni qui impiegate, tutte molto semplici, li trovate come al solito digitando **help(***nomedellafunzione***)** nella `Console di R`.

---

[1] Vedere il post Correlazione e causazione.

[2] Vedere il post Correlogrammi.

[3] Vedere il post Il set di dati Galton.

[4] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

giovedì 24 gennaio 2019

Correlogrammi

Oltre al calcolo dei coefficienti di correlazione parametrici e non parametrici, in **R** sono disponibili i **correlogrammi**, che forniscono una sintesi grafica delle informazioni sulla correlazione tra due variabili.

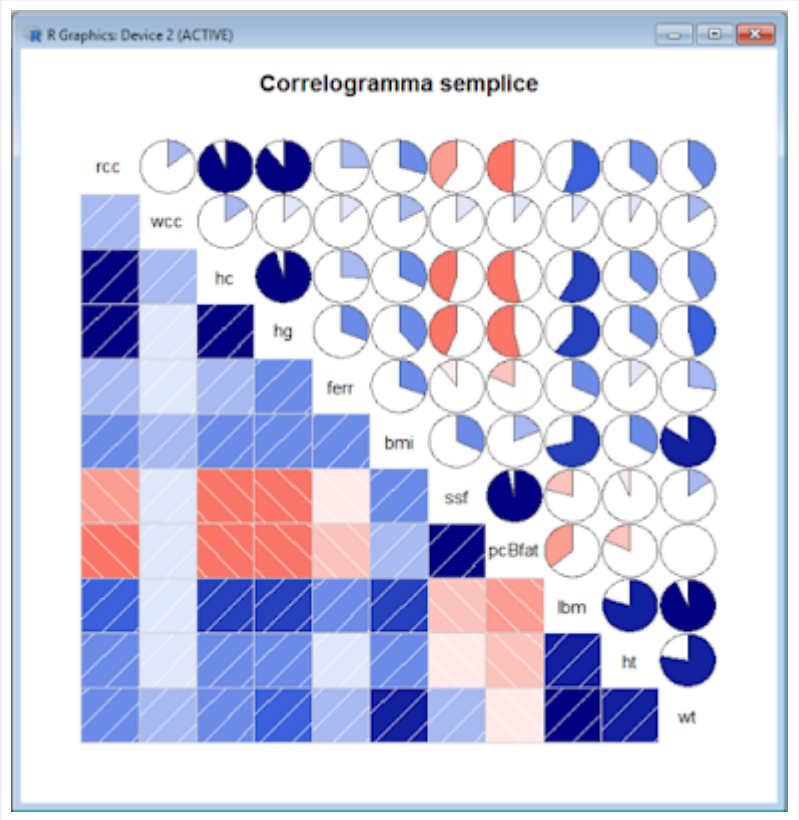
I correlogrammi possono essere generati mediante la funzione **corrgram()** contenuta in uno dei tanti pacchetti aggiuntivi disponibili in **R**, denominato **corrgram**, che deve essere preventivamente scaricato dal **CRAN**. I dati sono contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [1]. In questo script vediamo come con la funzione **corrgram()** sia possibile realizzare tre diversi correlogrammi, che forniscono differenti informazioni sulla correlazione tra le diverse variabili contenute nel set di dati **ais**.

Copiate e incollate nella `Console di R` la prima parte dello script e premete `↵ Invio`:

```
# CORRELOGRAMMI (1/3)
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
str(ais) # mostra la struttura di ais
#
library(corrgram) # carica il pacchetto che genera i correlogrammi
#
windows() # apre una nuova finestra
corrgram(ais, cor.method = "pearson", order=FALSE, lower.panel=panel.shade,
upper.panel=panel.pie, text.panel=panel.txt, main="Correlogramma semplice") #
correlogramma semplice
#
```

I dati sono caricati con **library(DAAG)**, viene mostrata la struttura del set di dati con **str(ais)**, quindi viene caricata il pacchetto necessario per rappresentare i correlogrammi con **library(corrgram)** [2] e viene aperta una nuova finestra grafica con **windows()**.

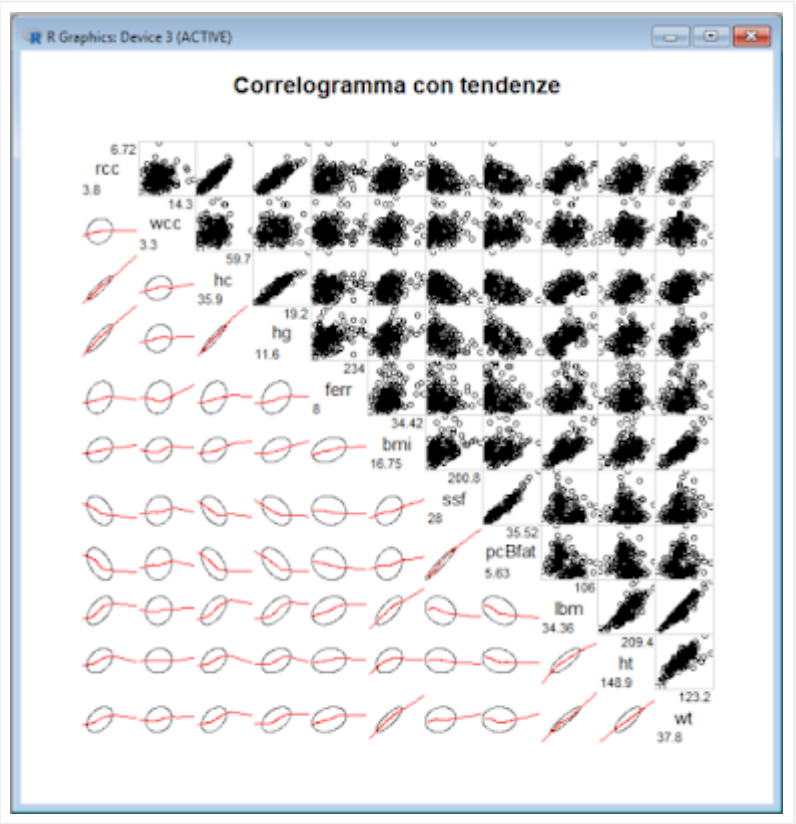
Nella funzione **corrgram()** compaiono come argomenti [2]:
→ il set di dati da analizzare (**ais**);
→ l'ordine nel quale presentare i risultati (**order=FALSE**), che lascia le variabili nello stesso ordine nel quale sono presenti nel set di dati originali. In alternativa l'ordine in cui presentare i risultati può essere espresso come **order="PCA"**, **order="OLO"**, **order="GW"**, **order="HC"** [3];
→ il metodo di correlazione (**cor.method = "pearson"**), corrispondente a uno dei coefficienti di correlazione parametrici e non parametrici, che qui è il classico **coefficiente di correlazione r di Pearson** (test parametrico), ma che può essere in alternativa, impiegando l'argomento **cor.method = "spearman"** o l'argomento **cor.method = "kendall"**, rispettivamente il **coefficiente di correlazione per ranghi ρ (rho) di Spearman** o il **coefficiente di correlazione τ (tau) di Kendall**, entrambi test non parametrici;
→ gli argomenti **lower.panel=panel.shade** e **upper.panel=panel.pie** che consentono di rappresentare i valori **r** di Pearson sotto forma di quantità di colore nelle torte nella metà superiore destra del correlogramma, nelle quali una maggior quantità di colore corrisponde a un più elevato valore di **r**, e di intensità del colore dei quadrati nella metà inferiore sinistra. In blu sono riportate le correlazioni positive in colore rosato quelle negative.



Un diverso correlogramma viene realizzato con la seconda parte dello script ponendo nella funzione **corrgram()** gli argomenti **lower.panel=panel.ellipse** e **upper.panel=panel.pts**. Copiate e incollate nella `Console di R` la seconda parte dello script e premete ↵ `Invio`:

windows() # apre una nuova finestra (2/3)
corrgram(ais, cor.method = "pearson", order=FALSE, lower.panel=panel.ellipse, upper.panel=panel.pts, text.panel=panel.txt, diag.panel=panel.minmax, main="Correlogramma con tendenze") #
correlogramma con evidenza delle tendenze
#

In alto a destra è riportato per ciascuna delle possibili coppie di variabili il grafico xy con la distribuzione dei valori osservati. In basso a sinistra sono mostrati gli elissoidi che indicano la tendenza delle variabili a variare congiuntamente con in rosso la stima della curva che ne approssima la tendenza, elissoidi sempre più stretti e più allungati e curva sempre meno curva man mano che la relazione tra le variabili si avvicina a una retta. Nella diagonale compaiono infine i nomi delle variabili con il valore minimo e il valore massimo osservati.

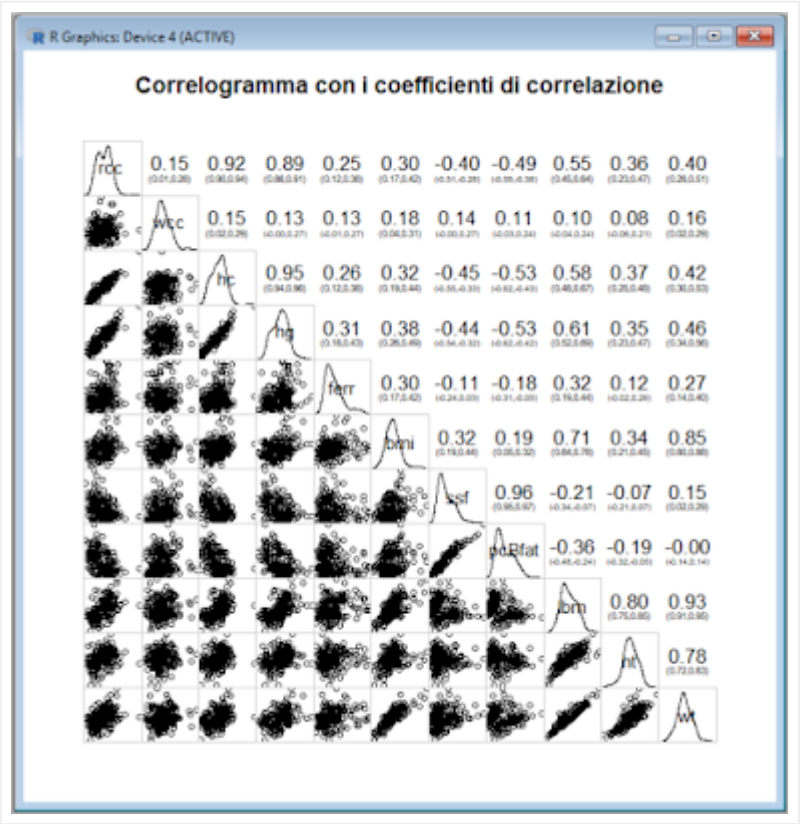


Infine il terzo correlogramma è realizzato con questa ultima parte dello script impiegando nella funzione **corrgram()** l'argomento **lower.panel=panel.pts**, l'argomento **upper.panel=panel.conf** e l'argomento **diag.panel=panel.density**.

Copiate e incollate nella `Console di R` la terza parte dello script e premete `↵ Invio`:

windows() # apre una nuova finestra (3/3)
corrgram(ais, cor.method = "pearson", order=FALSE, lower.panel=panel.pts, upper.panel=panel.conf, diag.panel=panel.density, main="Correlogramma con i coefficienti di correlazione") # correlogramma con i coefficienti di correlazione r di Pearson e i loro limiti di confidenza
#

In alto a destra per ciascuna delle possibili coppie di variabili è riportato il coefficiente di correlazione r di Pearson con i relativi limiti di confidenza al 95%. In basso sinistra è riportato per ciascuna delle possibili coppie di variabili il grafico xy con la distribuzione dei valori osservati. Nella diagonale sono riportati infine i kernel density plot di ciascuna variabile. La funzione **corrgram()** consente ovviamente anche in questo caso di calcolare, in alternativa al coefficiente di correlazione **r** di Pearson, il coefficiente di correlazione per ranghi **ρ** (rho) di Spearman o il coefficiente di correlazione **τ** (tau) di Kendall, impiegando rispettivamente l'argomento **cor.method = "spearman"** o l'argomento **cor.method = "kendall"**.



Da notare che in **R** di default le finestre grafiche si sovrappongono perfettamente l'una all'altra, iconizzatele o spostatele per rivedere quelle aperte in precedenza.

Potete riutilizzare facilmente lo script sostituendo all'oggetto **ais** l'oggetto contenente i vostri dati, opportunamente strutturati. Per una guida rapida all'importazione dei dati potete consultare i link:
→ importare i dati di un file .csv
→ importare i dati di un file .xls o .xlsx
→ gestione dei dati mancanti

---

[1] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[2] Per il pacchetto **corrgram** e la funzione **corrgram()** vedere il manuale di riferimento del pacchetto *Package 'corrgram'*. URL consultato il 23/01/2019: https://goo.gl/F4o92D

[2] *Getting Things in Order: An Introduction to the R Package seriation*. URL consultato il 23/01/2019: https://goo.gl/2LC8Wk

mercoledì 6 febbraio 2019

Regressione lineare semplice parametrica

La **regressione lineare** tout court, quella illustrata in tutti i testi di statistica e qui sviluppata con **R**, è:
→ una regressione lineare semplice (in contrapposizione alla regressione lineare multipla);
→ parametrica (in contrapposizione alla regressione lineare non parametrica);
→ x variabile indipendente (in contrapposizione ad alternative che non prevedono questo assunti).

Dato che la denominazione razionale, quella completa, diverrebbe chilometrica, e dato che l'espressione regressione lineare è però troppo generica, laddove è opportuno semplificare impiegherò la denominazione **regressione lineare ordinaria**, con riferimento al fatto che è quella non solo dovunque illustrata ma anche più ampiamente impiegata.

Gli assunti alla base del modello matematico-statistico implicano una serie di requisiti che devono essere soddisfatti dai dati, che sono ben chiariti ad esempio in Marubini [1] e in Snedecor [2], ma anche online [3]. Per il caso speciale, ma non infrequente, nel quale nessuna delle due variabili confrontate abbia i requisiti richiesti per essere considerata come variabile indipendente, vedere anche [4].

Qualora invece sia necessario l'impiego di "metodi robusti" di regressione - cioè di metodi che risentono poco della eventuale presenza di dati apparentemente anomali ma che non possono essere esclusi a priori dal calcolo della regressione - è possibile impiegare un metodo non parametrico [5].

Tornando alla nostra regressione lineare semplice parametrica `x` variabile indipendente, essendo `x` la variabile indipendente posta sull'asse delle ascisse, `y` la variabile dipendente posta sull'asse delle ordinate, ed essendo soddisfatti i requisiti previsti per i dati, il metodo dei minimi quadrati consente di calcolare l'intercetta a e il coefficiente angolare b della retta di regressione di equazione

$$y = a + b \cdot x$$

che meglio approssima la distribuzione dei dati sperimentali.

Il calcolo dell'equazione della retta di regressione viene effettuato mediante la funzione **lm()**, che può essere applicata anche al caso di più variabili indipendenti, consentendo quindi il calcolo della regressione lineare multipla [4]. Ma la cosa più interessante è che **R**, oltre al calcolo dell'equazione della retta di regressione, un calcolo di per sé semplice, fornisce una serie molto interessante di strumenti che consentono di valutare quanto i dati soddisfano i requisiti richiesti, o in altre parole di valutare se la regressione lineare descrive in modo adeguato la relazione tra le due variabili.

Qui impieghiamo come esempi due diversi set di dati, per uno solo dei quali, come vedremo, la regressione lineare fornisce risultati adeguati.

Il primo è il set di dati **ais**, nel quale prendiamo in considerazione la concentrazione degli eritrociti (espressa in 10^{12}/L) e il valore ematòcrito (espresso in %), due variabili che, analizzate mediante i coefficienti di correlazione parametrici e non parametrici, hanno mostrato di essere correlate, e che, all'ispezione visiva di una serie di grafici di dispersione (xy) realizzati mediante i correlogrammi, hanno mostrato valori ben allineati su una possibile retta.

Per procedere dovete scaricare e installare dal **CRAN** i pacchetti aggiuntivi **gvlma** e **car**, oltre al pacchetto aggiuntivo **DAAG** [6].

Copiate e incollate nella `Console di R` questo script e premete ↵ `Invio`:

```
# REGRESSIONE LINEARE SEMPLICE PARAMETRICA y = a + b · x
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(gvlma) # carica il pacchetto per la funzione gvlma()
library(car) # carica il pacchetto per analisi ouliers e analisi grafica
str(ais) # mostra la struttura dei dati
#
var_x <- ais$rcc # variabile eritrociti x in ascisse
var_y <- ais$hc # variabile ematòcrito y in ordinate
reglin <- lm(var_y ~ var_x) # calcola intercetta (a) e coefficiente angolare (b)
coefficients(reglin) # mostra i coefficienti dell'equazione hc = a + b · rcc
confint(reglin, level=0.95) # calcola gli intervalli di confidenza dell'intercetta e del coefficiente angolare
#
# analisi statistica della adeguatezza della regressione lineare
#
summary(reglin) # mostra un riepilogo dei risultati
t.test(residuals(reglin)) # verifica che la media degli errori non sia significativamente diversa da zero
shapiro.test(residuals(reglin)) # verifica la normalità della distribuzione degli errori
summary(gvlma(reglin)) # test globale per l'assunto di linearità
outlierTest(reglin) # valore p di Bonferonni per la presenza di dati anomali (outliers)
#
# analisi grafica della adeguatezza della regressione lineare
#
windows() # apre una nuova finestra
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
newx = seq(min(var_x), max(var_x), by = 0.01) # valori della x per i quali calcolare l'intervallo di confidenza
conf_interval <- predict(reglin, newdata=data.frame(var_x=newx), interval="confidence", level = 0.95) # calcola gli intervalli di confidenza
plot(var_x, var_y, xlab="Eritrociti (10^12/L)", ylab="Ematòcrito (%)", main="Regressione lineare y = a + b·x") # grafico dei dati
abline(reglin, col="lightblue") # retta di regressione
lines(newx, conf_interval[,2], col="blue", lty=2) # limite di confidenza inferiore
lines(newx, conf_interval[,3], col="blue", lty=2) # limite di confidenza superiore
#
plot(var_y, residuals(reglin), xlab="Ematòcrito (%)", ylab="Ematòcrito osservato - calcolato (%)", main="Analisi delle differenza residue") # grafico delle differenza tra ematòcrito osservato e ematòcrito calcolato con l'equazione della retta
#
influencePlot(reglin, fill=FALSE, xlab="t-quantili (il diametro dei cerchi", sub="è proporzionale alla distanza di Cook)", ylab="Residui studentizzati", main="Influenza dei dati") # grafico dell'influenza dei dati sulle conclusioni
#
qqPlot(reglin, xlab="t-quantili", ylab="Residui studentizzati", main="Quantili vs. residui") # mostra il grafico dei quantili per i residui studentizzati
#
```

Dopo avere caricato i pacchetti aggiuntivi e i dati, e dopo avere mostrato con **str(ais)** la struttura di questi ultimi, con **var_x <- ais$rcc** e con **var_y <- ais$hc** sono memorizzate negli oggetti **var_x** e **var_y** rispettivamente la variabile indipendente x, posta in ascisse, e la variabile dipendente y, posta in ordinate. In questo modo sarà possibile riutilizzare per intero lo script semplicemente sostituendo ad **ais$rcct** e ad **ais$hc** i vettori contenenti i propri dati.

Con **reglin <- lm(var_y ~ var_x)** l'equazione della retta di regressione che esprime la y in funzione della x viene calcolata e memorizzata nell'oggetto **reglin**, che a questo punto diventa l'argomento chiave, l'argomento contenente i dati in ingresso impiegati nelle funzioni successive [7].

Con le funzioni **coefficients()** e **confint()** sono infine mostrati i coefficienti della retta di regressione e i loro intervalli di confidenza al 95%

```
> coefficients(reglin) # mostra i coefficienti dell'equazione hc = a + b · rcc
(Intercept)       var_x
   8.183033    7.398052
> confint(reglin, level=0.95) # calcola gli intervalli di confidenza dell'intercetta
e del coefficiente angolare
               2.5 %    97.5 %
(Intercept) 6.173717 10.192350
var_x       6.974206  7.821898
```

e pertanto questa risulta essere l'equazione della retta di regressione:

$$hc = 8.183033 + 7.398052 \cdot rcc$$

A questo punto seguono due blocchi di codice, il primo per effettuare una analisi statistica, e il secondo per effettuare una analisi grafica della regressione, entrambe allo scopo di valutare, come già detto, se la regressione lineare descrive in modo adeguato la relazione tra le due variabili.

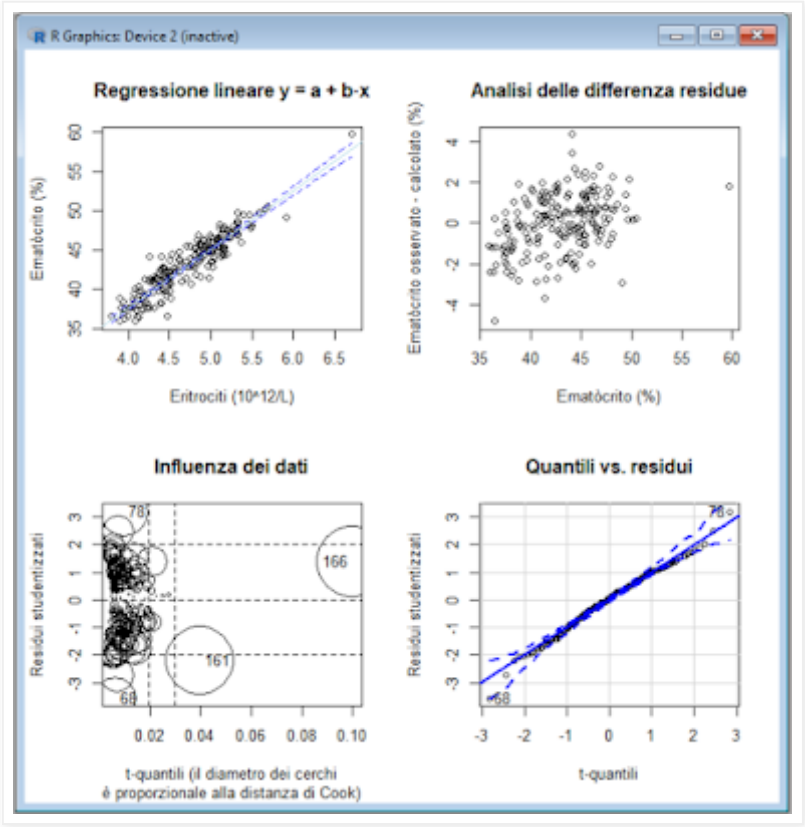
Per quanto concerne l'analisi statistica, le principali conclusioni sono quelle tratte con il **test globale per l'assunto di linearità** effettuato mediante la funzione **gvlma()**, che conferma il fatto che gli assunti che stanno alla base del modello di regressione lineare sono tutti soddisfatti:

```
ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = reglin)

                     Value p-value                Decision
Global Stat        3.479904  0.4809 Assumptions acceptable.
Skewness           2.233166  0.1351 Assumptions acceptable.
Kurtosis           0.958154  0.3277 Assumptions acceptable.
Link Function      0.005239  0.9423 Assumptions acceptable.
Heteroscedasticity 0.283345  0.5945 Assumptions acceptable.
```

Il **test di Bonferroni** non evidenzia dati anomali, ma segnala il dato numero 68 come quello che maggiormente si discosta dai rimanenti:

```
> outlierTest(reglin) # valore p di Bonferonni per la presenza di dati anomali
(outliers)
No Studentized residuals with Bonferonni p < 0.05
Largest |rstudent|:
    rstudent unadjusted p-value Bonferonni p
68 -3.569525         0.00044828     0.090552
```

Le conclusioni dell'analisi grafica confermano, con i valori delle **differenze residue** dispersi in modo casuale, che la regressione lineare descrive in modo adeguato la relazione tra le due variabili



e confermano anche la presenza di possibili dati anomali con il **grafico delle distanze di Cook** e con il **grafico dei quantili**, dati che sono evidenziati sia direttamente nei grafici, sia nei rispettivi riepiloghi riportati nella `Console di R`:

```
> influencePlot(reglin, xlab="t-quantili (il diametro dei cerchi", sub="è proporzionale alla distanza di Cook)", ylab="Residui studentizzati", main="Influenza dei dati") # grafico dell'influenza dei dati sulle conclusioni
      StudRes         Hat      CookD
68  -3.569525 0.006301040 0.03815682
78   3.186317 0.009724284 0.04766681
161 -2.179789 0.039758811 0.09655642
166  1.363972 0.099962743 0.10287127
> #
> qqPlot(reglin, xlab="t-quantili", ylab="Residui studentizzati", main="Quantili vs. residui") # mostra il grafico dei quantili per i residui studentizzati
[1] 68 78
```

I dati per i quali i metodi di identificazione meglio concordano sono il numero 68 e il numero 78, ma anche i dati numero 161 e numero 166 meritano di essere valutati. Si tratta di casi dei quali sarebbe importante controllare la validità, o che potrebbero essere situati in intervalli di valori per i quali potrebbe essere opportuno acquisire più dati.

Vediamo ora la stessa identica analisi applicata al set di dati galton [8]. I pacchetti aggiuntivi **psychTools**, **gvlma** e **car**, se non l'avete già fatto, devono essere preventivamente scaricati e installati dal **CRAN** [9].

Copiate e incollate nella `Console di R` questo script e premete ↵ `Invio`:

```
# REGRESSIONE LINEARE SEMPLICE PARAMETRICA y = a + b · x
#
library(psychTools) # carica il pacchetto che include il set di dati galton
library(gvlma) # carica il pacchetto per la funzione gvlma()
library(car) # carica il pacchetto per analisi outliers e analisi grafica
```

```
str(galton) # mostra la struttura dei dati
#
var_x <- galton$parent # variabile altezza dei padri x in ascisse
var_y <- galton$child # variabile altezza dei figli y in ordinate
reglin <- lm(var_y ~ var_x) # calcola intercetta (a) e coefficiente angolare (b)
coefficients(reglin) # mostra i coefficienti dell'equazione hc = a + b · rcc
confint(reglin, level=0.95) # calcola gli intervalli di confidenza dell'intercetta e del coefficiente angolare
#
# analisi statistica della adeguatezza della regressione lineare
#
summary(reglin) # mostra un riepilogo dei risultati
t.test(residuals(reglin)) # verifica che la media degli errori non sia significativamente diversa da zero
shapiro.test(residuals(reglin)) # verifica la normalità della distribuzione degli errori
summary(gvlma(reglin)) # test globale per l'assunto di linearità
outlierTest(reglin) # valore p di Bonferonni per la presenza di dati anomali (outliers)
#
# analisi grafica della adeguatezza della regressione lineare
#
windows() # apre una nuova finestra
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
newx = seq(min(var_x), max(var_x), by = 0.01) # valori della x per i quali calcolare l'intervallo di confidenza
conf_interval <- predict(reglin, newdata=data.frame(var_x=newx), interval="confidence", level = 0.95) # calcola gli intervalli di confidenza
plot(var_x, var_y, xlab="Altezza dei padri (pollici)", ylab="Altezza dei figli (pollici)", main="Regressione lineare y = a + b·x") # grafico dei dati
abline(reglin, col="lightblue") # retta di regressione
lines(newx, conf_interval[,2], col="blue", lty=2) # limite di confidenza inferiore
lines(newx, conf_interval[,3], col="blue", lty=2) # limite di confidenza superiore
#
plot(var_y, residuals(reglin), xlab="Altezza dei figli (pollici)", ylab="Altezza osservata - calcolata (pollici)", main="Analisi delle differenza residue") # grafico delle differenza tra altezza osservata e altezza calcolata con l'equazione della retta
#
influencePlot(reglin, fill=FALSE, xlab="t-quantili (il diametro dei cerchi", sub="è proporzionale alla distanza di Cook)", ylab="Residui studentizzati", main="Influenza dei dati") # grafico dell'influenza dei dati sulle conclusioni
#
qqPlot(reglin, xlab="t-quantili", ylab="Residui studentizzati", main="Quantili vs. residui") # mostra il grafico dei quantili per i residui studentizzati
#
```

Anche in questo caso, per quanto concerne l'analisi statistica, le principali conclusioni sono quelle tratte con il test globale per l'assunto di linearità effettuato mediante la funzione **gvlma()**, che però questa volta ci dice che gli assunti che stanno alla base del modello di regressione lineare, a parte la curtosi, non sono per niente soddisfatti:

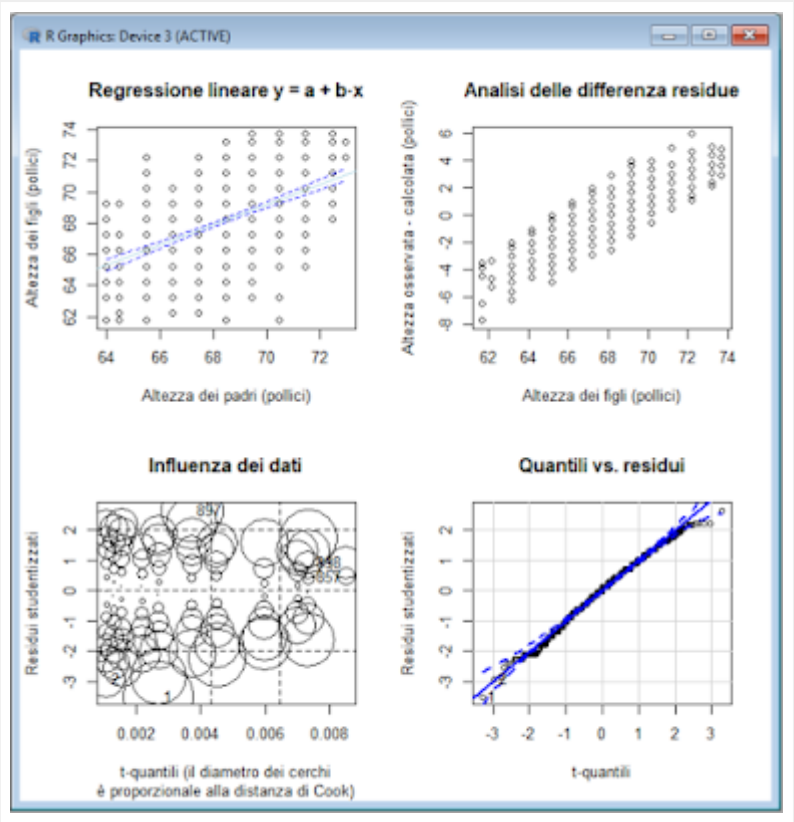
```
ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = reglin)
```

```
                    Value   p-value                   Decision
Global Stat        25.489 4.011e-05 Assumptions NOT satisfied!
Skewness            8.979 2.731e-03 Assumptions NOT satisfied!
Kurtosis            1.965 1.610e-01    Assumptions acceptable.
Link Function       4.632 3.138e-02 Assumptions NOT satisfied!
Heteroscedasticity  9.913 1.641e-03 Assumptions NOT satisfied!
```

Il **test di Bonferroni** non evidenzia dati anomali, ma segnala il dato numero 1 come quello che maggiormente si discosta dai rimanenti:

```
> outlierTest(reglin)  # valore p di Bonferonni per la presenza di dati anomali (outliers)
No Studentized residuals with Bonferonni p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferonni p
1 -3.512671         0.00046509      0.43161
```

Le conclusioni dell'analisi grafica confermano, con i valori delle **differenze residue** che mostrano una proporzionalità diretta con l'altezza, che i dati in questione violano uno degli assunti del modello di regressione lineare



e confermano anche la possibile presenza di dati anomali con il **grafico delle distanze di Cook** e con il **grafico dei quantili**, dati che sono evidenziati sia direttamente nei grafici, sia nei rispettivi riepiloghi riportati nella `Console di R`:

```
> influencePlot(lm(galton$child ~ galton$parent), xlab="t-quantili (il diametro dei cerchi", sub="è proporzionale alla distanza di Cook)", ylab="Residui studentizzati", main="Influenza dei dati") # grafico dell'influenza dei dati sulle conclusioni
       StudRes         Hat       CookD
1   -3.5126706 0.002699826 0.016499436
2   -2.9226403 0.001090010 0.004622768
857  0.4839886 0.008511029 0.001006222
```

```
897  2.6611087 0.003740530 0.013207269
898  0.9327550 0.008511029 0.003734739
> #
>   qqPlot(lm(galton$child   ~   galton$parent),   xlab="t-quantili",   ylab="Residui
studentizzati", main="Quantili vs. residui") # mostra il grafico dei quantili per i
residui studentizzati
[1] 1 2
```

I dati numero 1, 2, 857, 897, 898 forniti dalla funzione **influencePlot()** stanno di nuovo a indicare i casi che influenzano in modo importante la regressione, casi dei quali sarebbe importante controllare la validità, o che potrebbero essere situati in intervalli di valori per i quali potrebbe essere opportuno acquisire più dati. Anche la funzione **qqPlot()** fornisce, oltre al grafico, l'indicazione di due punti da controllare, che sono punti 1, 2 già indicati dalla funzione precedente.

Il set di dati galton, oltre a non soddisfare i requisiti fondamentali che si richiedono ai dati per l'applicazione della regressione lineare ordinaria, è anche un esempio di dati nei quali non è chiaro, a dispetto delle conclusioni tratte da Francis Galton di una "regressione verso la media" delle altezze dei figli rispetto a quelle dei padri, quale delle due variabili debba essere considerata la variabile indipendente. Le implicazioni di questo fatto, le conseguenze che esso determina nelle conclusioni tratte dalla regressione lineare, e il calcolo della regressione lineare con modelli alternativi, sono discussi a parte nel post La regressione lineare: assunti e modelli.

Come potete notare entrambi gli script sono stato realizzati in modo da rendere immediato il loro riutilizzo: è sufficiente assegnare alla variabile x (**var_x <-**) e alla variabile y (**var_y <-**) i vostri nuovi dati (e personalizzare opportunamente titoli e legende).

Per una guida rapida all'importazione dei dati potete consultare i link:
→ importazione dei dati da un file .csv
→ importazione dei dati da un file .xls o .xlsx
→ gestione dei dati mancanti

---

[1] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8. *Il modello statistico nella regressione lineare*, pp-305-308.

[2] Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6. *The matematical model in linear regression*, pp.153-157.

[3] *Regressione lineare*. Da Wikipedia, l'enciclopedia libera. URL consultato il 06/02/2019: https://goo.gl/YDtkZW

[4] Vedere il post La regressione lineare: assunti e modelli.

[5] Vedere il post Regressione lineare semplice non parametrica.

[6] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[7] Digitate **help(lm)** nella `Console di R` per la documentazione della funzione **lm()**.

[8] Vedere il post Il set di dati galton.

[9] Informazioni esaurienti sono contenute nei manuali di riferimento dei pacchetti aggiuntivi qui impiegati, che trovate alla pagina *Available CRAN Packages By Name*. URL consultato il 01/02/20198: https://goo.gl/hLC9BB

giovedì 14 febbraio 2019

Regressione lineare semplice non parametrica

La caratteristica saliente dei metodi di **regressione lineare non parametrica** è costituita dalla loro **robustezza**. Questo significa che si tratta di metodi che risentono poco della eventuale presenza di dati apparentemente anomali, ma che non possono essere esclusi a priori dal calcolo della regressione.

Essendo `n` il numero delle coppie di dati/punti esistono `n · (n – 1) / 2` modi di connettere due punti qualsiasi con una retta, cioè esistono `n · (n – 1) / 2` possibili coefficienti angolari [1]. Nel caso più semplice il coefficiente angolare `b` della retta di regressione è la mediana di questi valori e l'intercetta `a` è la mediana degli `n` valori possibili calcolati mediante il coefficiente angolare trovato. Questi sono concettualmente i principi alla base della regressione lineare non parametrica, con possibili varianti da metodo a metodo.

Qui vediamo, confrontati con la regressione lineare semplice parametrica (regressione lineare ordinaria), tre modelli di regressione lineare non parametrica:
→ la **regressione lineare di Thiel-Sen**;
→ la **regressione lineare di Siegel**;
→ la **regressione quantilica**.

Da notare che questi tre modelli di regressione lineare non parametrica assumono la `x` come variabile indipendente analogamente alla regressione lineare ordinaria.

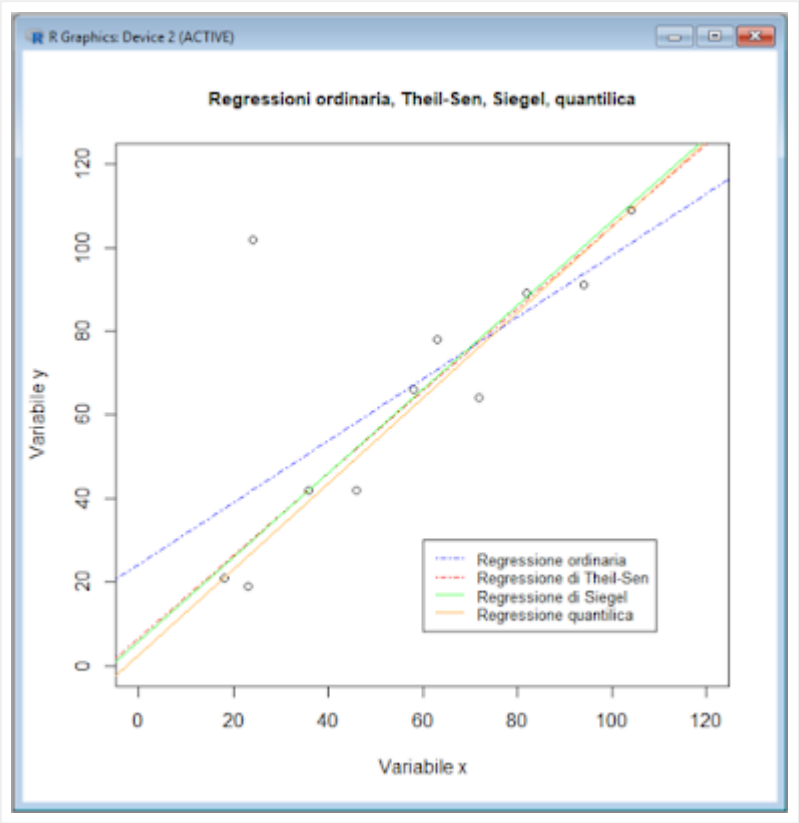
Le funzioni necessarie per il calcolo delle tre regressioni lineari non parametriche sono incluse in due pacchetti aggiuntivi, **mblm** e **quantreg**, che devono essere scaricati dal **CRAN**. Trovate come al solito la documentazione dei pacchetti e in particolare il loro manuale di riferimento sul repository della documentazione [2].

Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# REGRESSIONE LINEARE ORDINARIA E NON PARAMETRICA DI THEIL-SEN, DI SIEGEL, QUANTILICA
#
library(mblm) # carica il pacchetto per le regressioni di Theil-Sen e Siegel
library(quantreg) # carica il pacchetto per la regressione quantilica
#
var_x <- c(18, 46, 104, 72, 24, 63, 23, 58, 36, 82, 94) # variabile in ascisse
var_y <- c(21, 42, 109, 64, 102, 78, 19, 66, 42, 89, 91) # variabile in ordinate
#
regpar_yx <- lm(var_y ~ var_x) # regressione lineare ordinaria x variabile indipendente
a_yx <- regpar_yx$coefficients[1] # intercetta a
b_yx <- regpar_yx$coefficients[2] # coefficiente angolare b
#
reg_TS <- mblm(var_y ~ var_x, repeated=FALSE) # regressione lineare di Theil-Sen
a_TS <- reg_TS$coefficients[1] # intercetta a
b_TS <- reg_TS$coefficients[2] # coefficiente angolare b
#
reg_S = mblm(var_y ~ var_x, repeated=TRUE) # regressione lineare di Siegel
a_S <- reg_S$coefficients[1] # intercetta a
b_S <- reg_S$coefficients[2] # coefficiente angolare b
#
reg_q <- rq(var_y ~ var_x, tau=0.5) # regressione lineare quantilica
a_q <- reg_q$coefficients[1] # intercetta a
b_q <- reg_q$coefficients[2] # coefficiente angolare b
```

```
#
# traccia il grafico dei dati con le rette di regressione e aggiunge una legenda
#
windows() # apre una nuova finestra
plot(var_x, var_y, xlim = c(0,120), ylim = c(0,120), pch=1, xlab="Variabile x",
ylab="Variabile y", main="Regressioni ordinaria, Theil-Sen, Siegel, quantilica", cex.main =
0.9) # grafico dei dati
abline(a_yx, b_yx, col="blue", lty=4) # retta regressione ordinaria
abline(a_TS, b_TS, col="red", lty=4) # retta regressione di Theil-Sen
abline(a_S, b_S, col="green", lty=1) # retta regressione di Siegel
abline(a_q, b_q, col="orange", lty=1) # retta regressione quantilica
legend(60, 30, legend=c("Regressione ordinaria", "Regressione di Theil-Sen",
"Regressione di Siegel", "Regressione quantilica"), col=c("blue", "red", "green",
"orange"), lty=c(4,4,1,1), cex=0.8) # aggiunge al grafico la legenda
#
# crea una tabella con intercetta e coefficiente angolare delle quattro rette di regressione
#
cells <- c(a_yx,b_yx,a_TS,b_TS,a_S,b_S,a_q,b_q) # genera l'array cells con i valori numerici di
a e di b
rnames <- c("Regressione ordinaria", "Regressione di Theil-Sen", "Regressione di Siegel",
"Regressione quantilica") # genera l'array rnames con i nomi delle righe
cnames <- c("Intercetta a", "Coefficiente angolare b") # genera l'array cnames con i nomi
delle colonne
tabris <- matrix(cells, nrow=4, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) #
costruisce la tabella con i risultati a partire dagli array cells, rnames, cnames
tabris # mostra la tabella con i risultati
#
```

Questo è il grafico di dispersione (xy) dei dati, con le rette calcolate:



I risultati delle quattro equazioni

$$y = a + b \cdot x$$

sono così riportati nella `Console di R`:

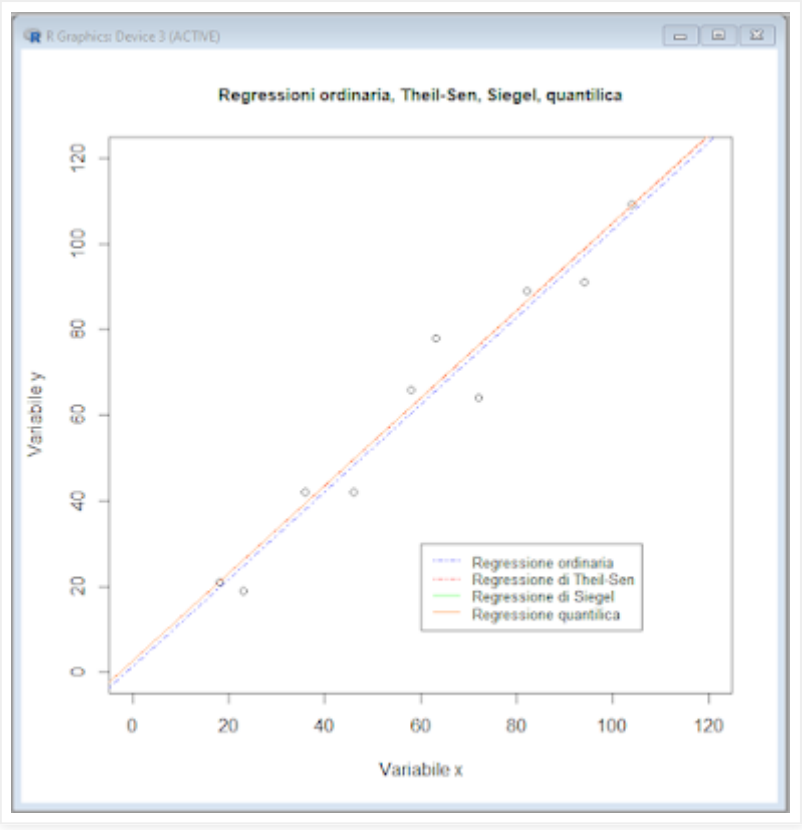
```
> tabris # mostra la tabella con i risultati
                        Intercetta a Coefficiente angolare b
Regressione ordinaria       24.078677               0.7389267
Regressione di Theil-Sen     6.529412               0.9852941
Regressione di Siegel        5.873402               1.0042750
Regressione quantilica       2.581395               1.0232558
```

Le cose da notare sono:
→ è evidente la presenza di un dato che si discosta in modo importante da tutti gli altri;
→ i risultati dei tre metodi non parametrici sono simili tra loro;
→ i risultati delle regressione ordinaria (parametrica) si discostano da quelli delle regressioni non parametriche.

Per verificare quanto il dato, chiaramente anomalo, influenzi le regressioni, eseguiamo nuovamente lo script, sostituendo la terza e la quarta riga con queste due, nelle quali è stato eliminato il dato, il quinto, quello con le coordinate (x,y) uguali a (**24**, **102**):

var_x <- c(18, 46, 104, 72, 63, 23, 58, 36, 82, 94) # variabile in ascisse
var_y <- c(21, 42, 109, 64, 78, 19, 66, 42, 89, 91) # variabile in ordinate

Ora i grafici delle rette sono praticamente sovrapposti (e difficilmente distinguibili tra loro)



mentre i risultati sono diventati i seguenti:

```
> tabris # mostra la tabella con i risultati
                        Intercetta a Coefficiente angolare b
Regressione ordinaria       1.337096               1.019512
Regressione di Theil-Sen    2.729167               1.020833
Regressione di Siegel       2.654334               1.022497
Regressione quantilica      2.581395               1.023256
```

Come si vede dal confronto con i risultati precedenti, con l'eliminazione del dato anomalo i risultati della regressione ordinaria sono cambiati: e questa fornisce ora risultati sovrapponibili a quelli dei tre metodi non parametrici. Poco sono cambiati, rispetti ai precedenti, i risultati della regressione di Thiel-Sen e di Siegel, mentre la regressione quantilica ha addirittura fornito identici valori nei due casi, un fatto molto interessante. I metodi non parametrici confermano la loro robustezza, mentre viene documentato quanto un singolo dato anomalo possa influire sulla regressione lineare ordinaria.

---

[1] Per 100 coppie di dati il numero dei coefficienti angolari (sui quali calcolare la mediana) è uguale quindi a 4 950. Con 500 dati passiamo a 124 740. Con 1 000 dati passiamo a 499 500.

[2] *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

martedì 12 marzo 2019

Grafici di dispersione (grafici xy)

I **grafici di dispersione** o **grafici xy** (grafici cartesiani, diagrammi cartesiani) o **scatterplot**, (scatter graph, scatter chart, scattergram, scatter diagram) sono lo strumento base per visualizzare su un sistema di coordinate cartesiane la relazione tra i valori di due variabili numeriche e quindi per lo studio delle distribuzioni bivariate.

Prendiamo come esempio i dati ematologici relativi agli eritrociti (o globuli rossi), la cui funzione essenziale come noto è trasportare l'ossigeno dai polmoni a tutti gli altri organi e tessuti, e per i quali valgono in media i seguenti dati (arrotondati per semplicità) :

→ gli eritrociti hanno una concentrazione nel sangue all'incirca di $5 \cdot 10^{12}$/L (ovvero ve ne sono all'incirca cinquemila miliardi in un litro di sangue);

→ un eritrocito ha un volume all'incirca di 90 fL (un femtolitro corrisponde a 10^{-15} litri quindi un globulo rosso ha un volume all'incirca di 0.00000000000090 litri).

Da questi dati si ricava che la percentuale del volume del sangue occupata dagli eritrociti (detta valore ematòcrito, e uguale alla concentrazione degli eritrociti moltiplicata per il volume di un eritrocito) è il 45% (la restante parte, esclusi i globuli bianchi e le piastrine, che occupano un volume irrilevante, è liquida ed è rappresentata dal plasma). Ma non solo. Da questi dati si ricava anche che il valore ematòcrito dipende in modo lineare dalla concentrazione degli eritrociti. Possiamo quindi dare un senso al prossimo script, che traccia i grafici di dispersione di queste due grandezze.

I dati sono contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [1].

Copiate lo script e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GRAFICI DI DISPERSIONE (XY)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
str(ais) # struttura di ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
plot(rcc, hc, xlab="Eritrociti (10^12/L)", ylab="Ematòcrito (%)", main="Grafico di dispersione") # grafico di dispersione semplice
#
plot(rcc, hc, pch=2, col="goldenrod", xlab="Eritrociti (10^12/L)", ylab="Ematòcrito (%)", main="Cambia simbolo e colore") # cambia simbolo e colore
#
plot(rcc, hc, pch=1, col="black", abline(lm(hc~rcc), col="red", lty=1, lwd=1), xlab="Eritrociti (10^12/L)", ylab="Ematòcrito (%)", main="Retta di regressione") # grafico con retta di regressione
#
plot(rcc, hc, xlim=c(3,7), ylim=c(30,70), xaxp=c(3, 7, 4), yaxp=c(30, 70, 4), pch=20, col="black", abline(lm(hc~rcc), col="black", lty=1, lwd=1), xlab="Eritrociti (10^12/L)", ylab="Ematòcrito (%)", main="Ridimensiona gli assi") # ridimensiona gli assi
#
detach(ais) # termina l'impiego diretto dei nomi delle variabili
#
```

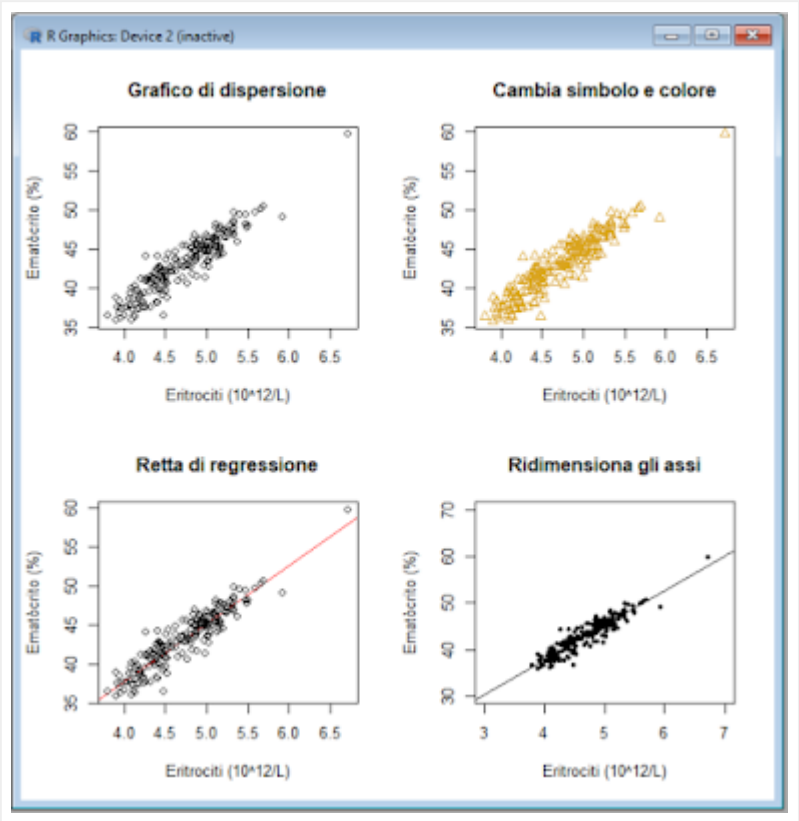
Dopo avere caricato il pacchetto che include il set di dati **ais** e dopo averne mostrato la sua struttura con **str(ais)**, viene impiegata la funzione **attach(ais)** al fine di consente, nelle funzioni successive, di impiegare direttamente i nomi delle variabili incluse nel set di dati.

Per semplificare la presentazione grafica, con **par(mfrow=c(2,2))** la finestra viene suddivisa in quattro quadranti, uno per ciascuno dei grafici che verranno preparati.

Il primo grafico, in alto a sinistra, lascia per la maggior parte degli argomenti della funzione **plot()** i valori di default, specificando solamente quelli essenziali [2]:
→ **rcc** la variabile da riportare sull'asse delle ascisse x;
→ **hc** la variabile da riportare sull'asse delle ordinate y;
→ **xlab=""** l'etichetta da riportare per l'asse delle x;
→ **ylab=""** l'etichetta riportata per l'asse delle y;
→ **main=""** il titolo del grafico che compare in alto.

Nel secondo grafico, in alto a destra, sono aggiunti gli argomenti:
→ **pch=2** che specifica per i dati l'impiego di un simbolo (triangolo) differente da quello di default (che è il cerchio) [3];
→ **col="goldenrod"** che specifica per il simbolo dei dati l'impiego di un colore diverso dal nero [4].

Nel terzo grafico, in basso a sinistra, viene riportata una retta mediante la funzione **abline()** che impiega come argomenti:
→ l'equazione della retta, che altro non è se non la retta di regressione calcolata con la funzione **lm()** che a sua volta impiega come argomenti **hc~rcc** per specificare rispettivamente la variabile in ordinate e la variabile in ascisse;
→ il colore della retta (**col="red"**);
→ il tipo di linea da impiegare (**lty=1**), uno dei sei possibili con i valori da 1 a 6 [5];
→ lo spessore della linea (**lwd=1**) espresso come multiplo dello spessore di default (1 uguale spessore, 2 spessore doppio, eccetera).



Nel quarto grafico, in basso a destra, compaiono quattro nuovi argomenti, che consentono di gestire in modo personalizzabile a piacere le scale degli assi e la loro suddivisione:

→ **xlim = c(3,7)** indica limite inferiore e limite superiore della scala applicata all'asse delle ascisse, trattandosi di valori di concentrazione degli eritrociti espressi in 10^{12}/L questo significa che l'asse delle ascisse riporterà i valori compresi tra $3 \cdot 10^{12}$/L e $7 \cdot 10^{12}$/L;
→ **ylim = c(30,70)** indica limite inferiore e limite superiore della scala applicata all'asse delle ordinate, che, trattandosi di valori di percentuale, varieranno quindi tra 30% e 70%;
→ **xaxp = c(3, 7, 4)** dice che sull'asse delle ascisse la scala che va da $3 \cdot 10^{12}$/L a $7 \cdot 10^{12}$/L deve essere suddivisa in quattro intervalli, quindi sull'asse delle ascisse compariranno cinque tacche con i valori 3, 4, 50, 6, 7;
→ **yaxp = c(30,70,4)** dice che sull'asse delle ordinate la scala che va da 30% e 70% deve essere suddivisa in quattro intervalli, quindi sull'asse delle ordinate compariranno cinque tacche con i valori 30, 40, 50, 60, 70.

Lo script si chiude con la funzione **detach()** che pone termine all'impiego diretto dei nomi delle variabili. In altre parole la variabile **rcc** ora diventa **ais$rcc**, la variabile **hc** ora diventa **ais$hc** e così via.

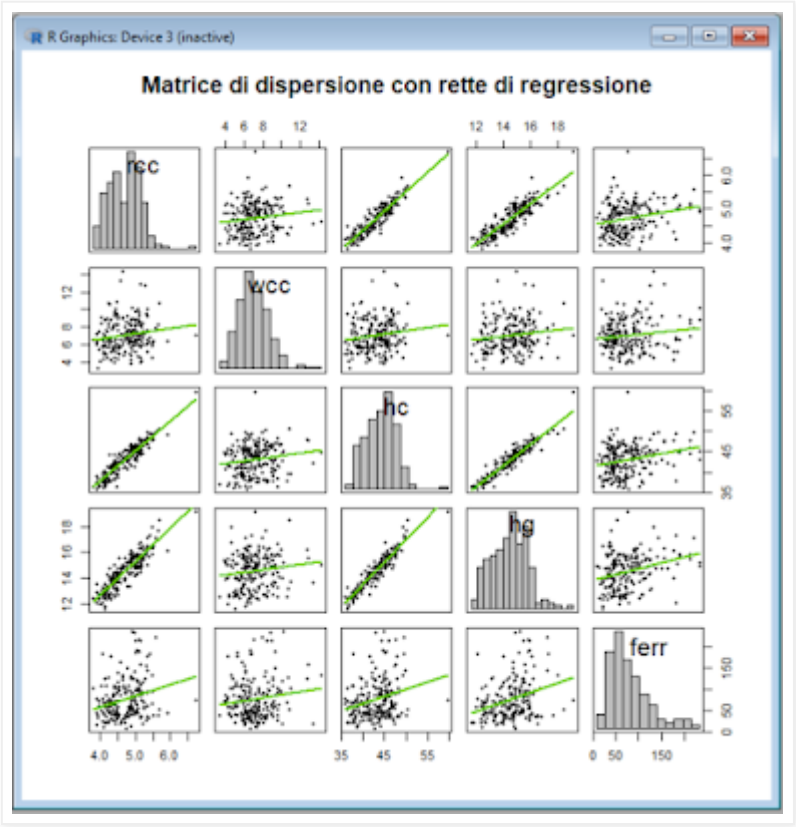
Vediamo ora due grafici che aiutano ad evidenziare le relazioni tra le variabili riportando tutti i possibili grafici di dispersione singoli che risultano incrociando i dati ematologici contenuti nelle seguenti variabili del set di dati **ais**: **rcc** (eritrociti), **wcc** (leucociti), **hc** (ematòcrito), **hg** (emoglobina) e **ferr** (ferritina).

I grafici sono realizzati mediante i pacchetti aggiuntivi **car** e **gclus**: se non l'avete già fatto, dovete scaricarli e installarli dal **CRAN**.

Copiate e incollate nella `Console di R` lo script e premete `↵ Invio`:

```
# GRAFICI DI DISPERSIONE MULTIPLI
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(car) # carica il pacchetto per la grafica
windows() # apre e inizializza una nuova finestra grafica
#
# grafici di dispersione con istogrammi delle variabili
#
scatterplotMatrix(~rcc+wcc+hc+hg+ferr, col="black", pch=20, regLine = list(method=lm, lty=1, lwd=2, col="chartreuse3"), smooth=FALSE, diagonal=list(method ="histogram", breaks="FD"), main="Matrice di dispersione con rette di regressione", data=ais) # mostra il grafico
#
```

L'argomento **~rcc+wcc+hc+hg+ferr** definisce le variabili da rappresentare, e può essere facilmente riscritto per cambiare la rappresentazione a piacere. Ad esempio ponendo questo argomento uguale a **~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt** si possono rappresentare in uno stesso grafico tutte le variabili del set di dati **ais**.



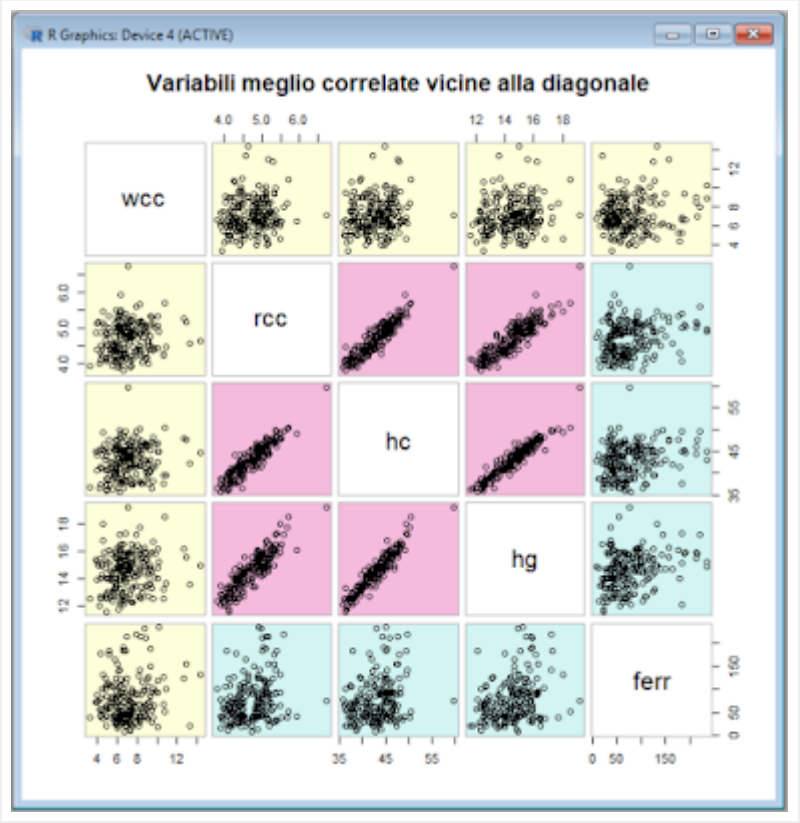
Per la retta di regressione sono impiegati una linea continua (**lty=1**), uno spessore doppio (**lwd=2**) e un colore (**col="chartreuse3"**) diverso dal nero, mentre l'argomento **smooth=FALSE** fa si che non venga aggiunta ai grafici la rappresentazione della funzione non lineare prevista di default nel pacchetto **car**.

Infine l'argomento **diagonal=list()** indica la rappresentazione della variabile riportata nella diagonale, che qui è l'istogramma (**method ="histogram"**) con il numero di classi (**breaks="FD"**) calcolato mediante la regola di Freedman-Diaconis. L'istogramma può essere sostituito con altri tipi di rappresentazione, l'elenco delle espressioni che è possibile impiegare in alternativa per questo argomento (provatele) è:
→ **diagonal=list(method="density", bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE)**
→ **diagonal=list(method="boxplot")**
→ **diagonal=list(method="qqplot")**
→ **diagonal=list(method="oned")**

Copiate e incollate nella `Console di R` lo script e premete `↵ Invio`:

```
# GRAFICI DI DISPERSIONE MULTIPLI ordinati in base al valore di r
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
library(gclus) # carica il pacchetto per la grafica
windows() # apre e inizializza una nuova finestra grafica
mydata <- ais[c(1,2,3,4,5)] # tabella con i dati delle variabili delle colonne 1,2,3,4,5
mydata.r <- abs(cor(mydata)) # calcola la correlazione
mydata.col <- dmat.color(mydata.r) # applica i colori
mydata.o <- order.single(mydata.r) # ordina le variabili, le meglio correlate più vicine alla diagonale
cpairs(mydata, mydata.o, panel.colors=mydata.col, gap=.5, main="Variabili meglio correlate vicine alla diagonale") # mostra il grafico
#
```

Viene realizzato un grafico di dispersione multiplo che incrocia ancora i dati ematologici degli atleti contenuti nelle cinque variabili **rcc** (eritrociti), **wcc** (leucociti), **hc** (ematòcrito), **hg** (emoglobina) e **ferr** (ferritina) ma con due novità.



La prima è che le variabili ora sono individuate non mediante il loro nome bensì indicando, con l'espressione **ais[c(1,2,3,4,5)]**, il set di dati che le contiene e il numero della colonna corrispondente a ciascuna variabile

La seconda novità è che impiegando la funzione **cpairs()** [6] i singoli grafici di dispersione vengono ordinati in base al valore del coefficiente di correlazione **r** di Pearson, in modo che le variabili meglio correlate, quelle per le quali la retta di regressione pare essere un'approssimazione ragionevole, siano le più vicine alla diagonale ed evidenziate in colore, aumentando così la chiarezza della rappresentazione grafica.

Potete riutilizzare facilmente lo script sostituendo all'oggetto **ais** l'oggetto contenente i vostri dati, opportunamente strutturati, e modificando conseguentemente i nomi della variabili. Per una guida rapida all'importazione dei dati potete consultare i link:
→ importazione dei dati da un file .csv
→ importazione dei dati da un file .xls o .xlsx
→ gestione dei dati mancanti

---

[1] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[2] Per la documentazione sugli argomenti della funzione **plot()** digitare **help(plot)** nella `Console di R`, e in aggiunta consultare la documentazione sugli argomenti della funzione **par()** con **help(par)**.

[3] Per i diversi simboli che è possibile impiegare per tracciare i punti vedere il post I simboli dei punti di R.

[4] Per i colori che è possibile impiegare vedere il post La tabella dei colori di R.

[5] Per le linee che è possibile impiegare vedere il post Gli stili delle linee di R.

[6] Digitate **help(cpairs)** nella `Console di R` per la documentazione della funzione **cpairs()** ovvero consultate il manuale di riferimento del pacchetto **gclus** su: *Available CRAN Packages By Name*. URL consultato il 20/10/2018: https://goo.gl/hLC9BB

giovedì 18 aprile 2019

Identificare i dati in un grafico xy con un click

Accade con un certa frequenza, nel corso dell'analisi esplorativa dei dati, osservando un grafico di dispersione (xy), di porsi la domanda: questo valore, che si discosta così tanto dagli altri, a quale dato corrisponde?

R fornisce con uno strumento molto efficace per identificare il numero del record/riga che corrisponde ad uno specifico punto riportato in un grafico, la funzione **identify()** [1], il cui utilizzo viene illustrato con questo script. Copiatelo e incollatelo nella Console di R e premete ↵ Invio:

```
# IDENTIFICA I DATI IN UN GRAFICO CON UN SEMPLICE CLICK
#
library(DAAG) # carica il pacchetto che include il set di dati ais
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
windows() # apre una nuova finestra
#
plot(rcc, hc, main="Identifica i punti in un grafico", xlab="Eritrociti (10^12/L)", ylab="Ematòcrito (%)", pch=1) # traccia il grafico di dispersione (xy)
#
# fare click accanto al punto da identificare, ripetere secondo necessità, premere <esc> per terminare
identify(rcc, hc, plot=TRUE, atpen=FALSE, offset=0.5, tolerance=0.25)
#
```

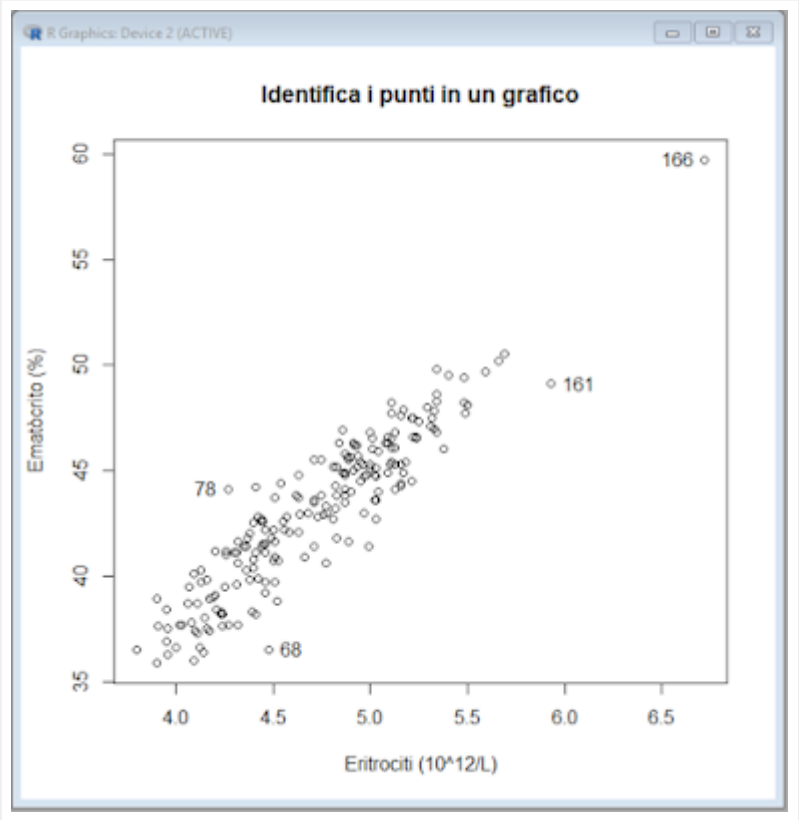
I dati impiegati sono contenuti nel set di dati ais. Dopo che con la funzione **plot()** [2] è stato tracciato il grafico di dispersione (xy), la funzione **identify()** consente di identificare nel grafico uno o più punti impiegando i seguenti argomenti:
→ **rcc** la variabile in ascisse;
→ **hc** la variabile in ordinate;
→ **plot=TRUE** che visualizza l'identificativo del dato sul quale si è fatto click con il mouse;
→ **atpen=FALSE** che posiziona l'identificativo del dato lievemente discosto dal punto al quale corrisponde (provare a cambiare l'argomento ponendo **atpen=TRUE** per vedere la differenza);
→ **offset=0.5** che indica la distanza che l'identificativo del dato deve avere dal punto al quale corrisponde;
→ **tolerance=0.25** che indica la tolleranza ammessa per la distanza definita con l'argomento **offset**.

Facendo click con il tasto sinistro del mouse accanto a un punto, vedrete comparire il suo identificativo. L'operazione può essere ripetuta per più punti. Di default viene emesso un suono ogniqualvolta si fa click e viene identificato un punto: per silenziare la funzione **identify()** è necessario aggiungere l'argomento **locatorBell=FALSE**.

Per uscire dal processo di identificazione e terminare lo script è sufficiente premere il tasto `<esc>`.

Questo è il grafico ottenuto dopo avere fatto click con il tasto sinistro del mouse sui valori che più si discostano dai rimanenti, valori identificati dalla funzione **identify()** come corrispondenti ai dati numero 68, 78, 161 e 166.

Come è facile intuire questa funzione grafica avanzata è molto interessante perché permette di individuare e identificare quelli che i test statistici possono avere indicato come possibili dati anomali (o dati aberranti o outliers).



Potete riutilizzare facilmente lo script sostituendo all'oggetto **ais** l'oggetto contenente i vostri dati, opportunamente strutturati, e modificando conseguentemente i nomi della variabili. Per una guida rapida all'importazione dei dati potete consultare i link:
→ importazione dei dati da un file .csv
→ importazione dei dati da un file .xls o .xlsx
→ gestionedei dati mancanti

---

[1] Per la documentazione della funzione **identify()** digitare **help(identify)** nella *Console di R*.

[2] Per la documentazione della funzione **plot()** digitare **help(plot)** nella *Console di R*.

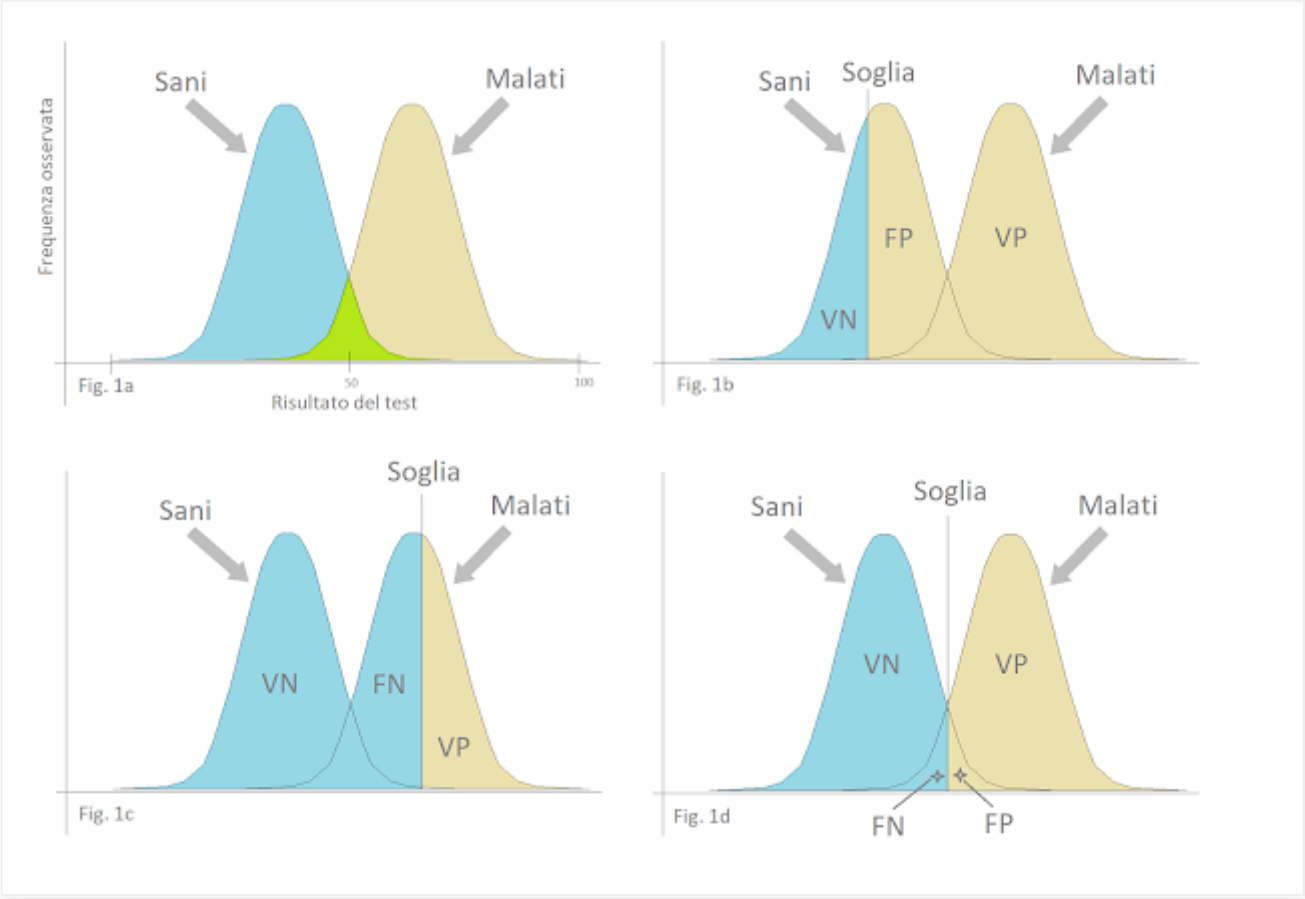
venerdì 10 maggio 2019

Curve ROC e valore soglia

Nel post Teorema di Bayes e diagnosi medica abbiamo visto che è possibile impiegare il teorema per rispondere alle domande: quale probabilità ha di essere affetto da una data malattia un soggetto cui un test per quella malattia è risultato positivo? quale probabilità ha di essere sano un soggetto cui il test è risultato negativo?

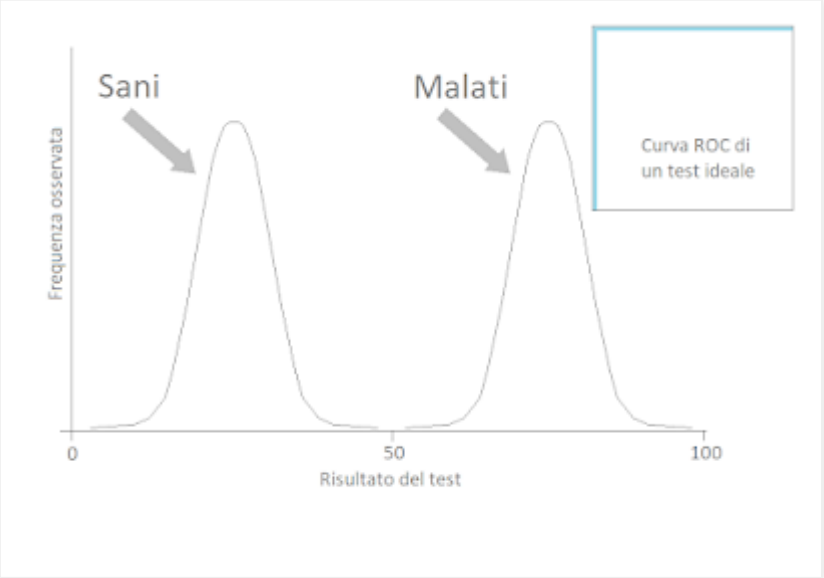
Per farlo dobbiamo calcolare il **valore soglia** che meglio discrimina tra sani e malati, e ricavare la **sensibilità** e la **specificità** del test che ad esso corrispondono, impiegando:
→ i risultati del test in questione nei sani e nei malati;
→ un metodo per calcolare il valore soglia del test che meglio discrimina tra sani e malati (sensibilità e specificità sono una conseguenza di questo).

I risultati del test sono raccolti eseguendolo, in condizioni controllate, nel corso di una sperimentazione, a un congruo numero di soggetti sani e di soggetti affetti dalla malattia diagnosticata dal test. Il metodo per calcolare il valore soglia applica la strategia diagnostica riportata nella Fig. 1d

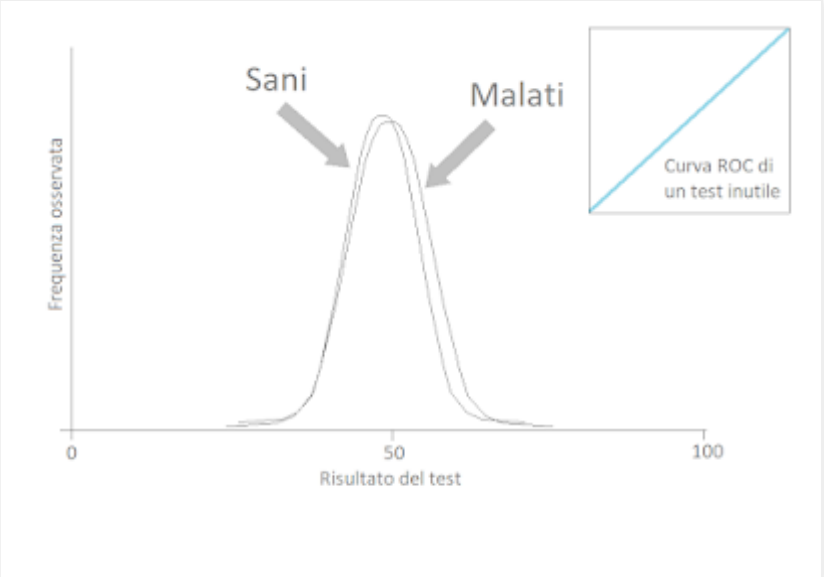


che consente di identificare malati (casi) e sani (controlli) massimizzando veri positivi (VP) e veri negativi (VN), minimizzando falsi positivi (FP) e falsi negativi (FN) e individuando quindi il miglior equilibrio tra sensibilità e specificità del test. Alcuni cenni su questa e le altre strategie applicabili sono riportati nel post Teorema di Bayes e diagnosi medica.

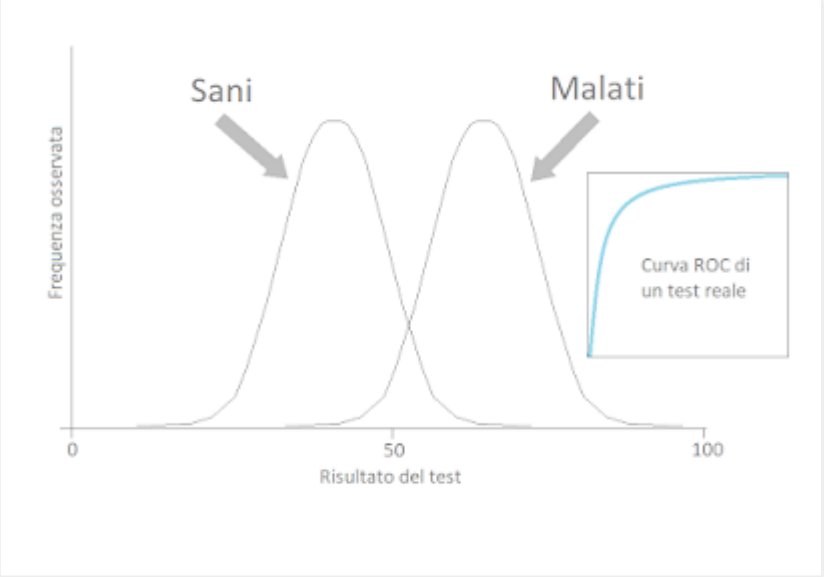
La relazione che intercorre tra i risultati del test ottenuti in un campione di soggetti sani, quelli ottenuti in un campione di soggetti malati e la loro curva ROC [1] è riportata in questa figura per un test diagnostico ideale, nel quale le distribuzioni sono completamente separate, e l'area sotto la curva AUC (Area Under the Curve) è uguale a 1



in quest'altra figura per un test diagnostico inutile nel quale le distribuzioni sono completamente sovrapposte, non è possibile discriminare, impiegando i risultati dello specifico test diagnostico, tra malati e sani, e l'AUC è uguale a 0.5



e infine in questa figura per un test diagnostico reale, nel quale è presente una parziale sovrapposizione tra i risultati del test nei sani e nei malati e l'AUC è tanto più prossima a 1 quanto meglio il test è in grado di discriminare i soggetti sani da quelli malati.



Quindi la curva ROC fornisce una misura della informazione fornita dal test, espressa come probabilità

(misurata dall'AUC) che va da 0.5 (equivalente a decidere se un soggetto è sano o malato mediante il lancio di una moneta) a 1 (il test ci dice con certezza se il paziente è sano o malato). Per impiegare le funzioni del pacchetto **pROC**, i risultati del test raccolti durante la sperimentazione devono essere strutturati in due variabili, una variabile qualitativa, in **R** definita "fattore", e qui denominata `classe`, che contiene la classe di appartenenza (qui vengono impiegati i valori previsti di default nel pacchetto e cioè 0 per indicare un soggetto sano/controllo e 1 per indicare un soggetto malato/caso) e una variabile quantitativa, qui denominata `valore`, che per ciascun caso riporta il risultato del test, come appare in questi dati, i primi cinque e gli ultimi cinque del file bayes.csv che ci apprestiamo a impiegare:

```
classe;valore
0;19
0;22
0;26
0;28
0;29
.....
1;235
1;237
1;237
1;242
1;242
```

Quindi la coppia di valori `0;19` indica che in un sano/controllo (`0`) il risultato fornito dal test era `19`; la coppia di valori `1;235` indica che in un malato/caso (`1`) il risultato fornito dal test era `235`; e così via.

Per proseguire ora è necessario:
→ effettuare il download del file `bayes.csv`
→ salvare il file nella cartella `C:\Rdati\`

Trovate il file, con le istruzioni per scaricare questo e gli altri file di dati impiegati nei post, alla pagina Dati.

Per effettuare i calcoli impieghiamo alcune funzioni contenute nei pacchetti aggiuntivi **pROC** e **sm** [2], che devono essere preventivamente scaricati dal **CRAN.** Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# CURVE ROC E VALORE SOGLIA
#
mydata <- read.table("c:/Rdati/bayes.csv", header=TRUE, sep=";") # importa i dati
attach(mydata) # consente di impiegare direttamente i nomi delle variabili
str(mydata) # mostra la struttura dei dati della tabella importata
head(mydata, n=5) # lista dei primi 5 casi
tail(mydata, n=5) # lista degli ultimi 5 casi
#
sani <- subset(valore, classe==0) # sottoinsieme con i valori dei soggetti sani
malati <- subset(valore, classe==1) # sottoinsieme con i valori dei soggetti malati
#
windows() # apre una nuova finestra
par(mfrow=c(2,2)) # predispone la suddivisione della finestra in quattro quadranti, uno per grafico
#
# primo istogramma
#
hist(malati, xlim=c(0,250), ylim=c(0,0.04), xaxp = c(0, 250, 5), yaxp = c(0, 0.04, 4),
breaks=seq(from=0, to=250, by=5), border="red", col="white", main="Istogramma",
```

xlab="Valore osservato", ylab="Densità", freq=FALSE, plot=TRUE) # traccia istogramma malati
hist(sani, xlim=c(0,250), breaks=seq(from=0, to=250, by=5), border="green4", col="lightgreen", freq=FALSE, add=TRUE, plot=TRUE) # sovrappone istogramma sani
legend(150, 0.04, legend=c("Sani", "Malati"), col=c("green", "red"), lty=c(1,1), lwd=c(1,1), cex=0.8) # aggiunge al grafico la legenda
#
secondo istogramma
#
hist(sani, xlim=c(0,250), ylim=c(0,0.04), xaxp = c(0, 250, 5), yaxp = c(0, 0.04, 4), breaks=seq(from=0, to=250, by=5), border="green4", col="lightgreen", main = "Istogramma", xlab = "Valore osservato", ylab = "Densità", freq=FALSE, plot=TRUE) # traccia istogramma sani
hist(malati, breaks=seq(from=0, to=250, by=5), border="red", col="white", main="", freq=FALSE, add=TRUE, plot=TRUE) # sovrappone istogramma malati
legend(150, 0.04, legend=c("Sani", "Malati"), col=c("green", "red"), lty=c(1,1), lwd=c(1,1), cex=0.8) # aggiunge al grafico la legenda
#
kernel density plot
#
library(sm) # carica il pacchetto
sm.density.compare(valore, classe, xlim=c(0,250), col=c("green", "red"), main="Kernel density plot", xlab="Valore osservato", ylab="Stima kernel di densità") # traccia kernel density plot separati per sani e malati
title(main="Kernel density plot") # aggiunge al grafico un titolo
legend(150, 0.03, legend=c("Sani", "Malati"), col=c("green", "red"), lty=c(1,2), lwd=c(1,1), cex=0.8) # aggiunge al grafico la legenda
#
curva ROC
#
library(pROC) # carica il pacchetto
roc(response=classe, predictor=valore, smooth=FALSE, auc=TRUE, ci=TRUE, plot=TRUE, col="orange", identity=TRUE, identity.col="grey", identity.lty=3, main="Curva ROC", xlab="Specificità", ylab="Sensibilità") # traccia la curva ROC e riporta l'AUC con gli intervalli di confidenza
#
calcola valore soglia, sensibilità e specificità con gli intervalli di confidenza
#
ci.thresholds(response=classe, predictor=valore, thresholds="best", conf.level=0.95, boot.n=100) # calcola il miglior valore soglia tra sani e malati e la sensibilità e la specificità corrispondenti con gli intervalli di confidenza al 95%
ci.se(response=classe, predictor=valore, specificities=seq(0,1,.1), conf.level=0.95, boot.n=100) # calcola gli intervalli di confidenza al 95% della sensibilità per valori di specificità da 0 a 1 con passo .1
ci.sp(response=classe, predictor=valore, sensitivities=seq(0,1,.1), conf.level=0.95, boot.n=100) # calcola gli intervalli di confidenza al 95% della specificità per valori di sensibilità da 0 a 1 con passo .1
#

La parte dello script che precede la creazione dei grafici si spiega da sola, con i commenti inseriti in ciascuna riga. L'unica cosa un po' particolare è la funzione **subset()** [3] che viene impiegata per generare, dai dati importati, due sottoinsiemi separati, uno contenente solamente i valori dei soggetti sani/controlli e l'altro contenente solamente i valori dei soggetti malati/casi, impiegando come argomenti:
→ il nome della variabile (**valore**) dalla quale si desidera estrarre il sottoinsieme di valori;

→ il criterio di selezione dei casi da estrarre, laddove **classe==0** indica di estrarre i valori per i quali nella variabile **classe** è riportato 0 (sano) e laddove **classe==1** indica di estrarre i valori per i quali nella variabile **classe** è riportato 1 (malato).

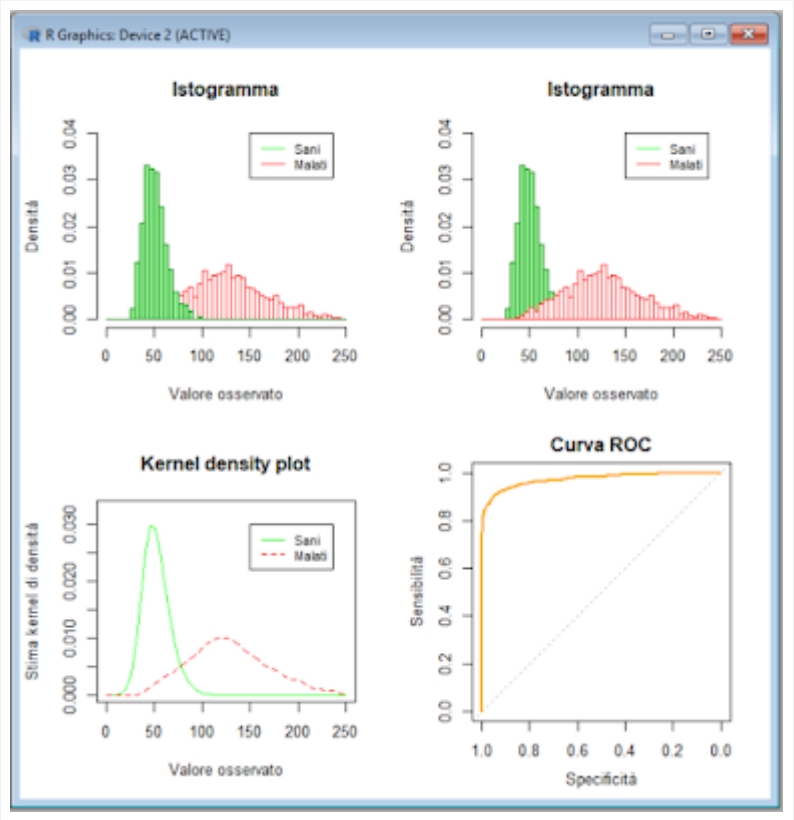
Nel blocco di codice che realizza gli istogrammi sono degni di nota i seguenti argomenti:
→ **xlim = c(0,250)** indica limite inferiore e limite superiore della scala applicata all'asse delle ascisse;
→ **ylim = c(0,0.04)** indica limite inferiore e limite superiore della scala applicata all'asse delle ordinate, sulla quale è riportata la densità (di probabilità) della distribuzione dei dati;
→ **xaxp = c(0, 250, 5)** specifica che sull'asse delle ascisse la scala che va da 0 a 250 deve essere suddivisa in cinque intervalli, quindi sull'asse delle ascisse compariranno sei tacche con i valori 0, 50, 100, 150, 200, 250;
→ **yaxp = c(0,0.04,4)** specifica che sull'asse delle ordinate la scala che va da 0 e 0.04 deve essere suddivisa in quattro intervalli, quindi sull'asse delle ordinate compariranno cinque tacche con i valori 0, 0.01, 0.02, 0.03, 0.04;
→ **breaks=seq(from=0, to=250, by=5)** traccia l'istogramma tra 0 e 250 con classi di ampiezza 5;
→ **freq=FALSE** comporta di riportare sull'asse delle ordinate non il numero di casi osservati in ciascuna classe bensì la densità di probabilità;
→ **add=TRUE** permette che il secondo istogramma sia sovrapposto al primo.

Per la funzione **legend()** si rimanda a [4].

Per i kernel density plot sovrapposti l'unica cosa significativa è che sono tracciati mediante la funzione **sm.density.compare()** del pacchetto aggiuntivo **sm**, mentre la funzione **legend()** semplicemente adatta ai nuovi grafici gli argomenti già visti nel caso degli istogrammi.

La curva ROC viene tracciata mediante la funzione **roc()** che prevede come argomenti:
→ la variabile che contiene la classificazione di ciascun caso (**response=caso**);
→ la variabile che contiene il valore di ciascun caso (**predictor=valore**);
→ **smooth=FALSE** in quanto non vogliamo che la curva ROC sia smussata (ma potete provare a farlo mettendo **TRUE**);
→ **auc=TRUE** in quanto vogliamo che sia calcolata l'area sotto la curva AUC;
→ **ci=TRUE** in quanto vogliamo che sia riportato l'intervallo di confidenza al 95% della AUC;
→ il colore della curva ROC (**col="orange"**);
→ **identity=TRUE** per riportare nel grafico la retta che va dall'angolo inferiore sinistro all'angolo superiore destro, **identity.col** per specificarne il colore, **identity.lty** per specificare lo stile della linea, che rappresenta la curva ROC di un test nel quale l'AUC è uguale a 0.5, i valori dei sani e quelli dei malati sono completamente sovrapposti, e non è possibile discriminare gli uni dagli altri (Fig. 2). In altre parole, se la curva ROC del test coincide con la diagonale, il test fornisce la stessa informazione che viene fornita dal lancio di una moneta;
→ i noti argomenti impiegati per aggiungere a un grafico il titolo e le legende degli assi.

La funzione **roc()** completa i grafici [5] con la curva ROC del test [6]



quindi riporta il valore dell'AUC (0.9734) con il suo intervallo di confidenza al 95%:

```
Data: valore in 853 controls (classe 0) < 843 cases (classe 1).
Area under the curve: 0.9734
95% CI: 0.9664-0.9805 (DeLong)
```

La funzione **ci.thresholds()** riporta il miglior valore soglia tra sani e malati (74.5), insieme ai valori di specificità (0.9484) e di sensibilità (0.9063) corrispondenti, con i relativi intervalli di confidenza al 95% (**conf.level=0.95**):

```
>       ci.thresholds(response=classe,       predictor=valore,       thresholds="best",
conf.level=0.95, boot.n=100) # calcola il miglior valore soglia tra sani e malati e
la sensibilità e la specificità corrispondenti con gli intervalli di confidenza al
95%
95% CI (100 stratified bootstrap replicates):
 thresholds sp.low sp.median sp.high se.low se.median se.high
       74.5 0.9349    0.9484  0.9643 0.8843    0.9063  0.9242
```

La funzione **ci.se()** consente di tabulare i valori di sensibilità corrispondenti ai valori di specificità prescelti, che qui con l'argomento **specificities=seq(0,1,.1)** sono fatti variare da 0 a 1 con passo 0.1, con i relativi intervalli di confidenza al 95% (**conf.level=0.95**):

```
>       ci.se(response=classe,       predictor=valore,       specificities=seq(0,1,.1),
conf.level=0.95, boot.n=100) # calcola gli intervalli di confidenza al 95% della
sensibilità per valori di specificità da 0 a 1 con passo .1
  sp se.low se.median se.high
 0.0 1.0000    1.0000  1.0000
 0.1 1.0000    1.0000  1.0000
 0.2 0.9941    0.9976  1.0000
 0.3 0.9882    0.9941  0.9988
 0.4 0.9844    0.9912  0.9975
 0.5 0.9754    0.9858  0.9923
 0.6 0.9709    0.9821  0.9899
```

```
 0.7 0.9567    0.9678  0.9762
 0.8 0.9480    0.9597  0.9719
 0.9 0.9138    0.9311  0.9453
 1.0 0.6494    0.6821  0.7788
```

La funzione **ci.sp()** consente di tabulare i valori di specificità corrispondenti ai valori di sensibilità prescelti, che qui con l'argomento **sensitivities=seq(0,1,.1)** sono fatti variare da 0 a 1 con passo 0.1, con i relativi intervalli di confidenza al 95% (**conf.level=0.95**):

```
> ci.sp(response=classe, predictor=valore, sensitivities=seq(0,1,.1), conf.level=0.95, boot.n=100) # calcola gli intervalli di confidenza al 95% della specificità per valori di sensibilità da 0 a 1 con passo .1
  se sp.low sp.median sp.high
 0.0 1.0000    1.0000  1.0000
 0.1 1.0000    1.0000  1.0000
 0.2 1.0000    1.0000  1.0000
 0.3 1.0000    1.0000  1.0000
 0.4 1.0000    1.0000  1.0000
 0.5 1.0000    1.0000  1.0000
 0.6 1.0000    1.0000  1.0000
 0.7 0.9965    0.9990  1.0000
 0.8 0.9900    0.9953  0.9999
 0.9 0.9312    0.9538  0.9753
 1.0 0.1354    0.1583  0.2446
```

Le funzioni **ci.treshold()**, **ci.se()**, **ci.sp()** impiegano un metodo di bootstrap per calcolare gli intervalli di confidenza: questo significa che i risultati generati ogni volta saranno lievemente differenti da quelli qui riportati.

L'AUC è uguale a `0.9734` (`0.9664-0.9805`); il miglior valore soglia tra sani e malati è `74.5` - per un valore inferiore o uguale al valore soglia il test è considerato negativo e per un valore superiore al valore soglia il test è considerato positivo [7].

Per il **valore soglia** `74.5` abbiamo:
→ un valore di **specificità del test** (mediana) uguale a `0.9484` che significa che il test è negativo nel 94.8% dei soggetti sani;
→ un valore di **sensibilità del test** (mediana) uguale a `0.9063` che significa che il test è positivo nel 90.6% dei soggetti malati.

Da notare che una volta fissato il valore soglia tra sani e malati, la sensibilità e la specificità ad esso corrispondenti diventano le due caratteristiche diagnostiche tipiche del test.

Infine per importare ed elaborare i vostri dati, dopo averli strutturati come nel file bayes.csv, dovete semplicemente sostituire nella prima riga di codice **c:/Rdati/bayes.csv** con nome e posizione del vostro file di dati.

Potete trovare:
→ nel post Curve ROC sovrapposte come riportare sullo stesso grafico due curve ROC per un più immediato confronto tra i risultati di due test per la diagnosi della stessa malattia;
→ nel post Sensibilità, specificità e valore predittivo come il valore predittivo del test positivo (la probabilità di essere malato per un soggetto che presenta un test positivo) e il valore predittivo del test negativo (la probabilità di essere sano per un soggetto che presenta un test negativo) possono essere calcolati impiegando il numero di casi osservati VP, FP, FN, VN;
→ nel post Teorema di Bayes e diagnosi medica come il valore predittivo del test positivo e quello del test negativo possono essere calcolati impiegando la sensibilità del test, la specificità del test e la prevalenza della malattia mediante la (iv) e la (v) lì riportate.

Rispondendo così finalmente alle due domande iniziali.

---

[1] Per una sintesi sul tema delle curve ROC vedere Altman nella pagina Bibliografia.

[2] Trovate la documentazione completa dei pacchetti, indispensabile per impiegarne queste e le altre funzioni, nei loro manuali di riferimento che trovate alla pagina *Available CRAN Packages By Name*. URL consultato il 10/05/2019: https://goo.gl/hLC9BB

[3] Digitate **help(subset)** nella `Console di R` per la documentazione della funzione **subset()**.

[4] Vedere il post Aggiungere la legenda a un grafico.

[5] A scopo didattico i grafici sono stati compattati in un'unica figura, ma è possibile riportarli separatamente, impiegando la trasparenza dei colori per un miglior aspetto della sovrapposizione (vedere il post Istogrammi e il post Kernel density plot).

[6] Sull'asse delle ascisse del grafico di una curva ROC sono normalmente riportati i valori `1 – Specificità`, valori che vanno da `0` a `1`, con lo zero a sinistra e i valori in ordine crescente fino a 1 verso destra: come si confà a un grafico cartesiano. La rappresentazione realizzata con il pacchetto **pROC** riporta invece sull'asse delle ascisse i valori di `Specificità`, che, da sinistra verso destra, vanno da `1` a `0`: una rappresentazione non canonica, che ha chiaramente lo scopo di evidenziare il rapporto inverso esistente tra sensibilità e specificità.

[7] L'esempio qui impiegato rappresenta la situazione più frequente, quella nella quale la malattia determina un aumento del risultato dei test, ma ovviamente è sufficiente ribaltare dati e conclusioni per riadattare il tutto a un caso nel quale la malattia determina una diminuzione del risultato del test.

venerdì 14 giugno 2019

Curve ROC sovrapposte

Le curve ROC sono già state illustrate [1]. Ora aggiungiamo un breve script che consente di riportare sullo stesso grafico le curve ROC di due diversi test per la diagnosi di una malattia X, qui denominati test 1 e test 2, onde confrontare anche visivamente la loro capacità di discriminare tra sani e malati, espressa numericamente come **AUC** (**A**rea **U**nder the **C**urve, area sotto la curva ROC).

Per proseguire è necessario:
→ effettuare il download del file `test_1.csv` e del file `test_2.csv`
→ salvare entrambi i file nella cartella `C:\Rdati\`

Trovate i file, con le istruzioni per scaricare questi e gli altri file di dati impiegati nei post, alla pagina Dati.

Volendo impiegare le funzioni del pacchetto **pROC** i file di dati devono contenere i risultati del/i test strutturati in due variabili, una variabile quantitativa, qui denominata `valore`, che per ciascun caso riporta il risultato del test, e una variabile qualitativa, in **R** definita "*fattore*", e qui denominata classe, che contiene la classe di appartenenza (qui vengono impiegati i valori previsti di default nel pacchetto e cioè `0` per indicare un soggetto sano/controllo e `1` per indicare un soggetto malato/caso) come vedete ad esempio aprendo con un editor di testo il file `test_1.csv` che contiene i risultati del `test 1`:

```
valore;classe
19;0
22;0
22;1
24;1
24;1
26;0
```

Quindi la coppia di valori `19;0` indica che in un soggetto sano/controllo (`0`) il risultato fornito dal test era `19`; la coppia di valori `22;0` indica che in un soggetto sano/controllo (`0`) il risultato fornito dal test era `22`; la coppia di valori `22;1` indica che in un soggetto malato/caso (`1`) il risultato fornito dal test era `22`; e così via.

Per effettuare i calcoli sono impiegate le funzioni del pacchetto aggiuntivo **pROC** [1] che deve essere scaricato, se non ancora fatto in precedenza, dal **CRAN.** Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# CONFRONTO TRA DUE CURVE ROC
#
# importa i dati dei due test da mettere a confronto
test_1 <- read.table("c:/Rdati/test_1.csv", header=TRUE, sep=";")
test_2 <- read.table("c:/Rdati/test_2.csv", header=TRUE, sep=";")
#
library(pROC)
#
# traccia la curva ROC del primo test
roc(response=test_1$classe, predictor=test_1$valore, smooth=FALSE, auc=TRUE, ci=FALSE, plot=TRUE, identity=FALSE, main="Confronto tra due curve ROC", xlab="Specificità", ylab="Sensibilità")
# calcola valore soglia, sensibilità e specificità del primo test
```

```
ci.thresholds(response=test_1$classe, predictor=test_1$valore, thresholds="best", conf.level=0.95, boot.n=100)
#
# sovrappone la curva ROC del secondo test
roc(response=test_2$classe, predictor=test_2$valore, smooth=FALSE, auc=TRUE, ci=FALSE, plot=TRUE, add=TRUE, col="red", lty=4)
# calcola valore soglia, sensibilità e specificità del secondo test
ci.thresholds(response=test_2$classe, predictor=test_2$valore, thresholds="best", conf.level=0.95, boot.n=100)
#
# aggiunge una legenda al grafico
legend(0.35, 0.8, legend=c(paste("Test 1, AUC =", round(auc(response=test_1$classe, predictor=test_1$valore), digits=3), sep=" "), paste("Test 2, AUC =", round(auc(response=test_2$classe, predictor=test_2$valore), digits=3), sep=" ")), col=c("black", "red"), lty=c(1,4), cex=0.8) # aggiunge al grafico la legenda
#
```

Le prime due righe di codice provvedono a importare i dati. Con la terza viene caricato il pacchetto **pROC**.

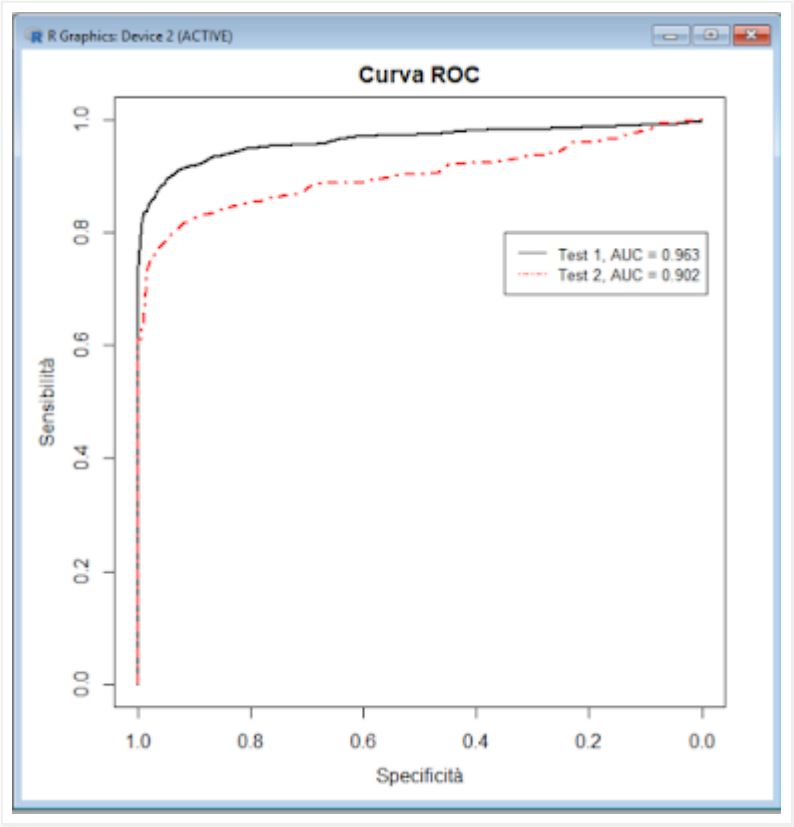
Con la quarta e la quinta riga di codice le funzioni **roc()** e **ci.treshold()** sono impiegate rispettivamente per tracciare il grafico e calcolare le statistiche del primo test. Entrambe le funzioni le trovate illustrate in dettaglio in [1], nel manuale di riferimento del pacchetto [1] e anche digitando **help(roc)** e **help(ci.treshold)** nella `Console di R`.

Nelle due righe successive la funzione **roc()** viene impiegata con l'argomento **add=TRUE** per sovrapporre la seconda curva ROC alla prima, quindi con la funzione **ci.treshold()** sono calcolate le statistiche del secondo test.

Infine con l'ultima riga di codice al grafico viene aggiunta una legenda impiegando la funzione **legend()** i cui argomenti sono:
→ **0.3** per indicare il valore in ascisse al quale posizionare l'angolo superiore sinistro del box della legenda;
→ **0.75** per indicare il valore in ordinate al quale posizionare l'angolo superiore sinistro del box della legenda;
→ **legend** che riporta con la funzione **c()** le etichette che descrivono il primo e il secondo test;
→ **col** che riprende con la funzione **c()** i colori assegnati alle due curve ROC;
→ **lty** che riprende lo stile delle linee delle due curve ROC;
→ **cex** che dimensiona i caratteri della legenda.

Da notare che la funzione **legend()** è stata condensata in una sola riga di codice sfruttando una caratteristica fondamentale della programmazione a oggetti, cioè il fatto che una funzione può impiegare come valore per un argomento il risultato di una funzione, che a sua volta può impiegare come valore per un argomento il risultato di una funzione, e così via, come avviene in questo caso, nel quale:
→ la funzione **legend()**, che costruisce la legenda,
→ impiega come terzo argomento **legend**, che riporta il contenuto della legenda
→ che a sua volta impiega il risultato della funzione **c()**, che fornisce i valori all'argomento **legend**
→ impiegando come primo argomento la funzione **paste()**
→ che impiega come secondo argomento il risultato della funzione **round()**, che arrotonda i risultati
→ impiegando come argomento il risultato della funzione **auc()**, che calcola il valore dell'area sotto la curva AUC impiegando come argomenti i dati del file `test_1.csv`
ripetendo infine gli ultimi tre passi anche per i dati del file `test_2.csv`. Anche se va a scapito della leggibilità del codice, che di norma è preferibile spezzare su più righe introducendo opportune variabili accessorie, questo è comunque un esempio della possibilità di concisione offerta da **R**.

Ecco il grafico ottenuto mediante lo script:



La conclusione è che per la diagnosi della malattia X è preferibile impiegare il test 1, che con una AUC = `0.963` fornisce un'informazione diagnostica superiore a quella del test 2 (AUC = `0.902`). E questo è espresso anche dal fatto che il test 1 ha sensibilità e specificità entrambe superiori a quelle del test 2.

Una analisi separata di ciascuno dei due test diagnostici qui messi a confronto può essere effettuata mediante lo script riportato in [1] semplicemente cambiando nella prima riga di codice il nome del file, sostituendo quindi `bayes.csv` prima con `test_1.csv` e poi con `test_2.csv`.

---

[1] Vedere il post Curve ROC e valore soglia.

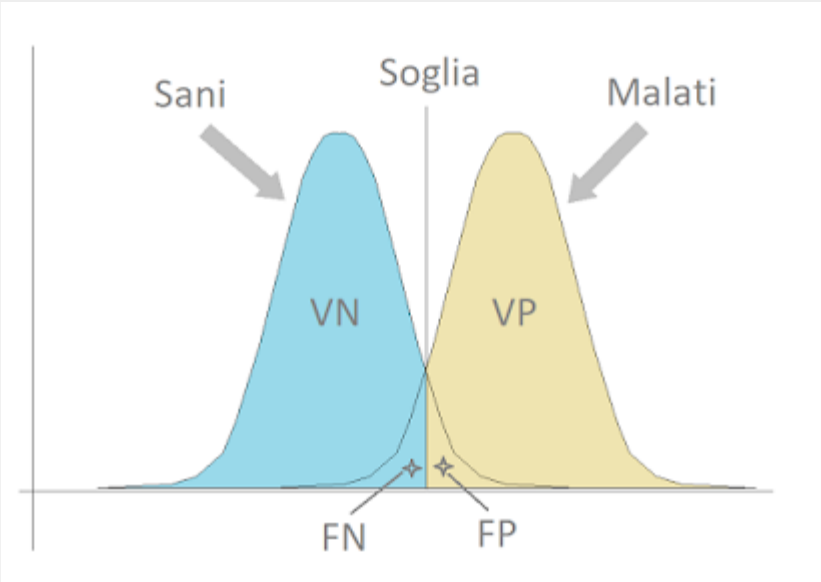
[1] Per la funzione **roc()** e la funzione **ci.tresholds()** vedere il manuale di riferimento del pacchetto *Package 'pROC'*. URL consultato il 14/06/2019: http://bit.ly/2HBk7R6

domenica 14 luglio 2019

Sensibilità, specificità e valore predittivo

Nel post Teorema di Bayes e diagnosi medica abbiamo visto che è possibile impiegare il teorema per rispondere alle due domande: quale probabilità ha di essere affetto da una specifica malattia un soggetto cui un dato test per quella specifica malattia è risultato positivo? quale probabilità ha di essere sano un soggetto cui il test è risultato negativo?

Nel post Curve ROC e valore soglia abbiamo visto, dati i risultati di un test per la diagnosi di una malattia X, come impiegare il pacchetto **pROC** per tracciare la curva ROC [1] e individuare il **valore soglia** che consente di classificare correttamente il maggior numero possibile di soggetti sani (VN) e di classificare correttamente il maggior numero possibile di soggetti malati (VP), minimizzando sia la possibilità di sbagliare classificando dei soggetti malati come sani (FN) sia la possibilità di sbagliare classificando dei soggetti sani come malati (FP). Dato il valore soglia individuato, risultano conseguentemente fissate la **sensibilità** e la **specificità** del test.



Ora impieghiamo il pacchetto **epiR** per calcolare le probabilità che rispondono alle due domande e cioè:
→ il **valore predittivo di un test positivo**, ovvero la probabilità di essere malato per un soggetto che presenta un test positivo, che risponde alla prima domanda
→ il **valore predittivo di un test negativo**, ovvero la probabilità di essere sano per un soggetto che presenta un test negativo, che risponde alla seconda domanda.

Si stabilisce se il test è positivo o negativo impiegando il valore soglia individuato, quindi si calcola il valore predittivo del test impiegando:
→ la sensibilità del test, cioè la probabilità che il test sia positivo in un soggetto malato;
→ la specificità del test, cioè la probabilità che il test sia negativo in un soggetto sano;
→ la prevalenza, cioè la probabilità della malattia.

La sensibilità e la specificità sono caratteristiche intrinseche al test, dipendono dal valore soglia individuato e sono indipendenti dalla prevalenza, che è invece una caratteristica della popolazione alla quale il test viene applicato.

Notare che la prevalenza, cioè la probabilità della malattia nella popolazione alla quale il test viene applicato, è la probabilità a priori della malattia, ovvero la probabilità per un dato paziente di essere malato (o sano) prima di avere eseguito il test. Il teorema di Bayes impiega la probabilità a priori (nel nostro caso la prevalenza della malattia), insieme a sensibilità e specificità del test, per calcolare

la probabilità a posteriori, ovvero la probabilità per un dato paziente di essere malato (o sano) dopo avere eseguito il test (il valore predittivo del test positivo o del test negativo).

Notare inoltre che nella (iii) del post Teorema di Bayes e diagnosi medica:
→ *P(T+|M+)/P(T+)* è una costante e rappresenta l'evidenza fornita dai fatti (nel nostro caso il risultato del test per la diagnosi della malattia)
→ *P(M+)* è l'informazione/probabilità a priori
→ *P(M+|T+)* è l'informazione/probabilità a posteriori

Pertanto il teorema può essere riscritto come

probabilità a posteriori = evidenza · probabilità a priori

espressione che sintetizza l'essenza del teorema di Bayes: che consente di aggiornare l'informazione in nostro possesso, la probabilità a priori, sulla base dell'evidenza fornita dai fatti.

Mentre i calcoli eseguiti impiegando i valori di probabilità sono riportati nell'appendice al termine del post, vediamo qui i calcoli eseguiti impiegando il numero di casi osservati, organizzati in questo modo

	Malattia +	**Malattia -**
Test +	VP	FP
Test -	FN	VN

e definiti come [2]:
→ **veri positivi** (VP), i soggetti malati con il test positivo;
→ **falsi positivi** (FP), i soggetti sani con il test positivo;
→ **falsi negativi** (FN), i soggetti malati con il test negativo;
→ **veri negativi** (VN), i soggetti sani con il test negativo.

Supponiamo ora (primo esempio) che il test per una data malattia sia stato eseguito in 800 soggetti sani, dipendenti di un Ospedale, impiegati come controlli, e in 800 malati, che sono stati raccolti in numero così elevato in quanto nell'Ospedale esiste un Reparto che si occupa specificamente di questa malattia. Questa è la tabella con i risultati del test

	Malattia +	**Malattia -**
Test +	700 (VP)	44 (FP)
Test -	100 (FN)	756 (VN)

I calcoli sono effettuati mediante il pacchetto aggiuntivo **epiR** [3] che deve essere preventivamente scaricato dal **CRAN.** Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# SENSIBILITA', SPECIFICITA' E VALORE PREDITTIVO
#
#
library(epiR) # carica il pacchetto
cells <- c(700,44,100,756) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Test +", "Test -") # genera l'array rnames con i nomi delle righe
cnames <- c("Malattia +", "Malattia -") # genera l'array cnames con i nomi delle colonne
mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames,
cnames)) # costruisce la matrice dati mymatrix a partire dagli array cells, rnames, cnames
mymatrix # mostra la matrice dati mymatrix
mystat <- epi.tests(mymatrix, conf.level=0.95) # calcola le statistiche
summary(mystat) # mostra le statistiche
#
```

Con la funzione **c()** [4] sono generati gli array che mediante l'operatore di assegnamento **<-** vengono denominati **cells**, **rnames**, **cnames** e che ora contengono rispettivamente i dati (**cells**), i nomi delle righe (**rnames**) e i nomi delle colonne (**cnames**).

Quindi mediante la funzione **matrix()** [5] gli array sono combinati nella matrice **mymatrix** di 2 righe (**nrow=2**) per due colonne (**ncol=2**) inserendo i valori per riga (**byrow=TRUE**) e quindi da sinistra a destra e dall'alto in basso ottenendo la matrice desiderata alla quale con l'ultimo argomento (**dimnames=list(rnames, cnames)**) sono assegnati i nomi alle righe e alle colonne.

Questo è il contenuto dell'oggetto **mymatrix** riportato da **R** con il comando **mymatrix**:

```
> mymatrix # mostra la matrice dati mymatrix
       Malattia + Malattia -
Test +        700         44
Test -        100        756
```

Impiegando la funzione **epi.test()** sono infine calcolate sui dati contenuti in **mymatrix** le statistiche, ciascuna con l'intervallo di confidenza al 95% (**conf.level=0.9**5), e i risultati sono riportati con la funzione **summary()**:

```
> summary(mystat) # mostra le statistiche
   statistic          est       lower       upper
1         ap   0.46500000  0.44033136  0.48979728
2         tp   0.50000000  0.47520704  0.52479296
3         se   0.87500000  0.85006839  0.89712668
4         sp   0.94500000  0.92686534  0.95975609
5    diag.ac   0.91000000  0.89490175  0.92357301
6    diag.or 120.27272727 83.14513765 173.97925284
7       nndx   1.21951220  1.16702078  1.28711106
8     youden   0.82000000  0.77693373  0.85688277
9     pv.pos   0.94086022  0.92141771  0.95670377
10    pv.neg   0.88317757  0.85974563  0.90393102
11    lr.pos  15.90909091 11.92293523  21.22792488
12    lr.neg   0.13227513  0.11003359  0.15901245
13    p.rout   0.53500000  0.51020272  0.55966864
14     p.rin   0.46500000  0.44033136  0.48979728
15    p.tpdn   0.05500000  0.04024391  0.07313466
16    p.tndp   0.12500000  0.10287332  0.14993161
17    p.dntp   0.05913978  0.04329623  0.07858229
18    p.dptn   0.11682243  0.09606898  0.14025437
```

Le prime 12 statistiche calcolate sono quelle che ci interessano, riportiamo (tra parentesi) il significato della sigla, le formule con cui sono calcolate e il risultato numerico ottenuto (est) con il limite inferiore (lower) e il limite superiore (upper) del relativo intervallo di confidenza al 95% (le statistiche da 13 a 18 riguardano l'outcome cioè il risultato, e non le consideriamo).

ap (prevalenza apparente, soggetti con il test positivo)
(VP+FP)/(VP+FP+FN+VN) = 0.4650000 (0.4403314 - 0.4897973)

tp (prevalenza reale, soggetti con la malattia)
(VP+FN)/(VP+FP+FN+VN) = 0.5000000 (0.4752070 - 0.5247930)

se (sensibilità, positività nei malati)
VP/(VP+FN) = 0.8750000 (0.8500684 - 0.8971267)

`sp` (specificità, negatività nei sani)
VN/(VN+FP) = `0.9450000` (`0.9268653` - `0.9597561`)

`diag.ac` (accuratezza diagnostica)
(VP+VN)/(VP+FP+FN+VN) = `0.9100000` (`0.8949017` - `0.9235730`)

`diag.or` (odd ratio)
rapporto LR+/LR- = `120.2727273` (`83.1451376` - `173.9792528`)

`nndx` (numero necessario per la diagnosi)
1/(sensibilità - (1 - specificità)) = `1.2195122` (`1.1670208` - `1.2871111`)

`youden` (indice di Youden)
sensibilità + specificità - 1 = `0.8200000` (`0.7769337` - `0.8568828`)

`pv.pos` (valore predittivo di un test positivo)
VP/(VP+FP) = `0.9408602` (`0.9214177` - `0.9567038`)

`pv.neg` (valore predittivo di un test negativo)
VN/(VN+FN) = `0.8831776` (`0.8597456` - `0.9039310`)

`lr.pos` (rapporto di verosimiglianza LR+ per un test positivo) [6]
(VP/(VP+FN))/(FP/(FP+VN)) = `15.9090909` (`11.9229352` - `21.2279249`)

`lr.neg` (rapporto di verosimiglianza LR- per un test negativo) [6]
(FN/(VP+FN))/(VN/(FP+VN)) = `0.1322751` (`0.1100336` - `0.1590125`)

La conclusione è che il valore predittivo di un test positivo, ovvero la probabilità di essere malato per un soggetto al quale il test è risultato positivo, è uguale al 94.1% e il valore predittivo di un test negativo, ovvero la probabilità di essere sano per un soggetto al quale il test è risultato negativo, è uguale all'88.3%.

Nel caso appena esaminato la prevalenza della malattia è del 50%, un valore che è dipeso dalle modalità di raccolta dei dati: per valutare adeguatamente il test è stato raccolto il maggior numero possibile di malati, un fatto che non riflette la reale diffusione della malattia. Ora supponiamo che il test venga applicato per valutare lo stato di salute di un individuo della popolazione generale, nella quale la reale diffusione della malattia, la sua prevalenza, è del 2%. I dati (secondo esempio) sono questi:

	Malattia +	**Malattia -**
Test +	14 (VP)	44 (FP)
Test -	2 (FN)	756 (VN)

Copiate questo script, identico al precedente tranne che per il fatto che contiene i nuovi dati, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# SENSIBILITA', SPECIFICITA' E VALORE PREDITTIVO
#
#
library(epiR) # carica il pacchetto
cells <- c(14,44,2,756) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Test +", "Test -") # genera l'array rnames con i nomi delle righe
cnames <- c("Malattia +", "Malattia -") # genera l'array cnames con i nomi delle colonne
```

mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # costruisce la matrice dati mymatrix a partire dagli array cells, rnames, cnames
mymatrix # mostra la matrice dati mymatrix
mystat <- epi.tests(mymatrix, conf.level=0.95) # calcola le statistiche
summary(mystat) # mostra le statistiche
#

Questi sono i dati contenuti nell'oggetto **mymatrix**

```
> mymatrix # mostra la matrice dati mymatrix
           Malattia + Malattia -
Test +          14          44
Test -           2         756
```

e questi sono, riportati con la funzione **summary()**, i risultati calcolati con la funzione **epi.test()**, ciascuno con l'intervallo di confidenza al 95% (**conf.level=0.95**)

```
> summary(mystat) # mostra le statistiche
   statistic           est         lower         upper
1         ap   0.071078431  0.0544119039   0.090919329
2         tp   0.019607843  0.0112480876   0.031647030
3         se   0.875000000  0.6165237632   0.984486396
4         sp   0.945000000  0.9268653430   0.959756088
5    diag.ac   0.943627451  0.9255197857   0.958436211
6    diag.or 120.272727273 26.5044169172 545.778047894
7       nndx   1.219512195  1.0590499968   1.840301892
8     youden   0.820000000  0.5433891062   0.944242484
9     pv.pos   0.241379310  0.1387012000   0.371689716
10    pv.neg   0.997361478  0.9905015414   0.999680303
11    lr.pos  15.909090909 11.3036587611  22.390907130
12    lr.neg   0.132275132  0.0361754961   0.483661941
13    p.rout   0.928921569  0.9090806715   0.945588096
14     p.rin   0.071078431  0.0544119039   0.090919329
15    p.tpdn   0.055000000  0.0402439118   0.073134657
16    p.tndp   0.125000000  0.0155136038   0.383476237
17    p.dntp   0.758620690  0.6283102838   0.861298800
18    p.dptn   0.002638522  0.0003196972   0.009498459
```

Si nota immediatamente che quando la prevalenza della malattia diminuisce dal 50% al 2% (precisamente diventa 1.96%)

```
                est       lower        upper
tp        0.5000000   0.4752070    0.5247930 [primo esempio]
tp       0.01960784  0.01124809   0.03164703 [secondo esempio]
```

il valore predittivo del test positivo diminuisce passando dal 94.1% al 24.1%

```
                est       lower        upper
pv.pos    0.9408602   0.9214177    0.9567038  [primo esempio]
pv.pos    0.24137931  0.13870120   0.37168972 [secondo esempio]
```

e il valore predittivo del test negativo aumenta passando da 88.3% a 99.7%

```
                est       lower        upper
pv.neg    0.8831776   0.8597456    0.9039310  [primo esempio]
pv.neg    0.99736148  0.99050154   0.99968030 [secondo esempio]
```

Mentre sensibilità e specificità non cambiano al cambiare della popolazione alla quale un test diagnostico viene applicato, in quanto sono caratteristiche intrinseche al test, la prevalenza della malattia non dipende dal test, ma è una caratteristica della popolazione alla quale il test viene applicato, e può cambiare notevolmente. Per esempio la prevalenza della malaria in Italia è diversa da quella che si ha nei paesi dell'Africa equatoriale; la prevalenza della malattia reumatica nei pazienti che accedono all'ambulatorio di un medico generico è differente dalla prevalenza della malattia reumatica nei pazienti ricoverati in un Reparto ospedaliero specializzato nella cura di questa malattia; la prevalenza del cancro al polmone è differente nei soggetti fumatori rispetto ai soggetti non fumatori; e così via. E questo cambia il valore predittivo del test in quanto al diminuire della prevalenza della malattia, cioè della probabilità a priori di malattia, diminuisce il valore predittivo del test positivo (la probabilità a posteriori di malattia in un soggetto cui il test è risultato positivo), e aumenta il valore predittivo del test negativo (la probabilità a posteriori di non-malattia in un soggetto cui il test è risultato negativo) [7].

Questo è un punto cruciale, e rende conto del perché in molte situazioni di bassa prevalenza un test sia utile, e venga impiegato, soprattutto per escludere una malattia. Mentre il teorema di Bayes mostra (e non solo in campo medico) come e quanto l'incertezza delle nostre conclusioni può essere ridotta grazie all'informazione/evidenza fornita dai fatti, e **R** fornisce opportuni strumenti di calcolo.

---

[1] Abbiamo anche visto in un altro post come sovrapporre due Curve ROC sovrapposte.

[2] Gerhardt W, Keller H. *Evaluation of test data from clinical studies*. Scand J Clin Lab Invest, 46, 1986 (supplement 181). URL consultato il 18/06/2019: http://bit.ly/2IVp4ox

[3] *Package 'epiR'*. URL consultato il 25/01/2019: https://goo.gl/wyuNsr

[4] Digitate **help(c)** nella `Console di R` per la documentazione della funzione **c()**.

[5] Digitate **help(matrix)** nella `Console di R` per la documentazione della funzione **matrix**.

[6] Deeks JJ, Altman DG. *Diagnostic tests 4: likelihood ratios*. BMJ 2004;329;168-169. URL consultato il 29/10/2018: https://goo.gl/ahpbpN

[7] Per i temi trattati nei post sul teorema di Bayes vedere Altman, Federspil, Galen, Gerhardt, Gigerenzer, Gill, Motterlini e Nordenström nella pagina Bibliografia.

APPENDICE

Si vuole dimostrare che i risultati ottenuti sopra con i conteggi dei casi sono identici a quelli ottenuti impiegando il teorema di Bayes con i valori di probabilità e le espressioni (iv) per il valore predittivo del test positivo e (v) per il valore predittivo del test negativo (vedi post Teorema di Bayes e diagnosi medica).

Primo esempio riportato sopra: abbiamo sensibilità del test 0.875, specificità del test 0.945 e prevalenza della malattia 0.50.

Valore predittivo del test positivo

$$P(M+|T+) = \frac{P(T+|M+) \cdot P(M+)}{P(T+|M+) \cdot P(M+) + P(T+|M-) \cdot P(M-)} \quad \text{(iv)}$$

che si legge come

$$P(M+|T+) = \frac{sensibilità \cdot prevalenza}{sensibilità \cdot prevalenza + (1 - specificità) \cdot (1 - prevalenza)}$$

quindi il valore predittivo del test positivo è uguale a

$$P(M+|T+) = \frac{0.875 \cdot 0.50}{(0.875 \cdot 0.50) + (0.055 \cdot 0.50)} = 0.9408602$$

Valore predittivo del test negativo

$$P(M-|T-) = \frac{P(T-|M-) \cdot P(M-)}{P(T-|M-) \cdot P(M-) + P(T-|M+) \cdot P(M+)} \text{ (v)}$$

che si legge come

$$P(M-|T-) = \frac{specificità \cdot (1 - prevalenza)}{specificità \cdot (1 - prevalenza) + (1 - sensibilità) \cdot prevalenza}$$

quindi il valore predittivo del test negativo è uguale a

$$P(M-|T-) = \frac{0.945 \cdot 0.50}{(0.945 \cdot 0.50) + (0.125 \cdot 0.50)} = 0.8831776$$

Secondo esempio riportato sopra: abbiamo sensibilità del test 0.875, specificità del test 0.945 e prevalenza della malattia 0.01969784.

Valore predittivo del test positivo

$$P(M+|T+) = \frac{0.875 \cdot 0.01960784}{(0.875 \cdot 0.01960784) + (0.055 \cdot 0.98039216)} = 0.24137928$$

Valore predittivo del test negativo

$$P(M-|T-) = \frac{0.945 \cdot 0.98039216}{(0.945 \cdot 0.98039216) + (0.125 \cdot 0.01960784)} = 0.99736148$$

I risultati del valore predittivo ottenuti impiegando i conteggi dei casi sono identici a quelli ottenuti impiegando il teorema di Bayes con i valori di probabilità, come volevasi dimostrare.

---

mercoledì 18 settembre 2019

Efficienza e accuratezza diagnostica [1]

Nel post Teorema di Bayes e diagnosi medica abbiamo visto che è possibile impiegare il teorema di Bayes per rispondere alle due domande: quale probabilità ha di essere affetto da una specifica malattia un soggetto cui un dato test per quella specifica malattia è risultato positivo? quale probabilità ha di essere sano un soggetto cui il test è risultato negativo?

E abbiamo visto come un modo semplice per farlo si basi sul numero di casi osservati, organizzati come in questa tabella

	Malattia +	Malattia -
Test +	VP	FP
Test -	FN	VN

nella quale sono definiti come:
→ **veri positivi** (VP) i soggetti malati con il test positivo;
→ **falsi positivi** (FP) i soggetti sani con il test positivo;
→ **falsi negativi** (FN) i soggetti malati con il test negativo;
→ **veri negativi** (VN) i soggetti sani con il test negativo.

Nel post Curve ROC e valore soglia abbiamo visto, dati i risultati di un test per la diagnosi di una malattia X, come impiegare il pacchetto **pROC** per tracciare la curva ROC [1] e ricavare il **valore soglia** (Fig. 1d) che consente di classificare correttamente il maggior numero possibile di soggetti sani (VN) e di classificare correttamente il maggior numero possibile di soggetti malati (VP), minimizzando sia la possibilità di sbagliare classificando dei soggetti malati come sani (FN) sia la possibilità di sbagliare classificando dei soggetti sani come malati (FP). Fissato il valore soglia, risultano determinate la **sensibilità** e la **specificità** del test.

Nel post Sensibilità, specificità e valore predittivo abbiamo impiegato il pacchetto **epiR** per calcolare le probabilità che rispondono alle due domande:
→ il **valore predittivo di un test positivo**, ovvero la probabilità di essere malato per un soggetto che presenta un test positivo, che risponde alla prima domanda;
→ il **valore predittivo di un test negativo**, ovvero la probabilità di essere sano per un soggetto che presenta un test negativo, che risponde alla seconda domanda.

In questo post, che conclude la saga bayesiana, impieghiamo **R** per completare quanto visto finora con una analisi numerica e una analisi grafica dettagliate [2] dei dati del file `bayes.csv` [3]. Nel primo passaggio, impiegando il pacchetto **ROCR**, sono calcolati, per ciascuno dei possibili valori soglia compresi tra il valore minimo e il valore massimo osservati, il numero di VP, FP, FN e VN. Nel secondo passaggio, impiegando semplicemente le funzioni base di **R**, sono calcolati e rappresentati graficamente, per ciascuno dei possibili valori soglia compresi tra il valore minimo e il valore massimo osservati, sensibilità, specificità, efficienza diagnostica, valore predittivo del test positivo, valore predittivo del test negativo e accuratezza diagnostica.

Per proseguire è necessario:
→ effettuare il download del file `bayes.csv`
→ salvare il file nella cartella `C:\Rdati\`

Trovate il file, con le istruzioni per scaricare questo e gli altri file di dati impiegati nei post, alla pagina Dati.

Il file contiene la variabile `classe` che indica la classe di appartenenza dell'osservazione/riga (qui vengono impiegati i valori previsti di default nel pacchetto e cioè 0 per indicare un soggetto sano/controllo e 1 per indicare un soggetto malato/caso) e la variabile quantitativa `valore`, che riporta il risultato del test, come illustrato in questo esempio che include le prime cinque e le ultime cinque osservazioni/righe del file:

```
classe;valore
0;19
0;22
0;26
0;28
0;29
.....
1;235
1;237
1;237
1;242
1;242
```

Inoltre il pacchetto aggiuntivo **ROCR** [4] deve essere scaricato dal **CRAN.** Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# EFFICIENZA E ACCURATEZZA DIAGNOSTICA
#
library(ROCR) # carica il pacchetto
mydata <- read.table("c:/Rdati/bayes.csv", header=TRUE, sep=";") # importa i dati
attach(mydata) # consente di impiegare direttamente i nomi delle variabili
#
# calcola sensibilità, specificità, valore predittivo
#
pred <- prediction(valore, classe) # genera mediante la  funzione prediction() un oggetto che contiene una pre-elaborazione dei dati
#
cut <- sort(unlist(pred@cutoffs), decreasing=FALSE) # calcola i valori soglia
vp <- sort(unlist(pred@tp), decreasing=TRUE) # calcola i veri positivi
fp <- sort(unlist(pred@fp), decreasing=TRUE) # calcola i falsi positivi
fn <- sort(unlist(pred@fn), decreasing=FALSE) # calcola i falsi negativi
vn <- sort(unlist(pred@tn), decreasing=FALSE) # calcola i veri negativi
#
tabpred <- data.frame(cut, vp, fp, fn, vn) # genera la tabella con i valori calcolati
names(tabpred) <- c("cutoff", "VP", "FP", "FN", "VN") # assegna i nomi alle variabili/colonne
tabpred # mostra la tabella
#
sens <- vp/(vp+fn) # calcola la sensibilità per ciascun valore soglia
spec <- vn/(vn+fp) # calcola la specificità per ciascun valore soglia
effi <- sens*spec # calcola la efficienza diagnostica per ciascun valore soglia
vptp <- vp/(vp+fp) # calcola il valore predittivo del test positivo per ciascun valore soglia
vptn <- vn/(vn+fn) # calcola il valore predittivo del test negativo per ciascun valore soglia
accu <- (vp+vn)/(vp+fp+fn+vn) # calcola la accuratezza diagnostica per ciascun valore soglia
#
tabprob <- data.frame(cut, sens, spec, effi, vptp, vptn, accu) # genera la tabella con i valori calcolati
names(tabprob) <- c("cutoff", "sensibilità", "specificità", "efficienza", "vptp", "vptn", "accuratezza") # assegna i nomi alle variabili/colonne
tabprob # mostra la tabella
#
```

```
paste("Massima efficienza diagnostica ", round(max(effi), digits=3), " per un valore soglia di ", cut[which.max(effi)], sep="") # riporta la massima efficienza diagnostica e il corrispondente valore soglia
paste("Massima accuratezza diagnostica ", round(max(accu), digits=3), " per un valore soglia di ", cut[which.max(accu)], sep="") # riporta la massima accuratezza diagnostica e il corrispondente valore soglia
#
# traccia istogrammi con curve di sensibilità, specificità ed efficienza diagnostica
#
sani <- subset(valore, classe==0) # sottoinsieme con i valori dei soggetti sani per tracciare gli istogrammi
malati <- subset(valore, classe==1) # sottoinsieme con i valori dei soggetti malati per tracciare gli istogrammi
#
windows() # apre una nuova finestra
par(mar=c(5,4,4,5)+.1) # margini per fare spazio alla scala dell'asse delle y sulla destra
#
hist(malati, xlim=c(0,250), ylim=c(0,150), xaxp = c(0, 250, 5), yaxp = c(0, 150, 3), breaks=seq(from=0, to=250, by=5), border="red", col="red", density = 20, angle = 45, main="Sensibilità, specificità, efficienza diagnostica", xlab="Risultato del test", ylab="Frequenza", freq=TRUE, plot=TRUE) # traccia istogramma malati
hist(sani, xlim=c(0,250), breaks=seq(from=0, to=250, by=5), border="green4", col="lightgreen", density = 20, angle = -45, freq=TRUE, add=TRUE, plot=TRUE) # sovrappone istogramma sani
#
par(new=TRUE) # sovrappone curva sensibilità
plot(cut, sens, xlim=c(0,250), ylim=c(0,1), yaxp=c(0,1,5), type="l", lty=1, lwd=2, pch=20, cex=0.5, col="blue", xaxt="n", yaxt="n", xlab="", ylab="") # traccia la curva
axis(4) # riporta un secondo asse delle y sulla destra per i valori di probabilità
mtext("Probabilità p", side=4, line=3) # aggiunge la legenda all'asse delle y di destra
#
par(new=TRUE) # sovrappone curva specificità
plot(cut, spec, xlim=c(0,250), ylim=c(0,1), yaxp=c(0,1,5), type="l", lty=1, lwd=2, pch=20, cex=0.5, col="mediumorchid", xaxt="n", yaxt="n", xlab="", ylab="") # traccia la curva
#
par(new=TRUE) # sovrappone curva efficienza diagnostica
plot(cut, effi, xlim=c(0,250), ylim=c(0,1), yaxp=c(0,1,5), type="l", lty=3, lwd=2, pch=20, cex=0.5, col="orange", xaxt="n", yaxt="n", xlab="", ylab="") # traccia la curva
#
legend(140, 0.97, legend=c("Sani", "Malati", "Sensibilità", "Specificità", "Efficienza diagnostica"), col=c("green", "red", "blue", "mediumorchid", "orange"), lty=c(1,1,1,1,3), lwd=c(1,1,2,2,2), cex=0.7) # aggiunge al grafico la legenda
#
# traccia istogrammi con curve di valore predittivo ed accuratezza diagnostica
#
windows() # apre una nuova finestra
par(mar=c(5,4,4,5)+.1) # margini per fare spazio alla scala dell'asse delle y sulla destra
#
hist(malati, xlim=c(0,250), ylim=c(0,150), xaxp = c(0, 250, 5), yaxp = c(0, 150, 3), breaks=seq(from=0, to=250, by=5), border="red", col="red", density = 20, angle = 45, main="Valore predittivo del test, accuratezza diagnostica", xlab="Risultato del test", ylab="Frequenza", freq=TRUE, plot=TRUE) # traccia istogramma malati
hist(sani, xlim=c(0,250), breaks=seq(from=0, to=250, by=5), border="green4", col="lightgreen", density = 20, angle = -45, freq=TRUE, add=TRUE, plot=TRUE) # sovrappone istogramma sani
```

```
#
par(new=TRUE) # sovrappone curva valore predittivo del test positivo
plot(cut, vptp, xlim=c(0,250), ylim=c(0,1), yaxp=c(0,1,5), type="l", lty=1, lwd=2, pch=20, cex=0.5, col="blue", xaxt="n", yaxt="n", xlab="",ylab="") # traccia la curva
axis(4) # riporta l'asse delle y sulla destra
mtext("Probabilità p", side=4, line=3) # aggiunge legenda asse y
#
par(new=TRUE) # sovrappone curva valore predittivo del test negativo
plot(cut, vptn, xlim=c(0,250), ylim=c(0,1), yaxp=c(0,1,5), type="l", lty=1, lwd=2, pch=20, cex=0.5, col="mediumorchid", xaxt="n", yaxt="n", xlab="", ylab="") # traccia la curva
#
par(new=TRUE) # sovrappone curva accuratezza diagnostica
plot(cut, accu, xlim=c(0,250), ylim=c(0,1), yaxp=c(0,1,5), type="l", lty=3, lwd=2, pch=20, cex=0.5, col="orange", xaxt="n", yaxt="n", xlab="", ylab="") # traccia la curva
#
legend(135, 0.98, legend=c("Sani", "Malati", "Valore predittivo del test +", "Valore predittivo del test -", "Accuratezza diagnostica"), col=c("green", "red", "blue", "mediumorchid", "orange"), lty=c(1,1,1,1,3), lwd=c(1,1,2,2,2), cex=0.7) # aggiunge al grafico la legenda
#
```

Importare ed elaborare i vostri dati con questo script risulterà semplicissimo: dovete solamente strutturare i dati come nel file `bayes.csv` [5] quindi modificare opportunamente il nome e la posizione del file nella prima riga di codice.

Con le prime tre righe di codice viene caricato il pacchetto, sono importati i dati, e viene reso possibile con la funzione **attach()** impiegare direttamente i nomi delle variabili.

La riga di codice cruciale è la quarta, che salva nell'oggetto **pred** i dati generati dalla funzione **prediction()** del pacchetto ROCR e precisamente il vettore che contiene tutti i possibili valori soglia (**pred@cutoffs**) e i vettori che riportano per ciascun valore soglia:
→ i veri positivi (**pred@tp**)
→ i falsi positivi (**pred@fp**)
→ i falsi negativi (**pred@fn**)
→ i veri negativi (**pred@tn**).

Questi cinque vettori sono quindi combinati nella tabella **tabpred** che così vi apparirà:

```
    cutoff  VP  FP  FN  VN
1       19 843 853   0   0
2       22 843 852   0   1
3       26 843 851   0   2
4       28 843 850   0   3
          ..........
49      73 771  55  72 798
50      74 766  48  77 805
51      75 763  44  80 809
52      76 760  42  83 811
53      77 755  39  88 814
          .........
200    235   6   0 837 853
201    237   4   0 839 853
202    242   2   0 841 853
203    Inf   0   0 843 853
```

Pertanto fissando ad esempio il valore soglia (cutoff) tra sani e malati a `75` avremo `763 VP`, `44 FP`, `80 FN` e `809 VN`, e così via.

Nelle righe successive a partire dai vettori **vp**, **fp**, **fn** e **vn** sono calcolati per ciascuno dei valori soglia (cutoff) individuati:
→ sensibilità = VP/(VP+FN)
→ specificità = VN/(VN+FP)
→ efficienza diagnostica = sensibilità · specificità
→ valore predittivo del test positivo = VP/(VP+FP)
→ valore predittivo del test negativo = VN/(VN+FN)
→ accuratezza diagnostica = (VP+VN)/(VP+FP+FN+VN)
quindi i vettori **sens**, **spec**, **effi**, **vptp**, **vptn**, **accu** che contengono i dati calcolati sono combinati nella tabella **tabprob**:

```
    cutoff sensibilità specificità  efficienza      vptp      vptn accuratezza
1       19 1.000000000 0.000000000 0.000000000 0.4970519       NaN   0.4970519
2       22 1.000000000 0.001172333 0.001172333 0.4973451 1.0000000   0.4976415
3       26 1.000000000 0.002344666 0.002344666 0.4976387 1.0000000   0.4982311
4       28 1.000000000 0.003516999 0.003516999 0.4979327 1.0000000   0.4988208
                                    .........
49      73 0.914590747 0.935521688 0.855619480 0.9334140 0.9172414   0.9251179
50      74 0.908659549 0.943728019 0.857527476 0.9410319 0.9126984   0.9262972
51      75 0.905100830 0.948417351 0.858413331 0.9454771 0.9100112   0.9268868
52      76 0.901542112 0.950762016 0.857151996 0.9476309 0.9071588   0.9262972
53      77 0.895610913 0.954279015 0.854662700 0.9508816 0.9024390   0.9251179
                                    .........
200    235 0.007117438 1.000000000 0.007117438 1.0000000 0.5047337   0.5064858
201    237 0.004744958 1.000000000 0.004744958 1.0000000 0.5041371   0.5053066
202    242 0.002372479 1.000000000 0.002372479 1.0000000 0.5035419   0.5041274
203    Inf 0.000000000 1.000000000 0.000000000       NaN 0.5029481   0.5029481
```

Da questa tabella sono ricavati il valore soglia che corrisponde alla massima efficienza diagnostica

```
[1] "Massima efficienza diagnostica 0.858 per un valore soglia di 75"
```

il valore soglia, identico al precedente, che corrisponde alla massima accuratezza diagnostica

```
[1] "Massima accuratezza diagnostica 0.927 per un valore soglia di 75"
```

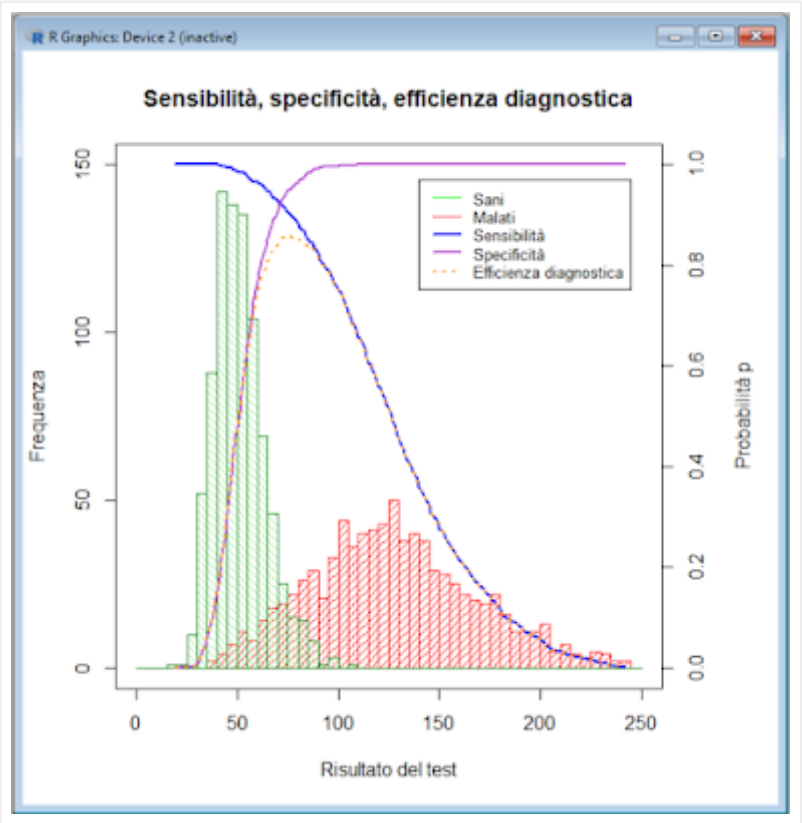
e i valori delle grandezze (sensibilità, specificità, valori predittivi) corrispondenti al valore soglia 75 così individuato.

Da notare nelle funzioni **paste()** che generano le due righe gli argomenti **cut[which.max(effi)]** e **cut[which.max(accu)]** nei quali la funzione **which.max()** restituisce la posizione, all'interno del rispettivo vettore, del valore massimo di efficienza o di accuratezza, posizione che viene impiegata per trovare nel vettore **cut** il valore soglia corrispondente. Scorrendo i dati della tabella è facile confermare che sia nel caso dell'efficienza diagnostica sia nel caso della accuratezza diagnostica il valore soglia "ottimale" tra sani e malati è quello che corrisponde al massimo valore trovato di efficienza o di accuratezza.

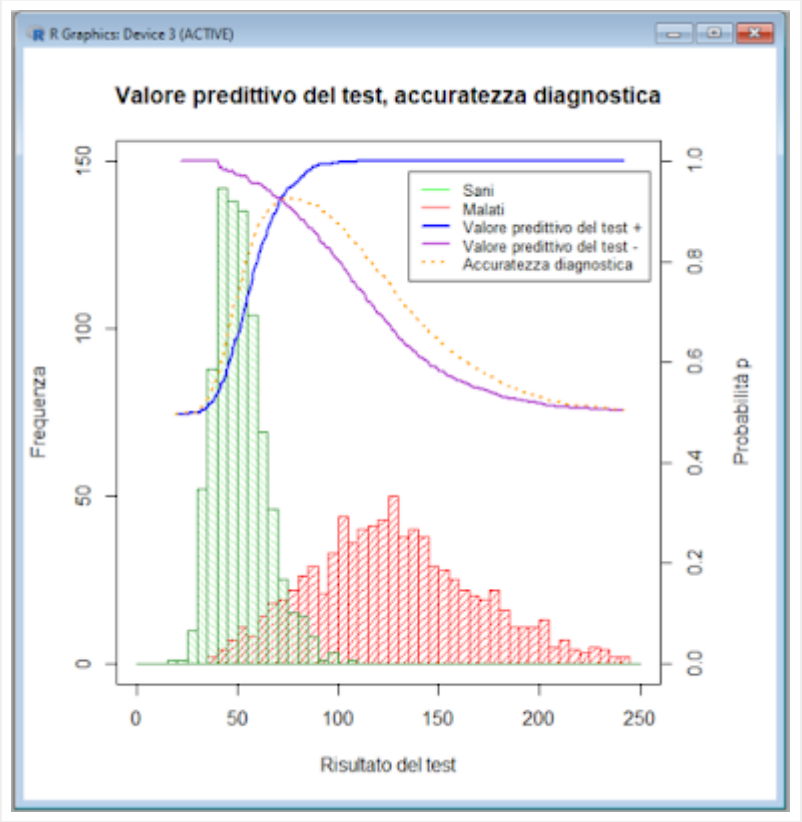
Dalla tabella sono infine generati i grafici che riportano le sei grandezze calcolate in funzione del valore soglia (cutoff). In questa parte l'unica cosa un po' particolare è la funzione **subset()** [6] che viene impiegata per generare, dai dati importati, due sottoinsiemi separati, uno contenente solamente i valori dei soggetti sani e l'altro contenente solamente i valori dei soggetti malati, impiegando come argomenti:
→ il nome della variabile (**valore**) dalla quale si desidera estrarre il sottoinsieme di valori;

→ il criterio di selezione dei casi da estrarre, laddove **classe==0** indica di estrarre i valori per i quali nella variabile **classe** è riportato 0 (sano) e laddove **classe==1** indica di estrarre i valori per i quali nella variabile **classe** è riportato 1 (malato).

Le funzioni e gli argomenti impiegati per costruire gli istogrammi con le distribuzioni dei risultati nel test nei sani e nei malati, e per sovrapporre ad essi le curve che rappresentano, in funzione dei valori soglia riportati in ascisse, la sensibilità, la specificità e l'efficienza diagnostica, sono illustrati sia nella documentazione di R delle funzioni **hist()** e **plot()** sia nei vari post dedicati alla rappresentazione grafica dei dati [7], ai quali si rimanda. Questo è il grafico risultante



Questo invece è il grafico che sovrappone agli istogrammi delle distribuzioni dei risultati nel test nei sani e nei malati le curve che rappresentano, in funzione dei valori soglia riportati in ascisse, il valore predittivo del test positivo, il valore predittivo del test negativo e l'accuratezza diagnostica



La conclusione è che un'analisi completa e dettagliata, sia numerica sia grafica, delle caratteristiche di un test per la diagnosi di una malattia, esempio paradigmatico di applicazione del teorema di Bayes, può essere realizzata a partire semplicemente dai conteggi di VP, FP, FN e VN effettuati per una serie opportuna di valori soglia, come magistralmente illustrato da Gerhardt e da Keller oltre trent'anni fa [2].

Per una versione lievemente più compatta del codice, che tabula in modo più compatto i dati e che impiega al posto degli istogrammi i kernel density plot, vedere nell'Indice il post Efficienza e accuratezza diagnostica [2].

Se strutturate i vostri dati seguendo le semplici indicazioni riportate sopra per il file bayes.csv per riutilizzare lo script dovete solamente, nella prima riga di codice, sostituire **c:/Rdati/bayes.csv** con posizione e nome del vostro file di dati.

---

[1] Abbiamo anche visto in un altro post come realizzare due Curve ROC sovrapposte.

[2] Gerhardt W, Keller H. *Evaluation of test data from clinical studies*. Scand J Clin Lab Invest, 46, 1986 (supplement 181). URL consultato il 18/06/2019: http://bit.ly/2IVp4ox

[3] L'esempio qui impiegato rappresenta la situazione più frequente, quella nella quale la malattia determina un aumento del risultato dei test, ma ovviamente è sufficiente ribaltare dati e conclusioni per riadattare il tutto a un caso nel quale la malattia determina una diminuzione del risultato del test.

[4] *Package 'ROCR'*. URL consultato il 15/06/2019: http://bit.ly/2MOmUvT

[5] Vedere il post Curve Roc e valore soglia.

[6] Digitate **help(subset)** nella `Console di R` per la documentazione della funzione **subset()**.

[7] Per la funzione **legend()** consultare anche il post Aggiungere la legenda a un grafico.

giovedì 3 ottobre 2019

Analisi dei gruppi (clustering gerarchico e non gerarchico)

L'obiettivo dell'**analisi dei gruppi** (*cluster analysis* o *clustering*) è concettualmente semplice: verificare la possibile esistenza, in un insieme di oggetti, di sottoinsiemi di oggetti particolarmente simili tra loro (gruppi/cluster).

L'analisi dei gruppi si applica a *dati multivariati* ed è un metodo statistico di tassonomia numerica che riveste un ruolo importante nella *analisi esplorativa dei dati*.

Nonostante alla base dell'analisi dei gruppi vi sia un'idea semplice e logica, vi sono numerosi modi per realizzarla, ed esistono:
→ *metodi gerarchici*, che danno luogo a una suddivisione ad albero (dendrogramma) in base alla distanza tra i singoli oggetti dell'insieme;
→ *metodi non gerarchici*, nei quali l'appartenenza di un oggetto (dell'insieme) ad uno specifico sottoinsieme/gruppo/cluster viene stabilita sulla base della sua distanza dal centro o dalla media dei dati o da un punto rappresentativo del cluster [1];
→ *metodi agglomerativi* (o *metodi bottom-up*) nei quali all'inizio del processo di classificazione ad ogni oggetto viene fatto corrispondere un cluster. In questo stadio gli oggetti sono considerati tutti dissimili tra di loro. Al passaggio successivo i due oggetti più simili sono raggruppati nello stesso cluster. Il numero dei cluster risulta quindi pari al numero di oggetti diminuito di uno. Il procedimento viene ripetuto ciclicamente, fino ad ottenere (all'ultimo passaggio) un unico cluster;
→ *metodi divisivi* (o *metodi top-down*) nei quali inizialmente tutti gli oggetti sono considerati come appartenenti ad un unico cluster, che viene via via suddiviso in cluster fino ad avere un numero di cluster uguale al numero degli oggetti;
→ *metodi esclusivi*, che prevedono che un oggetto possa appartenere esclusivamente a un cluster;
→ *metodi non esclusivi* (*fuzzy*) che prevedono che un oggetto possa appartenere, in modo quantitativamente diverso, a più di un cluster.

Da notare infine che non esiste una regola per stabilire il numero di cluster in cui sono aggregati gli oggetti, che va deciso caso per caso, cosa che complica ulteriormente lo scenario.

Per una trattazione completa dei diversi metodi di clustering, tutti metodi esclusivi tranne i metodi fuzzy, rimando ai post:
→ Analisi dei gruppi (clustering gerarchico) [include il confronto tra dendrogrammi]
→ Analisi dei gruppi (clustering non gerarchico) [include metodi agglomerativi e metodi divisivi]
→ Analisi dei gruppi (clustering non esclusivo) [fuzzy clustering]

Qui riporto una breve sintesi con i due metodi più frequentemente impiegati, e capostipiti delle due alternative più tradizionali nel campo del clustering, entrambi *metodi esclusivi*:
→ il **metodo di Ward**, un *metodo gerarchico* che misura la somiglianza di due oggetti/punti sulla base della loro *distanza euclidea*;
→ il **metodo di MacQueen**, un *metodo non gerarchico* che misura l'appartenenza di un oggetto/punto ad uno specifico cluster con l'algoritmo delle *k-means*.

Come dati impieghiamo i valori di BMI (indice di massa corporea) rilevati a livello europeo alcuni anni fa e pubblicati dall'Istat [2].

Per proseguire è necessario:
→ effettuare il download del file di dati `bmi.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un ↵ `Invio` al termine dell'ultima riga e salvarli

in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

```
Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7
Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2
```

Inoltre è necessario scaricare dal **CRAN** il pacchetto aggiuntivo **cluster** [3]. Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ANALISI DEI GRUPPI (CLUSTERING GERARCHICO E NON GERARCHICO)
#
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # standardizza le variabili
cbind(mydata, z) # mostra i dati originali e i rispettivi valori z
#
# clustering gerarchico con il metodo di Ward (distanza euclidea)
#
mat <- hclust(dist(z, method="euclidean"), method="ward.D") # genera la matrice delle
distanze euclidee e costruisce i cluster
plot(mat, main="Dendrogramma", xlab="BMI nei Paesi europei", sub="", ylab="Distanza
nei valori di BMI dei cluster", cex=0.7) # traccia il dendrogramma
rect.hclust(mat, k=4, border=c("red","blue","green","goldenrod")) # evidenzia i 4
gruppi/cluster
#
# clustering non gerarchico con il metodo di MacQueen (k-means )
#
library(cluster) # carica il pacchetto
windows() # apre e inizializza una nuova finestra grafica
myclust <- kmeans(z, 4, algorithm = c("MacQueen"), nstart=50) # genera i 4 gruppi/cluster
```

clusplot(z, myclust$cluster, color=TRUE, col.clus=c("blue", "goldenrod", "red", "green"), shade=FALSE, labels=3, lines=0, sub="", main="Grafico dei cluster", xlab="Componente principale 1", ylab="Componente principale 2", cex=0.6, col.txt="black", col.p="black") # traccia il grafico dei cluster per le prime due componenti principali
#

Con la prima riga di codice sono importati i dati nell'oggetto **mydata**.

Con la funzione **scale()** della seconda riga per ciascun dato viene calcolata la *deviata normale standardizzata z*. In pratica questa funzione prima calcola per i dati di ciascuna colonna/variabile la `media` e la `deviazione standard`, poi calcola per ciascuno dato `x` la corrispondente deviata normale standardizzata `z` come

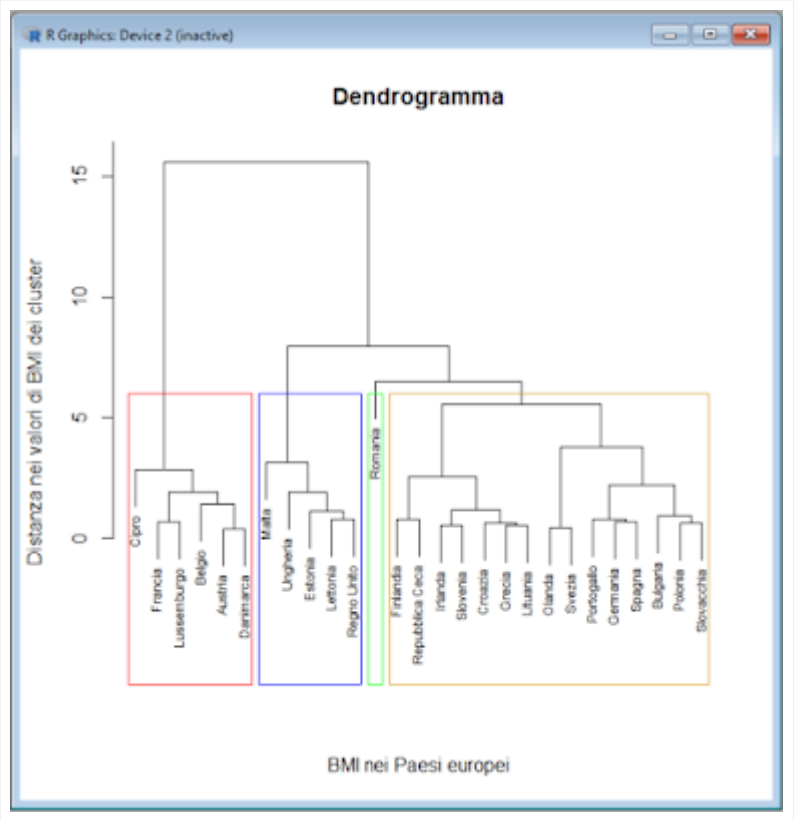
```
z = (x - media) / deviazione standard
```

I valori di `z` sono poi salvati nel nuovo oggetto qui denominato, per comodità mnemonica, **z**.

Nella terza riga infine i dati originali e i corrispondenti valori di deviata normale standardizzata `z` sono mostrati insieme, combinati con la funzione **cbind()**.

Dopo questi preliminari si passa al **clustering gerarchico con il metodo di Ward** con la funzione **hclust()** [5] e due soli argomenti:
→ il primo è costituito dalla matrice delle distanze calcolata con la funzione **dist()** sui dati standardizzati `z` impiegando per il calcolo delle distanze il metodo di euclideo (**method="euclidean"**), che prevede di misurare la distanza tra due punti con il teorema di Pitagora. I metodi che si possono impiegare in alternativa includono: "*euclidean*", "*maximum*", "*manhattan*", "*canberra*", "*binary*", "*minkowski*" [6];
→ il secondo è l'argomento **method=ward.D** che specifica come costruire i cluster, e può assumere in alternativa uno dei seguenti valori: "*ward.D*", "*ward.D2*", "*single*", "*complete*", "*average*" (= *UPGMA*), "*mcquitty*" (= *WPGMA*), "*median*" (= *WPGMC*), "*centroid*" (= *UPGMC*).

A partire dalla matrice delle distanze euclidee salvata nell'oggetto **mat**, con la funzione **plot()** viene tracciato il dendrogramma e con la funzione **rect.hclust()** e l'argomento **k=** viene stabilito in **4** il numero dei gruppi da evidenziare nel dendrogramma.



Per il **clustering non gerarchico con il metodo di MacQueen** dopo avere caricato il pacchetto **cluster** e aperto una nuova finestra grafica con **windows()**, impiegando la funzione **kmeans()** [7] viene generata la matrice delle distanze che viene salvata nell'oggetto **myclust**.

Viene poi impiegata per tracciare il grafico la funzione **clusplot()** nella quale si fanno notare i seguenti argomenti:
→ **color=TRUE** consente l'impiego dei colori nei dati riportati sul grafico;
→ **col.clus** nel quale la sequenza dei colori è stata adattata manualmente per avere gli stessi colori del dendrogramma;
→ **shade** che se posto **=TRUE** riporta all'interni dei cluster un tratteggio con una densità uguale al numero di punti inclusi nel cluster diviso per l'area dell'ellisse;
→ **labels=3** che è uno dei possibili valori per le etichette da riportare (vederli con **help(clusplot)**);
→ **lines=0** che dice di non riportare le lineee che collegano i cluster;
→ **sub=""** che elimina il sottotitolo previsto di default dalla funzione;
→ **cex=0.6** che riduce la dimensione dei caratteri dei nomi delle nazioni riportati all'interno dei cluster;
→ **col.txt** che definisce il colore del testo che compare all'interno dei cluster;
→ **col.p** che definisce il colore impiegato per rappresentare i punti all'interno dei cluster.

Da notare che se nella `Console di R` digitate

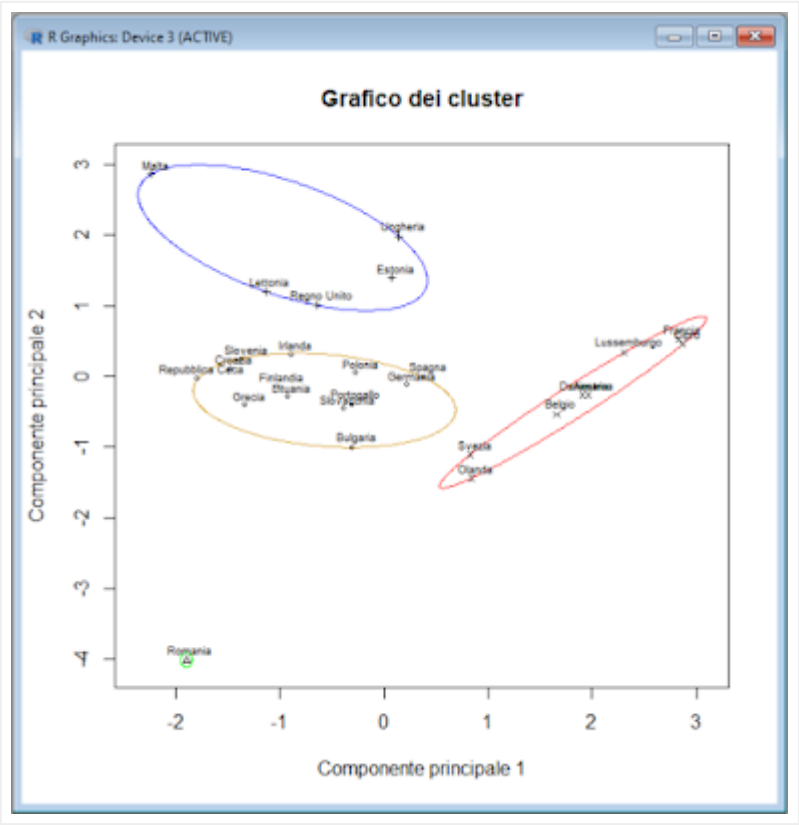
myclust$cluster

viene mostrato il vettore con l'elenco delle nazioni, ciascuna con il cluster cui appartiene

```
> myclust$cluster
      Austria       Belgio     Bulgaria        Cipro      Croazia
            4            4            2            4            2
   Danimarca      Estonia    Finlandia     Francia     Germania
            4            3            2            4            2
       Grecia      Irlanda     Lettonia     Lituania  Lussemburgo
            2            2            3            2            4
```

```
          Malta          Olanda        Polonia     Portogallo     Regno Unito
              3               4               2              2               3
Repubblica Ceca         Romania      Slovacchia       Slovenia         Spagna
              2               1               2              2               2
         Svezia        Ungheria
              4               3
```

Questo infine è il grafico dei cluster generati.



Per un confronto contenente anche il dettaglio numerico i dati originali forniti dall'Istat sono stati importati in un foglio elettronico nel quale sono stati ordinati e suddivisi manualmente impiegando i seguenti criteri:
→ è stato identificato un paese che si discosta in modo significativo da tutti gli altri essendo l'unico con una quota di obesi inferiore al 10% e l'unico con una quota di soggetti sovrappeso superiore al 40%;
→ sono stati raccolti in un unico gruppo i paesi con una percentuale di soggetti di peso normale superiore del 5% circa alla media dei restanti paesi, e precisamente superiore al 47%.
→ sono stati raggruppati insieme tutti i paesi rimanenti, distinguendo tra quelli con una quota di obesi uguale o superiore al 10% e inferiore al 20% e quelli con una quota di obesi uguale o superiore al 20% e inferiore al 30%.

Dalla tabella così costruita, che riconduce i risultati del clustering ai dati originari, risulta evidente che nel cluster rosso predomina il peso normale, nel cluster blu è dominante la categoria obeso, nel cluster giallo è dominante la presenza di sovrappeso, mentre un caso a sé stante è quello della Romania, con valori estremi sia nel sovrappeso (percentuale di casi molto elevata) sia nell'obeso (percentuale di casi molto bassa).

	A	B	C	D	E
1	Nazione	sottopeso	normale	sovrappeso	obeso
2	Danimarca	2.2	50.0	32.9	14.9
3	Austria	2.4	49.6	33.3	14.7
4	Francia	3.2	49.6	31.9	15.3
5	Lussemburgo	2.8	49.3	32.4	15.6
6	Olanda	1.6	49.0	36.0	13.3
7	Svezia	1.8	48.3	35.9	14.0
8	Belgio	2.7	48.0	35.3	14.0
9	Cipro	3.9	47.8	33.8	14.5
10	Malta	2.0	37.0	35.0	26.0
11	Lettonia	1.7	41.8	35.2	21.3
12	Ungheria	2.9	41.9	34.0	21.2
13	Estonia	2.2	43.9	33.5	20.4
14	Regno Unito	2.1	42.2	35.6	20.1
15	Repubblica Ceca	1.1	42.1	37.6	19.3
16	Slovenia	1.6	41.8	37.4	19.2
17	Irlanda	1.9	42.3	37.0	18.7
18	Croazia	1.9	40.7	38.7	18.7
19	Finlandia	1.2	44.1	36.4	18.3
20	Lituania	1.9	42.5	38.3	17.3
21	Grecia	1.9	41.3	39.4	17.3
22	Polonia	2.4	42.9	37.5	17.2
23	Germania	1.8	46.1	35.2	16.9
24	Spagna	2.2	45.4	35.7	16.7
25	Portogallo	1.8	44.6	36.9	16.6
26	Slovacchia	2.1	43.6	38.0	16.3
27	Bulgaria	2.2	43.8	39.2	14.8
28	Romania	1.3	42.9	46.4	9.4

Questa analisi, semplice ma di buon senso, coincide con quella fornita dai due metodi di clustering, a parte il fatto che il metodo di MacQueen include Svezia e Olanda nel cluster normale cerchiato in rosso mentre il metodo di Ward li include nel cluster sovrappeso cerchiato in giallo .

In conclusione vale la pena di notare alcune cose:
→ la standardizzazione, intesa come trasformazione dei dati nelle corrispondenti deviate normali standardizzate è d'obbligo per i metodi non gerarchici mentre per i metodi gerarchici potrebbe non essere necessaria - ma lo potrebbe essere in alcuni casi. Questo però apre un tema che va al di la dei limiti di questo post: qui è stata impiegata la standardizzazione, che personalmente ritengo opportuno eseguire sempre;
→ metodi di clustering che impiegano differenti criteri per il raggruppamento dei dati possono fornire risultati diversi;
→ la presenza di oggetti a cavallo tra due cluster (come qui accade per Svezia e Olanda) potrebbe deporre a favore dell'impiego del clustering con metodi non esclusivi (fuzzy) [8] assegnando gli oggetti contemporaneamente a tutti i cluster in modo quantitativamente diverso;
→ non esiste una "regola aurea" per stabilire il numero di cluster in cui aggregare gli oggetti;
→ anche se matematicamente definito, quindi esente da arbitrarietà, qualsiasi metodo di classificazione risente degli assunti che inevitabilmente devono essere posti alla sua base;
→ nel caso di dati non eccessivamente complessi una loro attenta disamina effettuata mediante strumenti tradizionali può essere di aiuto nella interpretazione dei risultati del clustering.

---

[1] Massart DL, Vandeginste BGM, Deming SN, Michotte Y, Kaufman L. *Chemometrics: a textbook*. Elsevier, New York, 1988, ISBN 0-444-42660-4, pp. 371-384.

[2] Vedere il post Indice di massa corporea (BMI).

[3] Vedere il manuale di riferimento del pacchetto *Package 'cluster'*. URL consultato il 27/09/2019: http://bit.ly/2neaTna

[4] Digitate **help(scale)** nella `Console di R` per la documentazione della funzione **scale()**.

[5] Digitate **help(hclust)** nella `Console di R` per la documentazione della funzione **hclust()**.

[6] Digitate **help(dist)** nella `Console di R` per vedere la documentazione della funzione **dist()**.

[7] Digitate **help(kmeans)** nella `Console di R` per vedere il significato degli argomenti impiegati.

[8] Vedere il post Analisi dei gruppi (clustering non esclusivo).

domenica 1 dicembre 2019

Inserire più grafici nella stessa immagine

Vediamo come sia possibile disporre in vari modi più grafici all'interno di una stessa immagine impiegando una sola riga di codice.

Per farlo ci serviamo delle funzioni **par()**, **layout()** e **matrix()** [1] impiegando come esempi alcuni grafici realizzati con la funzione **boxplot()** [2].

Il codice impiega solamente le *funzioni grafiche di base* di **R**. I dati sono contenuti nella tabella **ais** del pacchetto **DAAG** - accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [3].

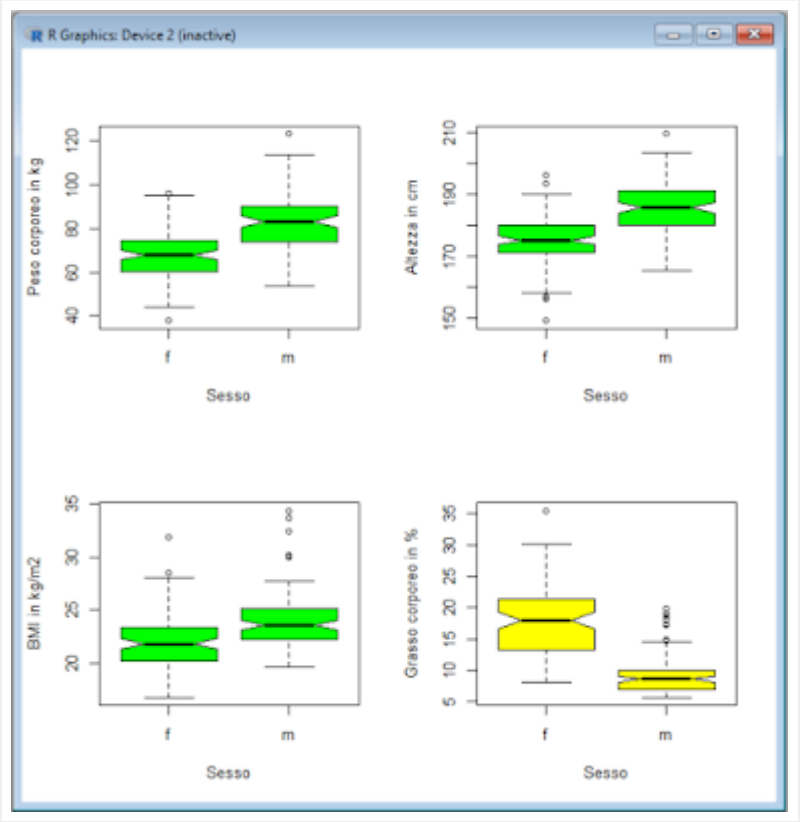
Copiate lo script che segue, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# INSERIRE PIU' GRAFICI NELLA STESSA IMMAGINE
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # quattro grafici in due righe per due colonne
#
boxplot(ais$wt~ais$sex, xlab="Sesso", ylab="Peso corporeo in kg", notch=TRUE,
col="green") # [1] valori del peso corporeo aggregati per sesso
#
boxplot(ais$ht~ais$sex, xlab="Sesso", ylab="Altezza in cm", notch=TRUE, col="green") #
[2] valori dell'altezza aggregati per sesso
#
boxplot(ais$bmi~ais$sex, xlab="Sesso", ylab="BMI in kg/m2", notch=TRUE,
col="green") # [3] valori dell'indice di massa coroporea (BMI) aggregati per sesso
#
boxplot(ais$pcBfat~ais$sex, xlab="Sesso", ylab="Grasso corporeo in %", notch=TRUE,
col="yellow") # [4] valori della percentuale di grasso corporeo aggregati per sesso
#
```

In questo primo script con **par(mfrow=c(2,2))** è stata predisposta la suddivisione della finestra grafica in **2** righe per **2** colonne, quindi in quattro quadranti,

1	2
3	4

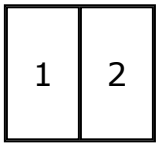
nei quali sono inseriti, procedendo da sinistra verso destra e dall'alto verso il basso, i quattro grafici realizzati con la funzione **boxplot()** sui dati separati per sesso (**~ais$sex**) della tabella **ais** - impiegando l'argomento **notch=TRUE** per riportare sui lati dei box le tacche (**notch**) o incisure che rappresentano i limiti di confidenza al 95% della mediana - con questo risultato.



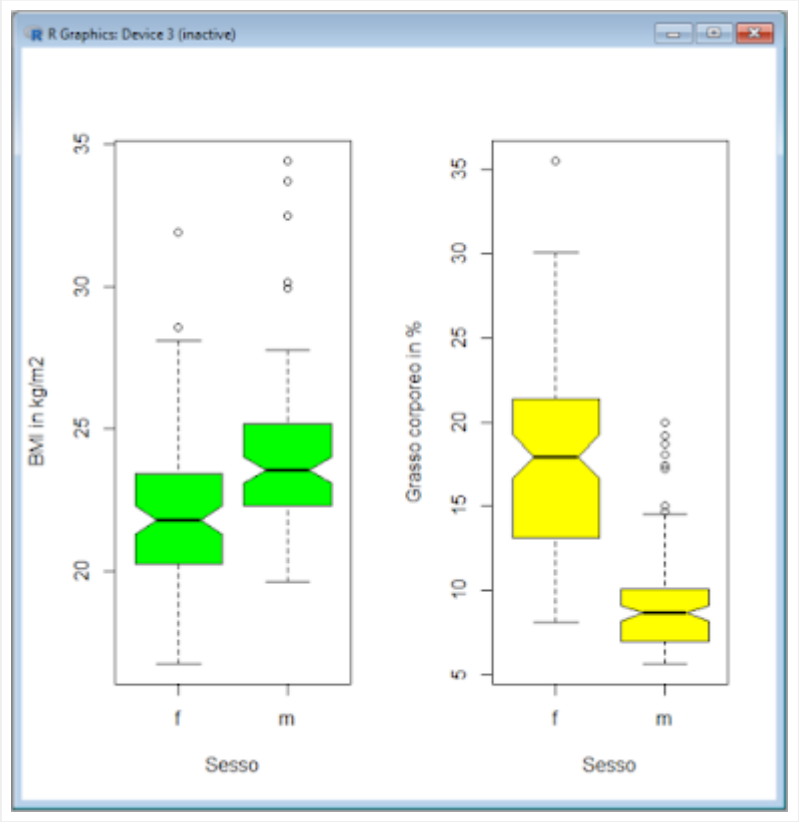
Ora copiate lo script che segue, incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# INSERIRE PIU' GRAFICI NELLA STESSA IMMAGINE
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(1,2)) # due grafici in una riga e due colonne
#
boxplot(ais$bmi~ais$sex, xlab="Sesso", ylab="BMI in kg/m2", notch=TRUE,
col="green") # [1] boxplot per sesso dell'indice di massa corporea (BMI)
#
boxplot(ais$pcBfat~ais$sex, xlab="Sesso", ylab="Grasso corporeo in %", notch=TRUE,
col="yellow") # [2] boxplot per sesso della percentuale di grasso corporeo
#
```

In questo secondo script con **par(mfrow=c(1,2))** è stata predisposta la suddivisione della finestra in **1** riga e **2** colonne



per realizzare due grafici affiancati, il primo sulla sinistra e il secondo sulla destra, con questo risultato.



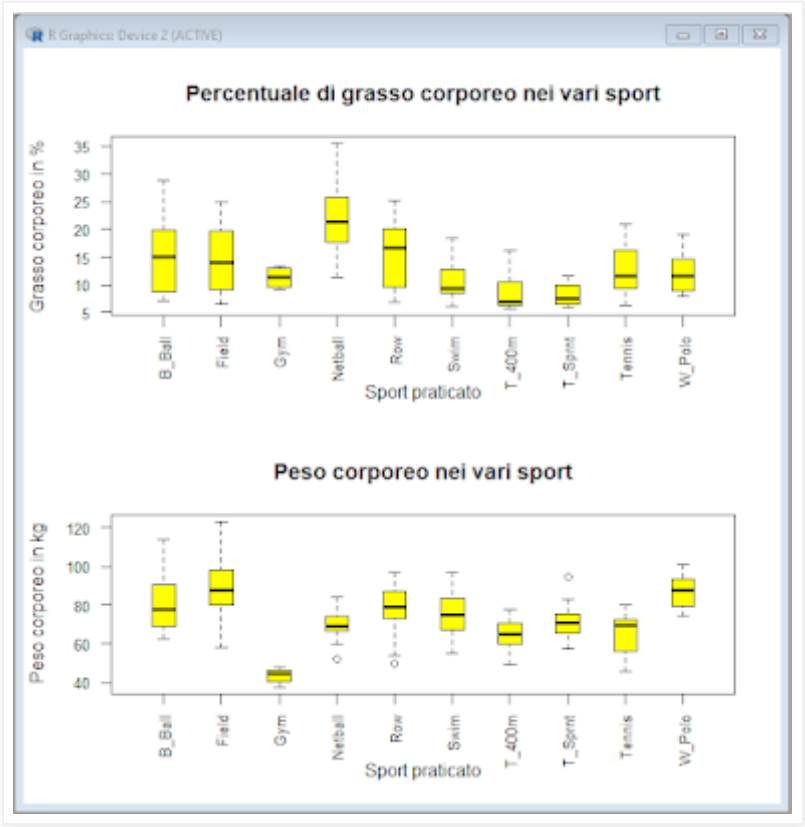
Copiate questo terzo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# INSERIRE PIU' GRAFICI NELLA STESSA IMMAGINE
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,1)) # due grafici in una colonna e due righe
#
boxplot(ais$pcBfat~ais$sport, horizontal=FALSE, boxwex=0.4, cex.axis=0.8, las=2,
main="Percentuale di grasso corporeo nei vari sport", xlab="Sport praticato",
ylab="Grasso corporeo in %", notch=FALSE, col="yellow") # [1] valori della percentuale di
grasso corporeo aggregati per sport
#
boxplot(ais$wt~ais$sport, horizontal=FALSE, boxwex=0.4, cex.axis=0.8, las=2,
main="Peso corporeo nei vari sport", xlab="Sport praticato", ylab="Peso corporeo in kg",
notch=FALSE, col="yellow") # [2] valori del peso corporeo aggregati per sport
#
```

In questo caso con **par(mfrow=c(2,1))** è stata predisposta la suddivisione della finestra in **2** righe e **1** colonna - riportando i boxplot in verticale (**horizontal=FALSE**), le etichette delle ascisse in verticale (**las=2**), riducendo la larghezza dei box (**boxwex=0.4**) e senza le incisure (**notch=FALSE**) che rappresentano i limiti di confidenza al 95% della mediana - per posizionare i due grafici uno sopra l'altro in questo modo

1
2

con questo risultato.



Potete anche aumentare il numero delle righe e/o delle colonne, prestando ovviamente sempre molta attenzione alla leggibilità e alla fruibilità del risultato finale.

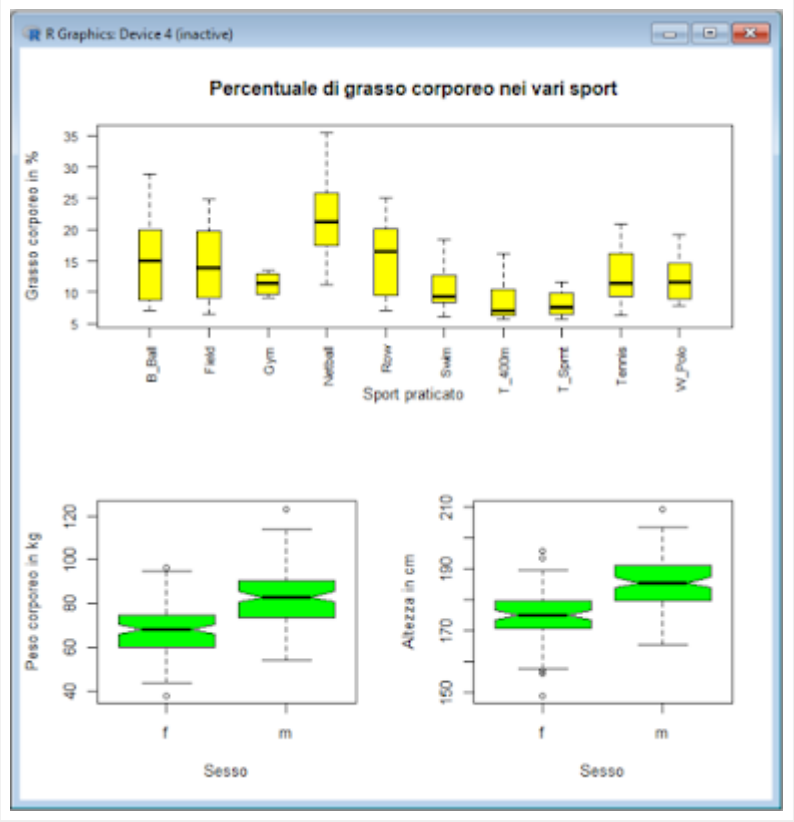
Copiate questo script, incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# INSERIRE PIU' GRAFICI NELLA STESSA IMMAGINE
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
layout(matrix(c(1,1,2,3), 2, 2, byrow=TRUE)) # tre grafici, il primo occupa riga 1/colonne 1 e 2, il secondo riga 2/colonna 1, il terzo riga 2/colonna 2
#
boxplot(ais$pcBfat~ais$sport, horizontal=FALSE, boxwex=0.4, cex.axis=0.8, las=2, main="Percentuale di grasso corporeo nei vari sport", xlab="Sport praticato", ylab="Grasso corporeo in %", notch=FALSE, col="yellow") # [1] valori della percentuale di grasso corporeo aggregati per sport
#
boxplot(ais$wt~ais$sex, xlab="Sesso", ylab="Peso corporeo in kg", notch=TRUE, col="green") # [2] valori del peso corporeo aggregati per sesso
#
boxplot(ais$ht~ais$sex, xlab="Sesso", ylab="Altezza in cm", notch=TRUE, col="green") # [3] valori di altezza aggregati per sesso
#
```

In questo caso vediamo come organizzare i grafici all'interno dell'immagine in maniera più articolata con **layout(matrix(c(1,1,2,3), 2, 2, byrow=TRUE))**. Viene infatti predisposta la suddivisione della finestra in **2** righe per **2** colonne (**... 2, 2, ...**) procedendo per riga (**byrow=TRUE**), ma specificando con il primo argomento **c(1,1,2,3)** che il primo grafico occuperà riga 1/colonne 1 e 2, il secondo grafico occuperà riga 2/colonna 1 e il terzo grafico occuperà riga 2/colonna 2, con questa logica

1	1

e questo risultato.



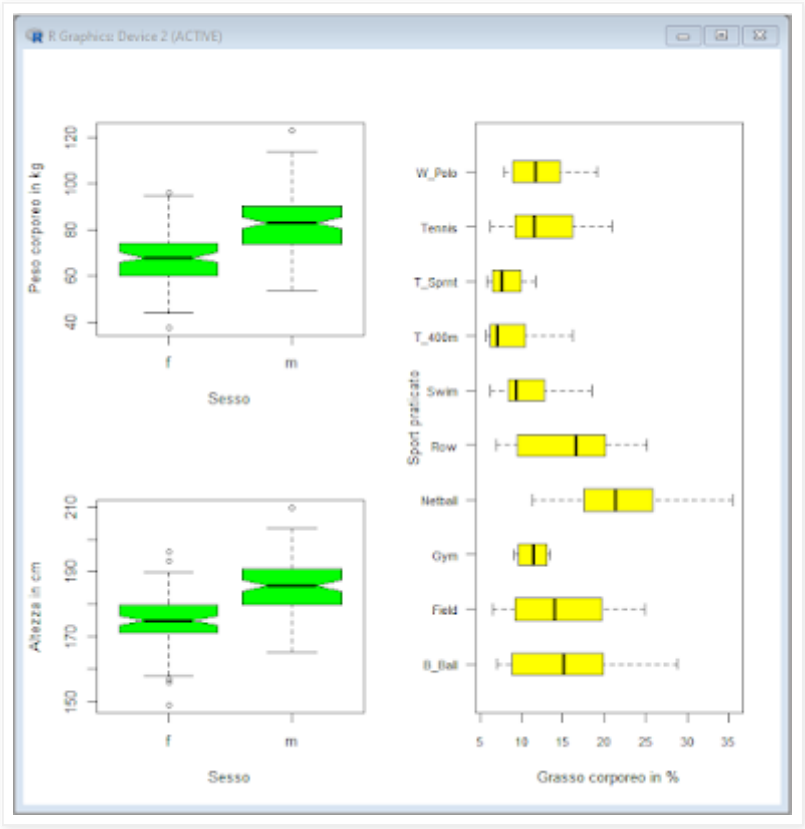
Copiate questo script, incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# INSERIRE PIU' GRAFICI NELLA STESSA IMMAGINE
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
layout(matrix(c(1,3,2,3), 2, 2, byrow=TRUE)) # (4) tre grafici, il primo occupa riga 1/colonna 1, il secondo riga 2/colonna 1, il terzo righe 1 e 2/colonna 2
#
boxplot(ais$wt~ais$sex, xlab="Sesso", ylab="Peso corporeo in kg", notch=TRUE, col="green") # [1] valori del peso corporeo aggregati per sesso
#
boxplot(ais$ht~ais$sex, xlab="Sesso", ylab="Altezza in cm", notch=TRUE, col="green") # [2] valori di altezza aggregati per sesso
#
boxplot(ais$pcBfat~ais$sport, horizontal=TRUE, boxwex=0.4, cex.axis=0.8, las=1, xlab="Grasso corporeo in %", ylab="Sport praticato", notch=FALSE, col="yellow") # [3] valori della percentuale di grasso corporeo aggregati per sport
#
```

In questo script con **layout(matrix(c(1,3,2,3), 2, 2, byrow=TRUE))** è stata predisposta di nuovo la suddivisione della finestra in **2** righe per **2** colonne (**... 2, 2, ...**) procedendo per riga (**byrow=TRUE**), ma questa volta specificando con il primo argomento **c(1,3,2,3)** che il primo grafico occuperà riga 1/colonna 1, il secondo grafico occuperà riga 2/colonna 1 e il terzo grafico occuperà le righe 1 e 2 della colonna 2 - con i boxplot in orizzontale (**horizontal=TRUE**) e le etichette delle ascisse in orizzontale (**las=1**) - con questa logica

1	3
2	3

e con questo risultato.



Finora abbiamo visto come organizzare più grafici all'interno della finestra grafica standard. Ma potremmo volere ampliare la finestra grafica, come facciamo ora riprendendo l'esempio precedente.

Copiate quest'ultimo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# INSERIRE PIU' GRAFICI NELLA STESSA IMMAGINE
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
#
dev.new(width=45, height=15, unit="cm") # nuova dimensione della finestra grafica
layout(matrix(c(1,2,3), 1, 3, byrow=TRUE)) # tre grafici in 1 riga e 3 colonne
#
boxplot(ais$wt~ais$sex, xlab="Sesso", ylab="Peso corporeo in kg", notch=TRUE,
col="green") # [1] valori del peso corporeo aggregati per sesso
#
boxplot(ais$ht~ais$sex, xlab="Sesso", ylab="Altezza in cm", notch=TRUE, col="green") #
[2] valori di altezza aggregati per sesso
#
boxplot(ais$pcBfat~ais$sport, horizontal=FALSE, boxwex=0.4, cex.axis=0.8, las=2,
main="Percentuale di grasso corporeo nei vari sport", xlab="Sport praticato",
ylab="Grasso corporeo in %", notch=FALSE, col="yellow") # [3] valori della percentuale di
grasso corporeo aggregati per sport
#
```

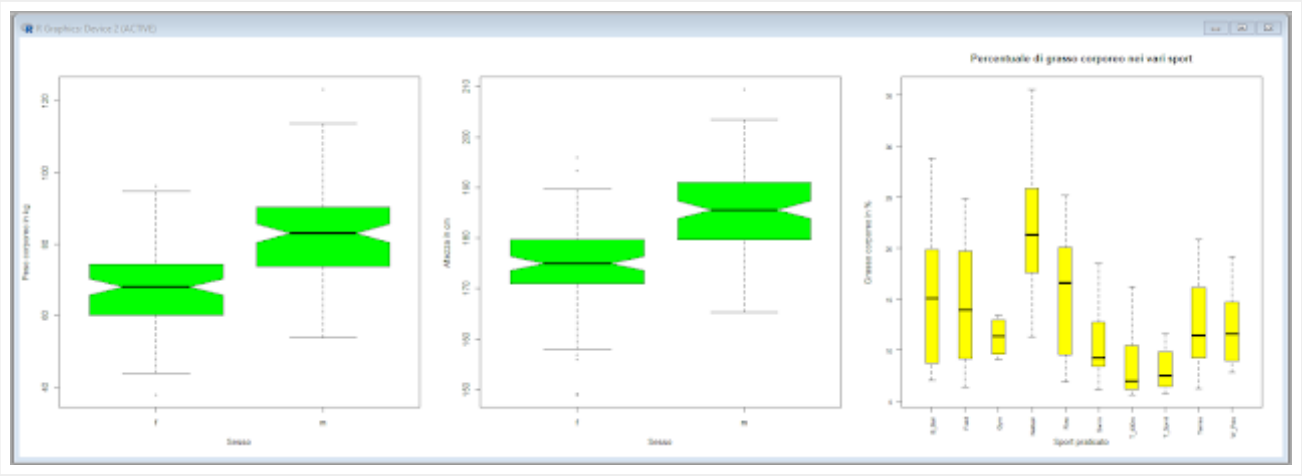
Per specificare le nuove dimensioni della finestra grafica impieghiamo la funzione **dev.new()** che richiede semplicemente larghezza della finestra, altezza della finestra e unità di misura. Queste

possono essere pollici (**unit="in"**), pixel (**unit="px"**) o centimetri (**unit="cm"**), come facciamo noi definendo quindi una finestra molto ampia, di **45** centimetri di larghezza (**width=**) per **15** centimetri di altezza (**height=**).

La funzione **layout(matrix(c(1,2,3), 1, 3, byrow=TRUE))** predispone la suddivisione della finestra in **1** riga per **3** colonne (**... 1, 3, ...**) procedendo per riga (**byrow=TRUE**) specificando con **c(1,2,3)** che vogliamo tre grafici affiancati, con questa logica

1	2	3

e con questo risultato:



Nel caso di grafici più complessi, come ad esempio quelli generati con il pacchetto **ggplot2**, le funzioni qui riportate potrebbero essere inadeguate, e per disporre più grafici all'interno della stessa immagine potrebbe essere necessario impiegare la funzione **grid.arrange()** inclusa nel pacchetto **gridExtra** [4]: trovate un esempio di applicazione di questa funzione nel post che illustra i grafici a punti [5].

Infine ricordo che impiegando l'utilità riportata nel post Salvare i grafici di R in un file potete salvare le immagini realizzate con **R** sotto forma di file `.bmp`, `.jpeg`, `.png`, `.pdf`, `.ps` per poterle stampare, archiviare, inserire in una pubblicazione, in un post o in un sito web.

---

[1] Digitate **help(*nomedellafunzione*)** nella `Console di R` per consultare la documentazione delle funzioni.

[2] La funzione **boxplot()** è illustrata nel post Grafici a scatola con i baffi (boxplot).

[3] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[4] Il manuale di riferimento *Package 'gridExtra'* lo trovate facendo click sul link alla voce *Reference manual* nella pagina di documentazione del pacchetto. URL consultato il 03/12/2022.

[5] Vedere il post Grafici a punti (dotplot).

mercoledì 25 dicembre 2019

Come stabilire il giusto numero di cifre significative

Per le **cifre significative** va ricordato che:
→ gli zero prima dei numeri diversi da zero non fanno parte delle cifre significative, quindi 0.0193 ha tre cifre significative;
→ gli zero dopo i numeri diversi da zero fanno parte delle cifre significative, quindi 0.30 ha due cifre significative;
→ il numero di cifre significative dipende dall'**errore del processo di misura**.

Si supponga che una serie di misure abbiano media = `9.37652`, deviazione standard = `0.10835` ed errore standard = `0.004203`. Dall'errore standard si ricava che le misure effettuate presentano una incertezza a livello della terza cifra decimale (`0.00`**`4`**`203`), dovuta all'errore del processo di misura. In questo caso non avrebbe senso riportare la media e la deviazione standard con più di tre cifre decimali: la media `9.37652` deve essere arrotondata e riportata come `9.376` e la deviazione standard `0.10835` deve essere arrotondata e riportata come `0.108` e il risultato sarà quindi espresso come `9.376 ± 0.108` (media ± deviazione standard) o come `9.376 ± 0.004` (media ± errore standard).

Se non si conosce l'errore standard in genere si esprime la deviazione standard con due cifre significative. Così avendo ottenuto ad esempio una media = `1.34648` e una deviazione standard = `0.10295` quest'ultima va riportata come `0.10` con due cifre significative e il risultato sarà quindi espresso, arrotondando anche la media a due decimali, come `1.35 ± 0.11` (media ± deviazione standard).

Per effettuare l'arrotondamento si può procedere seguendo quanto riportato nel post Come arrotondare i numeri.

lunedì 6 gennaio 2020

Test parametrici e non parametrici per più campioni indipendenti

Per confrontare **due** campioni indipendenti [1] è necessario impiegare:
→ un metodo parametrico, il test t di Student, quando i dati sono distribuiti in modo gaussiano;
→ metodi non parametrici, come il test di Wilcoxon per campioni indipendenti (in genere meglio noto come test U di Mann-Whitney) e il test di Kruskall-Wallis, quando i dati non sono distribuiti in modo gaussiano.

Tuttavia per confrontare **più di due** campioni indipendenti non è corretto applicare un test per due campioni indipendenti ripetitivamente a tutte le diverse possibili coppie di campioni: così, ad esempio, nel caso più semplice, quello di tre campioni, non è corretto confrontare mediante l'ordinario test t di Student (assumendo che si tratti di distribuzioni gaussiane) il primo campione con il secondo campione, il primo campione con il terzo e il secondo campione con il terzo. Questo perché la probabilità di commettere un errore di primo tipo [2], cioè la probabilità di considerare la differenza tra le medie di due campioni significativa quando invece non è significativa (probabilità di un falso positivo), nel caso di confronti multipli viene moltiplicata per il numero di confronti effettuati.

In altre parole un valore del test t di Student che nel caso di due campioni (per i quali un solo confronto è possibile) corrisponde a una probabilità di commettere un errore di primo tipo del `5%`:
→ nel caso di tre campioni (essendo 3 il numero di confronti possibili) corrisponde a una probabilità di commettere un errore di primo tipo del 3 x 5% = `15%`;
→ nel caso di quattro campioni (essendo 6 il numero di confronti possibili) corrisponde a una probabilità di commettere un errore di primo tipo del 6 x 5% = `30%`;
→ nel caso di cinque campioni (essendo 10 il numero di confronti possibili) corrisponde a una probabilità di commettere un errore di primo tipo del 10 x 5% = `50%`;
e così via.

Reciprocamente se nel confronto tra due campioni si assume come valore soglia per la significatività il classico `p = 0.05` è indispensabile ricordarsi che questo valore:
→ quando si confrontano tre campioni deve essere sostituito con `p = 0.017` (cioè 0.05/3);
→ quando si confrontano quattro campioni deve essere sostituito con `p = 0.008` (cioè 0.05/6);
→ quando si confrontano cinque campioni deve essere sostituito con `p = 0.005` (cioè 0.05/10);
e così via.

Anche se si potrebbe procedere seguendo queste semplici regole, di fatto sono disponibili test che, nel caso di **confronti multipli**, applicano automaticamente le correzioni necessarie per controbilanciare l'aumento dei falsi positivi, cioè l'aumento della probabilità di considerare la differenza tra le medie di due campioni significativa quando invece non è significativa.

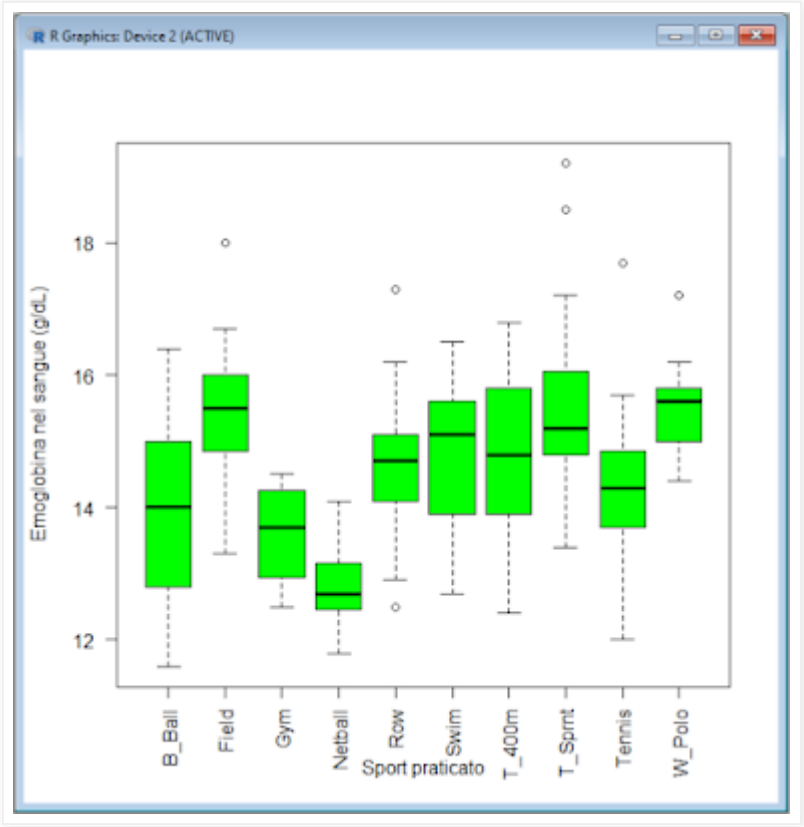
I principali test per il **confronto tra più di due campioni indipendenti** li vediamo ora, applicati ai valori di concentrazione dell'emoglobina nel sangue rilevata in 202 atleti australiani e riportata nella variabile **hg** della tabella **ais** inclusa nel pacchetto **DAAG**. Accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [3].

Copiate lo script che segue, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# CONFRONTARE TRA DI LORO PIU' CAMPIONI con un metodo parametrico
# test t di Student con la correzione di Bonferroni per confronti multipli
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
#
```

boxplot(hg~sport, data=ais, horizontal=FALSE, notch=FALSE, col="green", las=2, xlab="Sport praticato", ylab="Emoglobina nel sangue (g/dL)") # boxplot della concentrazione di emoglobina per sport praticato
#
pairwise.t.test(aishg, aissport, p.adjust.method="bonferroni") # test t di Student per campioni indipendenti con la correzione di Bonferroni per confronti multipli
#

Realizziamo un grafico a scatola con i baffi per la concentrazione **hg** dell'emoglobina nel sangue (espressa in g/dL) aggregando i valori in base allo sport praticato (**hg~sport**).



Ora, per avere un riscontro quantitativo e oggettivo delle differenze rilevate nel grafico, impiegando la funzione **pairwise.t.test()** impieghiamo il più classico test parametrico, con la più tradizionale correzione per confronti multipli, il **test t di Student con la correzione di Bonferroni** (**method="bonferroni"**):

```
        Pairwise comparisons using t tests with pooled SD

data:  ais$hg and ais$sport

         B_Ball Field   Gym    Netball Row    Swim   T_400m T_Sprnt Tennis
Field    0.0023 -       -      -       -      -      -      -       -
Gym      1.0000 0.1012  -      -       -      -      -      -       -
Netball  0.0050 2.4e-11 1.0000 -       -      -      -      -       -
Row      1.0000 0.1676  1.0000 6.5e-07 -      -      -      -       -
Swim     1.0000 1.0000  1.0000 2.3e-06 1.0000 -      -      -       -
T_400m   1.0000 0.8321  1.0000 2.7e-07 1.0000 1.0000 -      -       -
T_Sprnt  0.0015 1.0000  0.0606 5.1e-11 0.0928 0.5925 0.4455 -       -
Tennis   1.0000 0.2188  1.0000 0.0178  1.0000 1.0000 1.0000 0.1213  -
W_Polo   0.0036 1.0000  0.1078 8.8e-11 0.2180 1.0000 0.9715 1.0000  0.2490

P value adjustment method: bonferroni
```

Il test t di Student con la correzione di Bonferroni conferma che le atlete (sono solo donne) che praticano `Netball` hanno in comune una bassa concentrazione dell'emoglobina, e che la concentrazione media di questo gruppo si discosta significativamente dalle concentrazioni medie dei rimanenti gruppi di atleti, con la sola eccezione (`p = 1.0000`) del gruppo delle atlete che pratica la ginnastica (`Gym`). Questo è il dato certamente più rilevante, con poche altre differenze significative che sono evidenziate nella tabella.

Da notare che la funzione **pairwise.t.test()** prevede le seguenti correzioni alternative meno conservative [4] di quella di Bonferroni: secondo Holm (1979) (**method="holm"**), Hochberg (1988) (**method="hochberg"**), Hommel (1988) (**method="hommel"**), Benjamini & Hochberg (1995) (**method="BH"** o **method="fdr"**), Benjamini & Yekutieli (2001) (**method="BY"**), e **method="none"** per nessuna correzione (ovviamente sconsigliato).

Vediamo ora alcuni metodi non parametrici, contenuti nel pacchetto **PMCMRplus** [5], che è necessario scaricare e installare dal **CRAN**. Copiate lo script che segue, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# CONFRONTARE TRA DI LORO PIU' CAMPIONI con metodi non parametrici
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
#
boxplot(hg~sport, data=ais, horizontal=FALSE, notch=FALSE, col="green", las=2,
xlab="Sport praticato", ylab="Emoglobina nel sangue (g/dL)") # boxplot della
concentrazione di emoglobina per sport praticato
#
library(PMCMRplus) # carica il pacchetto
spearmanTest(hg~sport, data=ais) # rho di Spearman (test omnibus, non parametrico)
kwAllPairsConoverTest(hg~sport, data=ais) # test di Conover (non parametrico)
kwAllPairsDunnTest(hg~sport, data=ais) # test di Dunn (non parametrico)
kwAllPairsNemenyiTest(hg~sport, data=ais) # test di Nemenyi (non parametrico)
#
```

Quelli che qui impieghiamo sono tutti test non parametrici che quindi non prevedono che i dati siano distribuiti in modo gaussiano e che sono eseguiti mediante:
→ la funzione **spearmanTest()**, per il **test ρ (rho) di Spearman** [6], un test non parametrico che riporta un risultato unico per tutte le coppie di campioni (*test omnibus*);
→ la funzione **kwAllPairsConoverTest()** per il **test di Conover**, un test non parametrico basato sui ranghi di tipo Kruskal, che riporta un risultato per ogni coppia di campioni;
→ la funzione **kwAllPairsDunnTest()** per il **test di Dunn**, un test non parametrico basato sui ranghi di tipo Kruskal, che riporta un risultato per ogni coppia di campioni;
→ la funzione **kwAllPairsNemenyiTest()** per il **test di Nemenyi**, un test non parametrico basato sul confronto tra i ranghi, che riporta un risultato per ogni coppia di campioni.

I test di Conover, di Dunn e di Nemenyi, che riportano un risultato per ogni coppia di campioni, forniscono risultati sostanzialmente sovrapponibili, qui riporto per semplicità solamente i risultati dell'ultimo di questi.

```
> kwAllPairsNemenyiTest(hg~sport, data=ais) # test di Nemenyi (non parametrico)

          Pairwise comparisons using Tukey-Kramer-Nemenyi all-pairs test with Tukey-
Dist approximation

data: hg by sport

        B_Ball Field   Gym    Netball Row    Swim   T_400m T_Sprnt Tennis
Field   0.023  -       -      -       -      -      -      -       -
```

```
Gym     0.997  0.152   -     -       -     -     -      -       -
Netball 0.028  4.3e-09 0.996 -       -     -     -      -       -
Row     0.981  0.201   0.905 8.6e-05 -     -     -      -       -
Swim    0.867  0.697   0.785 7.8e-05 1.000 -     -      -       -
T_400m  0.844  0.578   0.785 2.0e-05 1.000 1.000 -      -       -
T_Sprnt 0.133  1.000   0.279 5.9e-07 0.573 0.934 0.891  -       -
Tennis  1.000  0.196   0.997 0.158   0.999 0.981 0.981  0.441   -
W_Polo  0.018  1.000   0.126 5.0e-09 0.160 0.614 0.494  1.000   0.158

P value adjustment method: single-step
alternative hypothesis: two.sided
Messaggio di avvertimento:
In kwAllPairsNemenyiTest.default(c(12.3, 12.7, 11.6, 12.6, 14, 12.5,  :
  Ties are present, p-values are not corrected.
```

Il pacchetto **PMCMRplus** include tra le molte altre anche la funzione **tukeyTest()** per il **test di Tukey**, un test parametrico,per il confronto di campioni con distribuzione gaussiana e con uguale varianza, che riporta un risultato per ogni coppia di campioni, un test sovrapponibile per applicazione e conclusioni al test t di Student con la correzione di Bonferroni, anche se meno conservativo. La sintassi è sempre la stessa, per visualizzarne i risultati copiate e incollate nella `Console di R` questa riga di codice e premete ↵ `Invio`:

tukeyTest(hg~sport, data=ais) # test di Tukey (parametrico)

Questo test fornisce risultati quasi identici a quelli del test t di Student con la correzione di Bonferroni:

```
        Pairwise comparisons using Tukey's test

data: hg by sport

        B_Ball Field   Gym    Netball Row    Swim   T_400m T_Sprnt Tennis
Field   0.0020 -       -      -       -      -      -      -       -
Gym     0.9982 0.0671  -      -       -      -      -      -       -
Netball 0.0043 2.4e-11 0.9553 -       -      -      -      -       -
Row     0.8104 0.1028  0.8151 6.4e-07 -      -      -      -       -
Swim    0.6798 0.4141  0.7173 2.3e-06 1.0000 -      -      -       -
T_400m  0.5633 0.3463  0.6829 2.7e-07 1.0000 1.0000 -      -       -
T_Sprnt 0.0013 1.0000  0.0429 5.1e-11 0.0623 0.2739 0.2224 -       -
Tennis  1.0000 0.1278  0.9871 0.0142  0.9993 0.9920 0.9872 0.0784  -
W_Polo  0.0031 1.0000  0.0709 8.8e-11 0.1275 0.4467 0.3833 1.0000  0.1418

P value adjustment method: single-step
alternative hypothesis: two.sided
```

In conclusione: nel nostro caso i risultati dei due classici test parametrici per il confronto tra medie, il test t di Student con la correzione di Bonferroni e il test di Tukey, sono molto simili a quelli dei principali test non parametrici per il confronto tra mediane (test ρ di Spearman, test di Conover, di Dunn, di Nemenyi). Questo supporta il fatto che i dati originali sono distribuiti in modo gaussiano: per definizione in distribuzioni perfettamente gaussiane i risultati dei test parametrici e dei test non parametrici coincidono, e coincidono medie e mediane. Tuttavia va sottolineato che un approccio rigoroso e puntuale deve prevedere una verifica preliminare della gaussianità dei dati che in questo caso viene lasciata come esercizio rimandando ai metodi che trovate descritti in altri post [7].

---

[1] Vedere il post Test parametrici e non parametrici per due campioni indipendenti.

[2] Wikipedia. *Type I and type II errors*. URL consultato il 10/01/2020: http://bit.ly/2R1VdyF

[3] Vedere il post Il set di dati ais. La concentrazione dell'emoglobina viene espressa in grammi per decilitro di sangue (g/dL). Nel post trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[4] Un metodo statistico "meno conservativo" a parità di condizioni riporta più frequentemente differenze significative. Personalmente preferisco in ogni caso impiegare i metodi più conservativi, in quanto riducono la probabilità di considerare significativa una differenza che invece non è significativa (ovvero riducono la probabilità di un falso positivo).

[5] **PMCMR** è l'acronimo di Calculate **P**airwise **M**ultiple **C**omparisons of **M**ean **R**ank Sums Extended. Si tratta di un pacchetto che include una serie molto ampia sia di test omnibus (che forniscono un solo risultato per tutti i confronti effettuati), sia per il confronto tra tutte le coppie di campioni, sia parametrici, sia non parametrici. Vedere sul **CRAN** il reference manual *PMCMRplus: Calculate Pairwise Multiple Comparisons of Mean Rank Sums Extended*. URL consultato il 18/12/2019: http://bit.ly/2EwRymI

[6] Per il test ρ (rho) di Spearman vedere anche il post Coefficienti di correlazione parametrici e non parametrici.

[7] Vedere:
→ il post Test di normalità (gaussianità)
→ il post Tabulare una serie di test di normalità (gaussianità)

giovedì 12 marzo 2020

Importazione dei dati da un file di solo testo

Abbiamo visto in precedenza come sono organizzati i dati in un file `.csv`, il formato raccomandato per **R**, e abbiamo visto che un file `.csv` è un semplice file di testo [1].

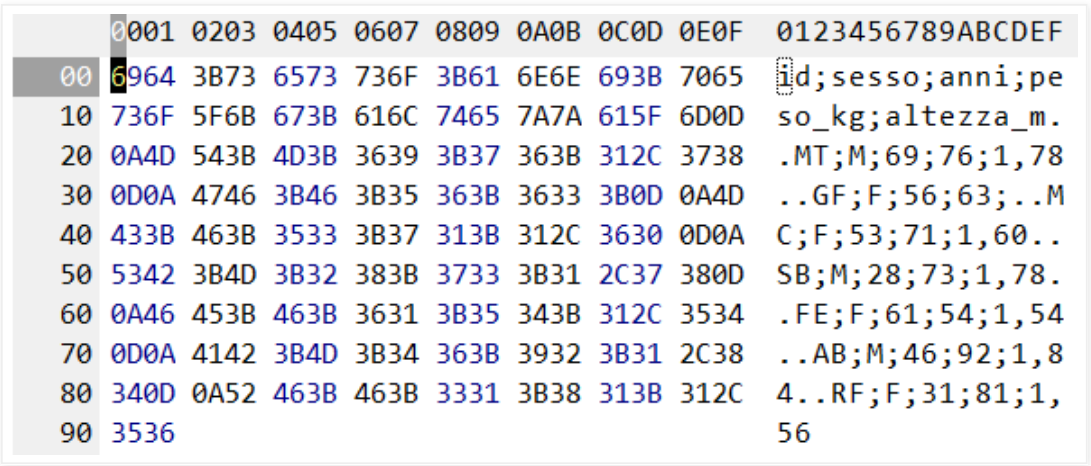
Se aprite con un editor di testo il file `importa_csv.csv` (se non l'avete già scaricato, trovate come farlo nella pagina Dati) questo è quello che vi appare:

id;sesso;anni;peso_kg;altezza_m
MT;M;69;76;1,78
GF;F;56;63;
MC;F;53;71;1,60
SB;M;28;73;1,78
FE;F;61;54;1,54
AB;M;46;92;1,84
RF;F;31;81;1,56

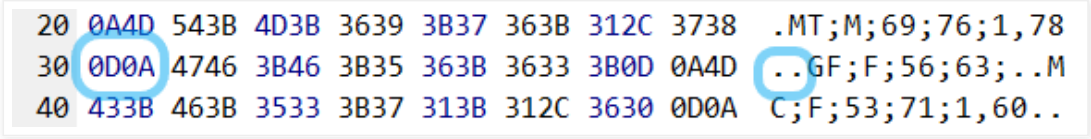
Se ci pensate bene, però, oltre a lettere, numeri, caratteri di interpunzione e qualche altro carattere accessorio, o, più tecnicamente, oltre a una serie di **caratteri ASCII stampabili** [2] nel testo è incluso un "a capo" (o se preferite un `↵ Invio`) che segna il passaggio da una riga alla successiva, o, in altre parole, che separa un record dal successivo: ed è chiaro che in qualche modo questo carattere, uno dei **caratteri non stampabili** (detti anche **caratteri di controllo**) del codice ASCII, deve essere stato codificato nel file.

Che proprio questo sia il caso è dimostrabile aprendo il file con un editor di testo che sia in grado di mostrare, oltre ai caratteri ASCII, anche i caratteri di controllo contenuti nel file, uno dei quali è appunto il carattere di "`a capo`".

Per esempio se aprite il file `importa_csv.csv` con **PSPad** [3] ed attivate nel menù `Visualizza` la funzione `Modalità esadecimale` vedete questo:



Sulla destra compaiono i caratteri ASCII contenuti nel file, sulla sinistra compare il valore esadecimale corrispondente. Sulla destra vedete ricorrere al termine di ogni record una sequenza di due punti consecutivi (`..`), sulla sinistra vedete che ad essi corrisponde la sequenza `0D0A` che è la rappresentazione esadecimale di `CR` (`Carriage Return`) e di `LF` (`Line Feed`) cioè rispettivamente del ritorno di carrello a inizio riga (`0D` esadecimale) e del passaggio ad una nuova riga (`0A` esadecimale) che l'informatica ha ereditato dalla macchina da scrivere



Questi due caratteri ASCII l'editor di testo li interpreta come caratteri di controllo, cioè come caratteri che non devono essere rappresentati, ma che indicano una azione da compiere: passare a una nuova riga di testo inserendo un "`a capo`" (un `↵ Invio`) [4].

R è ovviamente in grado di interpretare correttamente i caratteri di controllo, infatti se nella `Console di R` incollate questa riga di codice

mydatacsv <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",")

e premete ↵ `Invio` per importare i dati dal file `.csv`, quindi digitate

mydatacsv

e premete ↵ `Invio` per mostrare i dati importati, ottenete questo risultato:

```
> mydatacsv <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",")
> mydatacsv
  id sesso anni peso_kg altezza_m

1 MT     M   69      76      1.78
2 GF     F   56      63        NA
3 MC     F   53      71      1.60
4 SB     M   28      73      1.78
5 FE     F   61      54      1.54
6 AB     M   46      92      1.84
7 RF     F   31      81      1.56
```

(`NA` indica un dato mancante).

Ma con **R** è possibile un'altra cosa: si possono codificare i caratteri di controllo in chiaro, come viene fatto nel file `importa_txt.txt` che contiene gli stessi dati e che, aperto con un editor di testo, così vi appare:

MT;M;69;76;1,78\nGF;F;56;63;\nMC;F;53;71;1,60\nSB;M;28;73;1,78\nFE;F;61;54;1,54\nAB;M;46;92;1,84\nRF;F;31;81;1,56

Da notare che nel file `importa_txt.txt` rispetto al corrispondente file `.csv`:
→ i caratteri di controllo `CR` e `LF` (nascosti nel file `.csv`) sono stati sostituiti con i caratteri (in chiaro) `\n` ove la barra rovesciata `\` (o `backslash`) indica a **R** la presenza di un carattere di controllo e `n` sta per `New Line` ("passa a una nuova riga");
→ il file non contiene i nomi delle variabili, che devono essere specificati nella funzione **read.table()** mediante l'argomento **col.names**;
→ l'ultimo carattere nel file **deve** essere un ↵ `Invio` (cioè un "a capo") che in questo caso deve immediatamente seguire il `6` del dato finale `1,56`.

Il file `importa_txt.txt` potete scaricarlo e installarlo nella cartella `C:\Rdati` seguendo le istruzioni fornite nella pagina Dati. Ora copiate nella `Console di R` questa riga di codice

mydatatxt <- read.table("c:/Rdati/importa_txt.txt", header=TRUE, col.names=c("id", "sesso", "anni", "peso_kg", "altezza_m"), sep=";", dec=",", allowEscapes=TRUE)

e premete ↵ `Invio` per importare i dati, quindi digitate

mydatatxt

e premete ↵ `Invio` per visualizzare i dati importati.

Da notare che l'argomento **allowEscapes=TRUE** fa si che **R** interpreti la sequenza `\n` come un carattere di controllo (un ↵ `Invio`) e non come due semplici caratteri di testo. Come vedete il risultato ottenuto

```
> mydatatxt <- read.table("c:/Rdati/importa_txt.txt", header=TRUE,
col.names=c("id","sesso","anni","peso_kg","altezza_m"), sep=";", dec=",",
allowEscapes=TRUE)
> mydatatxt
  id sesso anni peso_kg altezza_m

1 MT     M   69      76      1.78
2 GF     F   56      63        NA
3 MC     F   53      71      1.60
```

```
4 SB      M   28       73       1.78
5 FE      F   61       54       1.54
6 AB      M   46       92       1.84
7 RF      F   31       81       1.56
```

è identico a quello ottenuto importando i dati del file `.csv`.

Il senso di tutto ciò? Semplice. Guardate il set di dati Galton, si tratta di 928 righe di dati, ciascuna corrispondente a un record, più una riga per i nomi delle variabili. Se invece di un file `.csv` impiegate un file `.txt` separando i record con `\n` potete compattare i dati in un numero enormemente inferiore di righe, cosa che ad esempio rende possibile inserire un numero così elevato di record in un testo.

Addendum

Se siete arrivati fino a qui, avete certamente notato che nel post manca una cosa fondamentale: un metodo semplice per generare un file di testo nel quale i caratteri di controllo nascosti `CR` e `LF` sono sostituiti con un `\n` riportato in chiaro.

In effetti ci sono programmi che consentono di effettuare la sostituzione online: alcuni forniscono come risultato un file nel quale la sequenza di caratteri di controllo `CRLF` è automaticamente sostituita con `\n` [5], altri forniscono come risultato un file nel quale detta sequenza è sostituita con uno spazio vuoto [6], che successivamente può essere sostituito manualmente con `\n`.

Qui vediamo come effettuare la sostituzione con la seconda modalità, che è la più semplice, impiegando **R** e un comune editor di testo.

Scaricate dal **CRAN** e installate il pacchetto **stringr** [7]. Quindi copiate e incollate nella `Console di R` questo script e premete `↵ Invio`

```
# SALVA I DATI IN UN FILE DI TESTO ELIMINANDO I CARATTERI DI CONTROLLO CR e LF
#
library(stringr) # carica il pacchetto
mydata <- readLines("C:/Rdati/importa_csv.csv") # legge il file
newdata <- mydata[-c(1)] # elimina la prima riga che contiene i nomi delle variabili
mystring <- str_c(newdata, collapse=" ") # concatena i record in un'unica stringa separandoli con uno spazio
write(mystring, file="C:/Rdati/temp.txt") # salva la stringa in un file di testo
```

Va subito precisato che il messaggio che vi comparirà

```
Warning message:
In readLines("C:/Rdati/importa_csv.csv") :
incomplete final line found on 'C:/Rdati/importa_csv.csv'
```

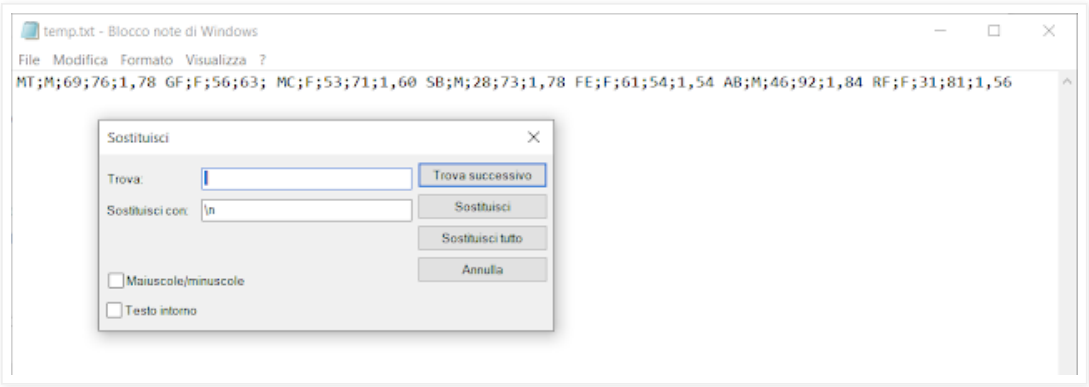
non indica un errore, ma è un semplice avvertimento che segnala la mancanza della sequenza `CRLF` (cioè di un `↵ Invio`) alla fine del file: i dati sono importati correttamente, come potete constatare digitando **mydata**.

Nelle cinque righe di codice, nell'ordine:
→ con la funzione **library()** viene caricato il pacchetto **stringr**;
→ con **readLine(s)** un normale file .csv (vedere il post Importazione dei dati da un file .csv) viene importato riga per riga;
→ con **[-c(1)]** viene eliminata la prima riga, quella che contiene i nomi delle variabili;
→ con la funzione **str_c()** del pacchetto **stringr** le righe (record) del file importate sono concatenate in un'unica stringa separando l'una dall'altra con uno spazio (**collapse=" "**) ;
→ la stringa che risulta dal concatenamento viene salvata nel file `C:/Rdati/temp.txt`.

Digitate **mydata**, poi **newdata** e infine **mystring** per visualizzare la progressiva trasformazione dei dati.

Infine, impiegando un editor di testo come il `Blocco note di Windows`:

→ aprite il file `C:/Rdati/temp.txt`
→ utilizzate la funzione `Modifica > Sostituisci` per sostituire lo spazio vuoto con i caratteri `\n`



A questo punto ricordatevi che l'ultimo carattere del file deve essere un ↲ `Invio` (cioè un "a capo") che dovete inserire manualmente immediatamente dopo il `6` del dato finale `1,56`. Poi salvate il risultato nel file definitivo, denominato a piacere ma possibilmente per coerenza con estensione `.txt`, e avrete finalmente un file di dati di **R** in formato testo come questo

MT;M;69;76;1,78\nGF;F;56;63;\nMC;F;53;71;1,60\nSB;M;28;73;1,78\nFE;F;61;54;1,54\nAB;M;46;92;1,84\nRF;F;31;81;1,56

che impiega come separatore di record la sequenza di caratteri `\n` codificata in chiaro, e che può essere importato in **R** mediante la funzione **read.table()** impiegando l'argomento **allowEscapes=TRUE**.

Supponendo che abbiate denominato questo file `C:\Rdati\ilmiofile_txt.txt` se copiate nella `Console di R` questa riga di codice

mydatatxt <- read.table("c:/Rdati/ilmiofile_txt.txt", header=TRUE, col.names=c("id", "sesso", "anni", "peso_kg", "altezza_m"), sep=";", dec=",", allowEscapes=TRUE)

e premete ↲ `Invio` per importare i dati, quindi digitate

mydatatxt

e premete ↲ `Invio` per visualizzare i dati importati, potete verificare che il tutto ha funzionato correttamente e che pertanto siete in grado, al bisogno, da un lato di generare un file di testo con i caratteri di controllo in chiaro, e dall'altro lato di importarne i dati in **R**.

---

[1] Vedere il post Importazione dei dati da un file .csv

[2] Vedere il post Codifica dei caratteri ASCII, ANSI, Unicode (UTF)

[3] *PSPad freeware editor*. URL consultato il 16/03/2020: http://bit.ly/3aUXAws

[4] In realtà esistono tre modi per codificare un "a capo" (un ↲ `Invio`), e dipendono dal sistema operativo impiegato:
→ `CRLF` (`\r\n`): usato da MS-DOS e Microsoft Windows, al quale si fa qui riferimento;
→ `LF` (`\n`): usato da sistemi Unix, Linux e Apple (GNU/Linux, Mac OS X e macOS);
→ `CR` (`\r`): usato da Commodore e Apple (Mac OS fino alla versione 9 inclusa).

[5] *Remove/Replace all line breaks of a text*. URL consultato il 16/03/2020: http://bit.ly/39Wt2JF

[6] *Convert Newlines to Spaces*. URL consultato il 16/03/2020: http://bit.ly/2x481ht

[7] Vedere il manuale di riferimento del pacchetto *Package 'stringr'*. URL consultato il 16/03/2020: http://bit.ly/33lINaC

mercoledì 8 aprile 2020

Analizzare graficamente la distribuzione di una variabile

Questa volta l'obiettivo è di fornire uno strumento pratico, immediatamente riadattabile a nuovi dati, che condensa in una sola immagine quattro grafici che illustrano la distribuzione di una variabile: una sintesi grafica che rappresenta un utile complemento ai test di normalità [1].

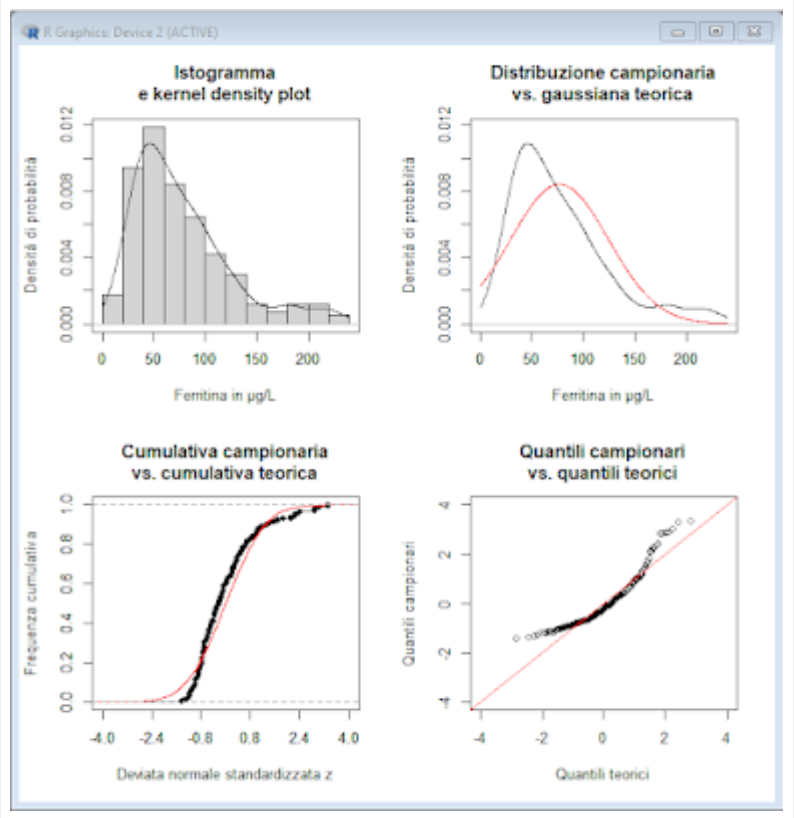
Come dati impieghiamo la concentrazione delle ferritina nel sangue rilevata in 202 atleti australiani riportata nella variabile **ferr** della tabella **ais** inclusa nel pacchetto **DAAG**. Potete scaricare il pacchetto dal **CRAN** oppure acquisire la tabella seguendo le indicazioni alternative riportate in [2].

Incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# ANALIZZARE GRAFICAMENTE LA DISTRIBUZIONE DI UNA VARIABILE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
mydata <- ais$ferr # salva in mydata i valori della variabile ferritina
par(mfrow=c(2,2)) # suddivide la finestra in quattro quadranti, uno per grafico
#
# istogramma e kernel density plot
#
hist(mydata, xlim=c(0, 250), ylim=c(0, 0.012), freq=FALSE, breaks="FD",
main="Istogramma\ne kernel density plot", xlab="Ferritina in µg/L", ylab="Densità di
probabilità") # traccia l'istogramma dei dati
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
plot(density(mydata, n=1024, from=0, to=250), xlim=c(0, 250), ylim=c(0, 0.012),
yaxt="n", col="black") # sovrappone il kernel density plot della distribuzione campionaria
#
# distribuzione campionaria vs. gaussiana teorica
#
plot(density(mydata, n=1024, from=0, to=250), xlim=c(0, 250), ylim=c(0, 0.012)) #
traccia il kernel density plot della distribuzione campionaria
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
x <- seq(0, 250, length.out=1024) # calcola i valori in ascisse della gaussiana teorica
y <- dnorm(x, mean=mean(mydata), sd=sd(mydata)) # calcola i valori in ordinate della
gaussiana teorica
plot(x, y, xlim=c(0, 250), ylim=c(0, 0.012), yaxt="n", col="red", type="l") # sovrappone la
distribuzione gaussiana teorica in colore rosso
title(main="Distribuzione campionaria\nvs. gaussiana teorica", xlab="Ferritina in µg/L",
ylab="Densità di probabilità") # aggiunge titolo e legende degli assi
#
# distribuzione cumulativa campionaria vs. distribuzione cumulativa teorica
#
par(new=FALSE, ann=TRUE) # per mostrare titolo e legende degli assi
plot(ecdf(scale(mydata)), main="Cumulativa campionaria\nvs. cumulativa teorica",
xlab="Deviata normale standardizzata z", ylab="Frequenza cumulativa", xlog = FALSE,
ylog = FALSE, xlim=c(-4, 4), ylim=c(0, 1), xaxp=c(-4, 4, 5), yaxp=c(0, 1, 5)) # traccia il
grafico della distribuzione cumulativa campionaria
par(new=TRUE, ann=FALSE) # consente di sovrapporre il grafico successivo
plot(ecdf(rnorm(10000, mean=0, sd=1)), col="red", xlog=FALSE, ylog=FALSE, xlim=c(-4,
4), ylim=c(0, 1), xaxp=c(-4, 4, 5), yaxp=c(0, 1, 5)) # sovrappone la distribuzione cumulativa
teorica in colore rosso
#
# quantili campionari vs. quantili teorici
#
```

par(new=FALSE, ann=TRUE) # per mostrare titolo e legende degli assi
qqnorm(scale(mydata), main="Quantili campionari\nvs. quantili teorici", xlab="Quantili teorici", ylab="Quantili campionari", xlim=c(-4, 4), ylim=c(-4, 4)) # traccia il grafico della distribuzione dei quantili campionari
abline (0, 1, col="red") # sovrappone la distribuzione dei quantili teorica in colore rosso
#

La ferritina è una proteina che, pur con alcune limitazioni, fornisce una misura dei depositi di ferro presenti nell'organismo, necessari per garantire una adeguata produzione di emoglobina. L'analisi grafica dei dati consente di confermare graficamente, per la concentrazione della ferritina nel sangue, una distribuzione non gaussiana.



Il codice dello script è illustrato dai commenti via via inseriti, ma vediamo brevemente il significato di questa rappresentazione.

Il primo grafico presenta la distribuzione dei dati sotto forma del classico istogramma [3] impiegando la funzione **hist()**, calcolando il numero di classi (**breaks="FD"**) mediante la regola di Freedman-Diaconis [4], e ponendo in ordinate la densità di probabilità (**freq=FALSE**).

Questo consente di sovrapporre all'istogramma mediante la funzione **plot()**, con la stessa scala delle ordinate, il corrispondente kernel density plot, la cui densità di probabilità è calcolata mediante la funzione **density()** in corrispondenza di 1024 valori (**n=1024**) compresi tra 0 (**from=0**) e 250 (**to=250**).

Con il secondo grafico entriamo nel vivo del confronto tra la distribuzione dei dati osservata, o distribuzione empirica o distribuzione campionaria che dir si voglia, e quella che i dati dovrebbero avere se fossero distribuiti secondo una gaussiana.

Questo viene fatto riportando prima il kernel density plot calcolato come al punto precedente, e riportando per confronto (in colore rosso) la distribuzione gaussiana teorica corrispondente, ottenuta calcolando con la funzione **dnorm()** la densità di probabilità (**y**) - quella che avremmo se dati con la media **mean=mean(mydata)** e la deviazione standard **sd=sd(mydata)** dei dati osservati fossero distribuiti in modo gaussiano - in corrispondenza di 1024 valori (**length.out=1024**) della **x** ottenuti

con la funzione **seq()**.

Il terzo grafico riporta la distribuzione cumulativa campionaria calcolata con la funzione **ecdf()** sui dati (**mydata**) del campione, e riporta per confronto (in colore rosso) la corrispondente distribuzione cumulativa teorica, calcolata sempre con la funzione **ecdf()** su un campione di 10 000 dati distribuiti in modo gaussiano con media **0** e deviazione standard **1**, generati impiegando la funzione **rnorm()** con gli argomenti **10 000, mean=0, sd=1**.

Il quarto grafico impiega la funzione **qqplot()** per calcolare i quantili della distribuzione campionaria in funzione dei quantili teorici, e ne confronta l'andamento con quello previsto (la linea retta in colore rosso) per dati che seguono una distribuzione gaussiana. Concettualmente, si tratta della trasformazione lineare del grafico della distribuzione cumulativa di cui al punto precedente.

In definitiva vediamo come i grafici, realizzati in modo semplice, forniscono una indicazione concisa di come e di quanto una distribuzione campionaria si discosta da una distribuzione gaussiana. E, come già detto all'inizio, offrono una sintesi che rappresenta un indispensabile complemento ai test statistici di normalità (gaussianità) [1].

Aggiungo che, nonostante siano ovviamente tecnicamente ineccepibili, distribuzione cumulativa e quantili tendono a comprimere, dal punto di vista grafico, la differenza tra la distribuzione campionaria e la distribuzione attesa nel caso di dati distribuiti in modo gaussiano. Tale differenza per certi versi appare più evidente nel semplice confronto tra la distribuzione osservata, rappresentata sotto forma di kernel density plot e la corrispondente distribuzione gaussiana (grafico in alto a destra).

I dettagli delle funzioni e degli argomenti impiegati sono riportati nei post che trattano le rispettive rappresentazioni grafiche (vedere alla pagina Indice) e sono come sempre accessibili digitando **help(***nomedellafunzione***)** nella `Console di R`.

Si fa infine notare la comparsa nel codice **R** di un **\n** che impone un "a capo" in alcune delle righe di testo.

Per riutilizzare questo script per analizzare altri dati è sufficiente:
→ nella prima riga di codice sostituire **ais** con il nome che volete assegnare ai vostri dati e sostituire **C:/Rdati/ais.csv** con nome e posizione del file dal quale importare i dati, adeguando al bisogno il separatore dei campi (**sep=";"**) e il separatore dei decimali (**dec=","**) a quelli da voi impiegati;
→ essendo (ad esempio) **dati** il nome dei vostri dati e **var** il nome della variabile che volete analizzare, sostituire nella seconda riga di codice **ais$ferr** con **dati$var**;
→ adattare opportunamente **xlim**, **ylim**, titoli e legende degli assi.

Vale infine la pena di ricordare che, impiegando l'utilità riportata nel post Salvare i grafici di R in un file, potete anche salvare le immagini realizzate con **R**, come quella qui riportata, sotto forma di file .bmp, .jpeg, .png, .pdf, .ps per poterle stampare, archiviare, inserire in una pubblicazione, in un post o in un sito web.

---

[1] Vedere ad esempio la sintesi dei test statistici riportata nel post Tabulare una serie di test di normalità (gaussianità).

[2] Vedere nel post Il set di dati ais come acquisire i dati della tabella **ais** anche senza istallare il pacchetto **DAAG**. La concentrazione della ferritina è espressa in microgrammi per litro di siero (µL).

[3] Da notare che a partire da **R** versione **4.0** le barre dell'istogramma di default sono riempite in colore grigio. Per una trattazione dettagliata, che include anche vari script con rappresentazoni utili nella pratica, vedere il post Istogrammi.

[4] Vedere: *Wikipedia, the free encyclopedia*. Freedman–Diaconis rule. URL consultato il 12/01/2023.

martedì 9 giugno 2020

Inserimento manuale dei dati [3]

Abbiamo già visto alcuni esempi di dati riportati direttamente (cablati) nel codice **R**, dove comunque possono essere modificati a piacere [1, 2]. Una modalità molto elementare ma utile nella pratica quando si debbano inserire pochi dati, come in un test chi-quadrato, in un grafico a torta o quant'altro.

Vediamo ora due nuovi esempi un po' particolari, in quanto prevedono l'**inserimento dei dati in modo interattivo**, cioè da parte di un operatore che deve inserire manualmente i dati nella `Console di R` su richiesta, nel punto previsto dallo script, perché l'esecuzione dello script possa proseguire. Si tratta di una modalità di inserimento dei dati che, mentre consente di familiarizzare con qualche aspetto un po' più tecnico di **R**, può risultare utile in casi particolari.

Copiate lo script che segue quindi incollatelo nella `Console di R` e premete `↲ Invio`:

```
# INSERIMENTO MANUALE DEI DATI [3a]
# inserimento di un singolo dato
#
# predispone la funzione inserisci_un_dato() compresa tra { e }
inserisci_un_dato <- function() {
# legge il dato inserito nella Console di R
datoinserito <- readline("Dato da elaborare = ")
# converte il dato inserito in un numero
datoinserito <- as.numeric(datoinserito)
# calcola la radice quadrata del numero inserito e mostra il risultato
print(paste("La radice quadrata di", datoinserito, "è", sqrt(datoinserito)), quote=FALSE)
}
# esegue la funzione inserisci_un_dato() in modalità interattiva attendendo un input da tastiera
if(interactive()) inserisci_un_dato()
```

Con le prime quattro righe di codice viene predisposta la funzione **inserisci_un_dato**, il cui codice è incluso tra la parentesi graffa **{** e la parentesi graffa **}**. La funzione prevede di acquisire il dato (un solo dato) dalla `Console di R` con **readline()**, trasformare quanto immesso in un numero con **as.numeric()** quindi calcolarne la radice quadrata e mostrare il risultato con la quarta riga di codice.

L'ultima riga di codice con **if(interactive())** manda in esecuzione la funzione **inserisci_un_dato()** in modo interattivo, restando in attesa dell'immissione da tastiera nella Console di R del dato.

I commenti inseriti nel codice **R** sono sufficientemente esplicativi di quanto avviene, mentre la documentazione delle funzioni impiegate può essere visualizzata come al solito digitando **help(***nomedellafunzione***)** nella `Console di R`.

Ora copiate lo script che segue quindi incollatelo nella `Console di R` e premete `↲ Invio`:

```
# INSERIMENTO MANUALE DEI DATI [3b]
# inserimento di più dati
#
# imposta 1 come dato iniziale
datoinserito <- 1
# ripete le istruzioni inserite nel loop fino a quando il dato inserito è diverso da 0 (zero)
while(datoinserito!=0) {
```

```
# richiede di inserire la temperatura in gradi Celsius
datoinserito <- readline("Temperatura in gradi Celsius = ")
# converte il dato inserito in un numero
datoinserito <- as.numeric(datoinserito)
# effettua il calcolo e presenta il risultato
print(paste(datoinserito, "gradi Celsius corrispondono a", ((datoinserito * 1.8) + 32),
"gradi Fahrenheit"), quote=FALSE)
}
```

Mediante la funzione **while()** viene realizzato un loop, il cui codice è incluso tra la parentesi graffa **{** e la parentesi graffa **}**, che viene reiterato fino a quando il dato inserito risulta diverso da 0 (**datoinserito!=0**). Quando il dato inserito è `0` il loop viene terminato.

Il codice prevede che nella `Console di R` sia inserito manualmente da tastiera un dato di temperatura (**datoinserito**) in gradi Celsius (°C), per il quale viene calcolata e mostrata la corrispondente temperatura in gradi Fahrenheit (°F) essendo

$$°F = (°C \cdot 1.8) + 32$$

Possono essere inseriti anche valori negativi di temperatura in gradi Celsius.

Anche in questo caso i commenti inseriti nel codice **R** sono sufficientemente esplicativi di quanto avviene mentre di nuovo al bisogno la documentazione delle funzioni impiegate può essere visualizzata digitando **help(***nomedellafunzione***)** nella `Console di R`.

Per entrambi gli script ci si può inoltre divertire a immaginare i controlli che potrebbero essere eseguiti sul dato inserito, e per il secondo script anche a trovare qualcosa di meglio dell'uscita dal loop piuttosto grezza realizzata inserendo 0 come dato, ma questo va al di la degli scopi del post.

In definitiva i due script, pur nella loro semplicità, forniscono come sottoprodotto utile due esempi didattici:
→ di come con **R** sia possibile **realizzare una funzione**, cioè un blocco di codice che può essere eseguito semplicemente richiamando **nomedellafunzione(***eventualiargomenti***)**;
→ di come con **R** sia possibile **realizzare un loop** (ciclo, iterazione), cioè la ripetizione ciclica di una sequenza di operazioni finché risulta vera la condizione di controllo prefissata, una struttura di controllo ampiamente utilizzata in qualsiasi linguaggio di programmazione, oltre che all'interno delle funzioni di **R**.

---

[1] Vedere il post Inserimento manuale dei dati [1] per l'assemblamento dei vettori in matrici.

[2] Vedere il post Inserimento manuale dei dati [2] per l'assemblamento di vettori in tabelle.

Nota: parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14
6	7	18	12

11	8	17	13
7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

Di fatto i vettori sono matrici aventi una sola riga o una sola colonna. Una matrice con una sola riga e più colonne è detta matrice riga o vettore riga, mentre una matrice con una sola colonna e più righe è detta matrice colonna o vettore colonna.

lunedì 3 agosto 2020

Calcolare la media cumulativa e la mediana cumulativa

Data una sequenza di valori numerici la **media cumulativa** è la media, calcolata in corrispondenza di ciascun dato, di tutti i dati dal primo fino a quello corrente.

Così ad esempio in corrispondenza del dato numero 5 la media cumulativa è la media dei dati da 1 a 5, in corrispondenza del dato numero 6 la media cumulativa è la media dei dati da 1 a 6, in corrispondenza del dato numero 7 la media cumulativa è la media dei dati da 1 a 7 e così via. Lo stesso principio vale per la **mediana cumulativa** che a sua volta è la mediana, calcolata in corrispondenza di ciascun dato, di tutti i dati dal primo fino a quello corrente.

Per il calcolo della media cumulativa e della mediana cumulativa impieghiamo le funzioni contenute nel pacchetto **cumstats**, che deve essere scaricato dal **CRAN**.

Copiate e incollate questo script nella `Console di R` e premete `↲ Invio`:

```
# CALCOLARE LA MEDIA CUMULATIVA E LA MEDIANA CUMULATIVA
#
library(cumstats) # carica il pacchetto
rileva_x <- seq(1:50) # numero progressivo della rilevazione in ascisse
valore_y <- c(5, 9, 12, 7, 8, 13, 26, 7, 8, 19, 10, 7, 5, 36, 12, 18, 31, 17, 16, 43, 7, 12, 9, 27, 15, 18, 24, 29, 32, 23, 29, 12, 7, 48, 34, 45, 19, 12, 16, 18, 22, 31, 38, 7, 11, 4, 38, 36 ,43, 15) # valore osservato corrispondente in ordinate
mydata <- data.frame(rileva_x, valore_y) # combina i vettori in una matrice
#
mydata$mediacum <- cummean(mydata$valore_y) # calcola la media cumulativa e la aggiunge in una nuova colonna
mydata$medianacum <- cummedian(mydata$valore_y) # calcola la mediana cumulativa e la aggiunge in una nuova colonna
mydata # mostra la matrice con i dati definitivi
#
plot(mydata$rileva_x, mydata$valore_y, xlim=c(0,50), ylim=c(0,60), type="l", lty=1, lwd=1, col="black", main="Valori osservati, media cumulativa e mediana cumulativa", xlab="Rilevazioni", ylab="Valori osservati") # grafico dei dati originali
lines(mydata$rileva_x, mydata$mediacum, xlim=c(0,50), ylim=c(0,60), type="l", lty=2, lwd=1, col="red") # sovrappone grafico della media cumulativa
lines(mydata$rileva_x, mydata$medianacum, xlim=c(0,50), ylim=c(0,60), type="l", lty=2, lwd=1, col="blue") # sovrappone grafico della mediana cumulativa
#
legend(0,60, legend=c("Valori osservati nelle rilevazioni", "Media cumulativa dei valori osservati", "Mediana cumulativa dei valori osservati"), col=c("black", "red", "blue"), lty=c(1,2,2), lwd=c(1,1,1), cex=0.8) # aggiunge la legenda
#
```

Dopo avere caricato il pacchetto **cumstats** i dati impiegati come esempio sono generati in questo modo:
→ nel vettore **rileva_x** viene riportato il numero progressivo della rilevazione;
→ nel vettore **valore_y** viene riportato il valore numerico osservato in corrispondenza di ciascuna rilevazione, valore numerico sul quale calcoleremo la media cumulativa e la mediana cumulativa.
→ con la funzione **dataframe()** i due vettori sono combinati nella matrice **mydata** [1].

A questo punto alla matrice **mydata** sono aggiunte due colonne:
→ la colonna (vettore) **mediacum** nella quale è riportata la media cumulativa calcolata in corrispondenza di ogni dato con la funzione **cummmean()**;

→ la colonna (vettore) **medianacum** nella quale è riportata la mediana cumulativa calcolata in corrispondenza di ogni dato con la funzione **cummedian()**.

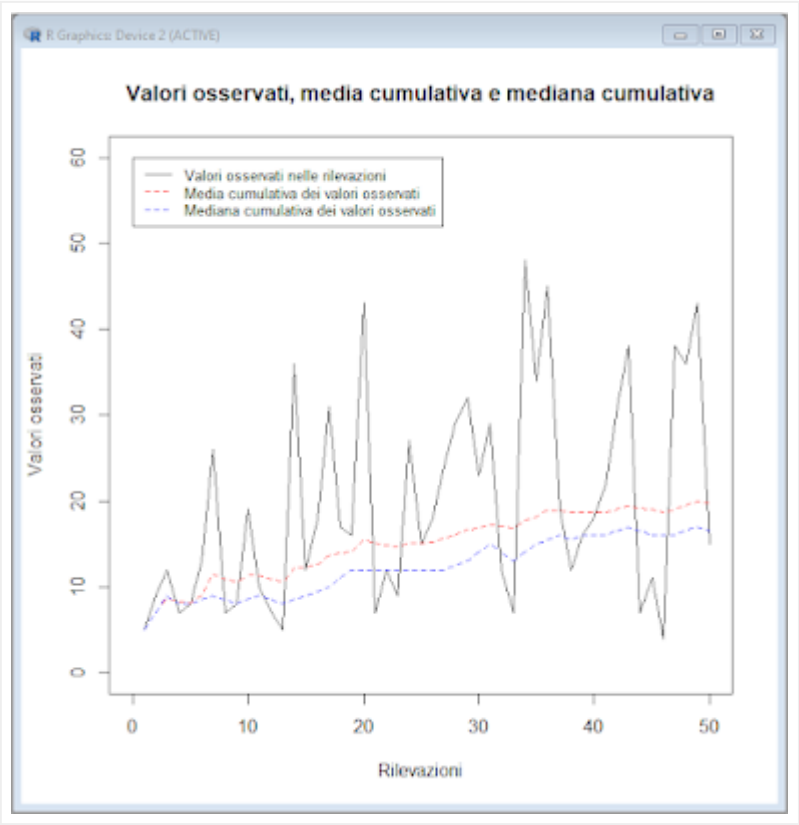
Per i dettagli delle funzioni impiegate digitare **help(***nomedellafunzione***)** nella `Console di R`. Al bisogno potete consultare il manuale del pacchetto **cumstats** [2].

Non resta quindi che mostrare il contenuto della matrice **mydata** nella versione definitiva contenente il numero della rilevazione (`rileva_x`), il valore osservato (`valore_y`) e le rispettive media cumulativa (`mediacum`) e mediana cumulativa (`medianacum`)

```
> mydata # mostra la matrice con i dati definitivi
   rileva_x valore_y  mediacum medianacum
1         1        5  5.000000        5.0
2         2        9  7.000000        7.0
3         3       12  8.666667        9.0
4         4        7  8.250000        8.0
5         5        8  8.200000        8.0
6         6       13  9.000000        8.5
7         7       26 11.428571        9.0
8         8        7 10.875000        8.5
9         9        8 10.555556        8.0
10       10       19 11.400000        8.5
11       11       10 11.272727        9.0
12       12        7 10.916667        8.5
13       13        5 10.461538        8.0
14       14       36 12.285714        8.5
15       15       12 12.266667        9.0
16       16       18 12.625000        9.5
17       17       31 13.705882       10.0
18       18       17 13.888889       11.0
19       19       16 14.000000       12.0
20       20       43 15.450000       12.0
21       21        7 15.047619       12.0
22       22       12 14.909091       12.0
23       23        9 14.652174       12.0
24       24       27 15.166667       12.0
25       25       15 15.160000       12.0
26       26       18 15.269231       12.0
27       27       24 15.592593       12.0
28       28       29 16.071429       12.5
29       29       32 16.620690       13.0
30       30       23 16.833333       14.0
31       31       29 17.225806       15.0
32       32       12 17.062500       14.0
33       33        7 16.757576       13.0
34       34       48 17.676471       14.0
35       35       34 18.142857       15.0
36       36       45 18.888889       15.5
37       37       19 18.891892       16.0
38       38       12 18.710526       15.5
39       39       16 18.641026       16.0
40       40       18 18.625000       16.0
41       41       22 18.707317       16.0
42       42       31 19.000000       16.5
43       43       38 19.441860       17.0
44       44        7 19.159091       16.5
```

```
45        45        11 18.977778        16.0
46        46         4 18.652174        16.0
47        47        38 19.063830        16.0
48        48        36 19.416667        16.5
49        49        43 19.897959        17.0
50        50        15 19.800000        16.5
```

che sono impiegati per la successiva rappresentazione grafica



che viene realizzata con tre sole funzioni:
→ la funzione **plot()** che traccia il grafico del valore osservato **mydata$valore_y** in corrispondenza di ciascuna rilevazione **mydata$rileva_x** impiegando una linea (**type="l"**) continua (**lty=1**) di larghezza unitaria (**lwd=1**) e di colore nero (**col="black"**);
→ la funzione **lines()** che viene impiegata una prima volta per sovrapporre il grafico della media cumulativa con una linea tratteggiata (**lty=2)** di colore rosso (**col="red"**) e una seconda volta per sovrapporre il grafico della mediana cumulativa con una linea tratteggiata (**lty=2**) di colore blu (**col="blue"**)
→ la funzione **legend()** che riporta la legenda esplocativa dei grafici [3].

Ovviamente lo script può essere riadattato per rappresentare dati differenti ma anche per riportare solamente l'uno o l'altro dei grafici.

---

[1] Parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14
6	7	18	12
11	8	17	13
7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

[2] Vedere la voce *Reference manual* del pacchetto cumstats: Cumulative Descriptive Statistics. URL consultato il 02/01/2023.

[3] Per un approfondimento sull'impiego della funzione **legend()** rimando al post Aggiungere la legenda a un grafico.

giovedì 13 agosto 2020

Tabulare una serie di test di normalità (gaussianità)

Quando con **R** si esegue una serie di **test di normalità (gaussianità)** su una variabile, può essere fastidioso consultare e confrontare tra loro risultati che nella `Console di R` sono riportati separatamente e lontani l'uno dall'altro. Personalmente trovo che gli stessi risultati compattati in una tabella e incolonnati siano molto meglio fruibili. Ma non solo. I valori di probabilità `p` incolonnati sono poi ancor meglio confrontabili tra loro se riportati in formato fisso anziché in formato esponenziale.

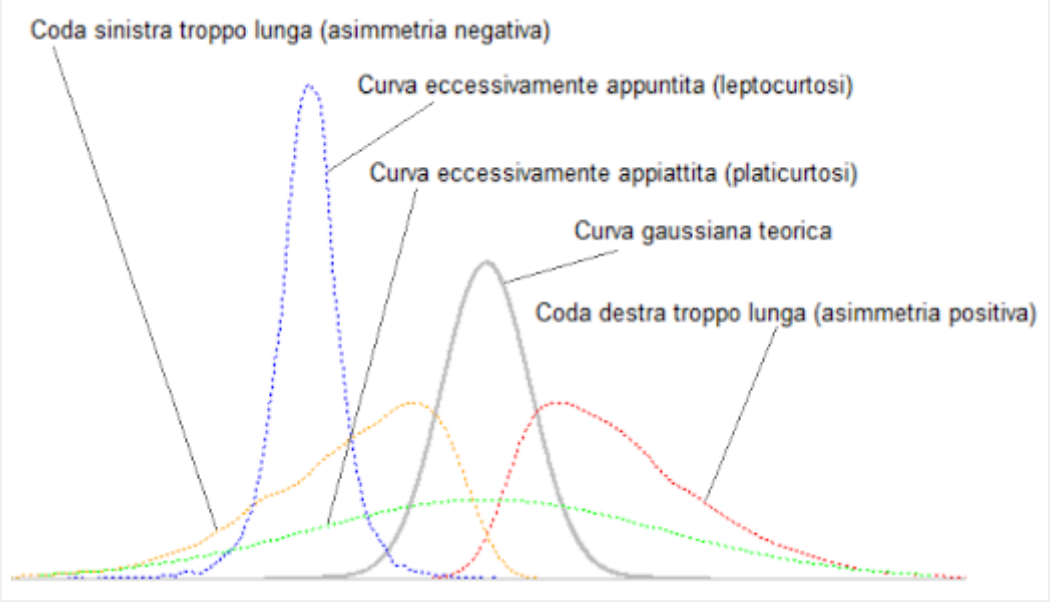
Vediamo quindi come incolonnare e riportare in formato fisso i risultati dei più comuni test di gaussianità con lo script che segue. Accertatevi di avere installati i pacchetti **moments** e **nortest**. Altrimenti dovete scaricarli e installarli dal **CRAN**. Accertatevi inoltre di avere installato il pacchetto **DAAG** che contiene i dati impiegati nello script, o in alternativa procedete come indicato [1].

Copiate lo script, incollatelo nella `Console di R` e premete `↵ Invio`.

```
# TABULARE UNA SERIE DI TEST DI NORMALITA' (GAUSSIANITA')
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
mydata <- ais[,c(5)] # salva in mydata la colonna 5 con la variabile ferritina
#
# asimmetria, curtosi e quattro test di gaussianità
#
library(moments) # carica il pacchetto
agostino.test(mydata) # test di D'Agostino per il coefficiente di asimmetria
anscombe.test(mydata) # test di Anscombe-Glynn per il coefficiente di curtosi
library(nortest) # carica il pacchetto
ad.test(mydata) # test di Anderson-Darling per la gaussianità
cvm.test(mydata) # test di Cramer-von Mises per la gaussianità
pearson.test(mydata) # test chi-quadrato di Pearson per la gaussianità
sf.test(mydata) # test di Shapiro-Francia per la gaussianità
#
# sintetizza in una tabella i test con i risultati e i valori di p
#
stat <- c(names(agostino.test(mydata)$statistic[2]),
names(anscombe.test(mydata)$statistic[2]), names(ad.test(mydata)$statistic),
names(cvm.test(mydata)$statistic), names(pearson.test(mydata)$statistic),
names(sf.test(mydata)$statistic)) # array con le denominazioni delle statistiche calcolate
#
ris <- c(agostino.test(mydata)$statistic[2], anscombe.test(mydata)$statistic[2],
ad.test(mydata)$statistic, cvm.test(mydata)$statistic, pearson.test(mydata)$statistic,
sf.test(mydata)$statistic) # array con i risultati delle statistiche calcolate
#
pval <- c(agostino.test(mydata)$p.value, anscombe.test(mydata)$p.value,
ad.test(mydata)$p.value, cvm.test(mydata)$p.value, pearson.test(mydata)$p.value,
sf.test(mydata)$p.value) # array con i valori di p
#
mydataset <- data.frame(stat, ris, pval) # combina gli array in una tabella
#
rownames(mydataset) <- c(agostino.test(mydata)$method,
anscombe.test(mydata)$method, ad.test(mydata)$method, cvm.test(mydata)$method,
pearson.test(mydata)$method, sf.test(mydata)$method) # aggiunge i nomi delle righe
```

colnames(mydataset) <- c("Statistica", "Risultato", "Valore p") # aggiunge i nomi delle colonne
mydataset # mostra la tabella
options(scipen=999) # esprime i numeri in formato fisso
mydataset # mostra nuovamente la tabella
options(scipen=0) # ripristina la notazione scientifica/esponenziale
#

Il **test di asimmetria** di D'Agostino **agostino.test()** valuta quanto e come la simmetria della distribuzione campionaria (quella effettivamente osservata) si discosta da quella della distribuzione gaussiana teorica, che è per definizione perfettamente simmetrica. Il **test di curtosi** di Anscombe-Glynn **anscombe.test()** valuta se la distribuzione campionaria è troppo appiattita (bassa e larga) o troppo appuntita (alta e stretta) rispetto alla distribuzione gaussiana teorica, in ciascun punto della quale vale un preciso rapporto tra distanza dal centro della distribuzione e altezza della curva.



Mentre i due test precedenti valutano separatamente asimmetria e curtosi, gli altri quattro test qui impiegati, il test di Anderson-Darling **ad.test()**, il test di Cramer-von Mises **cvm.test()**, il test chi-quadrato di Pearson **pearson.test()** e il test di Shapiro-Francia **sf.test()**, sono **test globali di gaussianità** (forniscono una valutazione globale del grado di scostamento della distribuzione osservata dalla distribuzione gaussiana teorica, ma non del tipo di scostamento).

Dopo avere calcolato e riportato separatamente i risultati dei sei test, viene generata la tabella che mostra per ciascun test la denominazione, il simbolo della statistica calcolata, il risultato numerico di detta statistica e il corrispondente valore di probabilità p procedendo in questo modo:
→ per ciascun test viene riportato nell'array (o se preferite vettore) **stat** il simbolo della statistica calcolata **names()**;
→ per ciascun test viene riportato nell'array **ris** il risultato numerico della statistica calcolata **$statistic**;
→ per ciascun test viene riportato nell'array **pval** il valore di probabilità p del risultato numerico **$p.value**;
→ mediante la funzione **data.frame()** gli array **stat**, **ris** e **pval** sono combinati nella tabella **mydataset**;
→ mediante la funzione **rownames()** a ciascuna riga della tabella **mydataset** viene assegnata la denominazione del test corrispondente **$method**;
→ mediante la funzione **colnames()** sono assegnati i nomi alle colonne della tabella **mydataset** che contengono, nell'ordine, il simbolo **stat** della statistica (`Statistica`), il risultato **ris** della statistica (`Risultato`) e il valore **pval** di probabilità p (`Valore p`) [2].

La tabella viene quindi mostrata nella `Console di R` prima con i valori numerici espressi in notazione scientifica

```
                                 Statistica  Risultato     Valore p
D'Agostino skewness test                  z  6.1376120 8.377114e-10
Anscombe-Glynn kurtosis test              z  2.9448982 3.230609e-03
Anderson-Darling normality test           A  6.0970351 4.903750e-15
Cramer-von Mises normality test           W  0.9447263 2.495032e-09
Pearson chi-square normality test         P 63.7722772 2.530969e-08
Shapiro-Francia normality test            W  0.8913765 1.251172e-09
```

e poi con i valori numerici espressi in notazione fissa mediante **options(scipen=999)**

```
                                 Statistica  Risultato               Valore p
D'Agostino skewness test                  z  6.1376120 0.0000000083771142715
Anscombe-Glynn kurtosis test              z  2.9448982 0.0032306945215406870
Anderson-Darling normality test           A  6.0970351 0.0000000000000490375
Cramer-von Mises normality test           W  0.9447263 0.0000000249503195167
Pearson chi-square normality test         P 63.7722772 0.0000002530968668914
Shapiro-Francia normality test            W  0.8913765 0.0000000125117173937
```

L'incolonnamento e il formato fisso rendono sicuramente più chiara la sintesi e la valutazione comparativa dei risultati numerici. La probabilità `p` è la probabilità di osservare per caso il risultato numerico della statistica (`z`, `A`, `W`, `P` secondo i casi) che misura lo scostamento della distribuzione osservata dalla distribuzione gaussiana teorica. Se tale probabilità è sufficientemente piccola, si conclude che il risultato ottenuto non è imputabile al caso e che pertanto la distribuzione dei dati si discosta significativamente dalla distribuzione gaussiana. Tutti e sei i test confermano che la ferritina non segue una distribuzione gaussiana: infatti abbiamo un valore $p < 0.05$ - cioè un p inferiore al valore del 5% tradizionalmente assunto come soglia per la significatività - in tutti i casi, e addirittura di molti ordini di grandezza inferiore al 5% in cinque casi su sei. Potete avere una conferma visiva di queste conclusioni con le rappresentazioni grafiche, sempre altamente consigliate come complemento all'analisi statistica, che potete realizzare con lo script riportato nel post Analizzare graficamente la distribuzione di una variabile.

Come al solito lo script è realizzato in modo da essere facilmente riutilizzabile, essendo sufficiente a questo scopo:
→ sostituire la prima riga di codice con il codice per importare i vostri dati [3];
→ sostituire nella seconda riga di codice **ais[,c(5)]** con il nome della variabile da analizzare contenuta nei dati che avete importato.

---

[1] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto DAAG.

[2] Potete al bisogno approfondire la documentazione delle funzioni qui impiegate digitando **help(***nomedellafunzione***)** nella `Console di R`.

[3] Per una guida rapida all'importazione dei dati potete consultare i post:
→ importazione dei dati da un file .csv
→ importazione dei dati da un file .xls o .xlsx
→ gestione dei dati mancanti

domenica 4 ottobre 2020

Grandezze e unità di misura

Se è vero che la **statistica** è la tecnica che consente di fornire un supporto scientifico alle **evidenze** fornite dalle **misure**, ci sono dal punto di vista metodologico, nelle misure, quattro aspetti sull'importanza dei quali non mi stancherò mai di richiamare l'attenzione:
→ i presupposti per l'impiego alternativo di scale nominali, scale ordinali e scale numeriche [1];
→ l'utilizzo di un sistema di unità di misura razionale, documentato e condiviso - oltre che adottato per legge anche in Italia - quale è il *Sistema internazionale di unità (SI)*, applicandone integralmente e senza eccezioni le indicazioni [2, 3];
→ la necessità di arrotondare i risultati al giusto numero di cifre significative [4];
→ il rispetto delle regole per rappresentare i numeri che indicano come separare i decimali e come raggruppare le cifre [5].

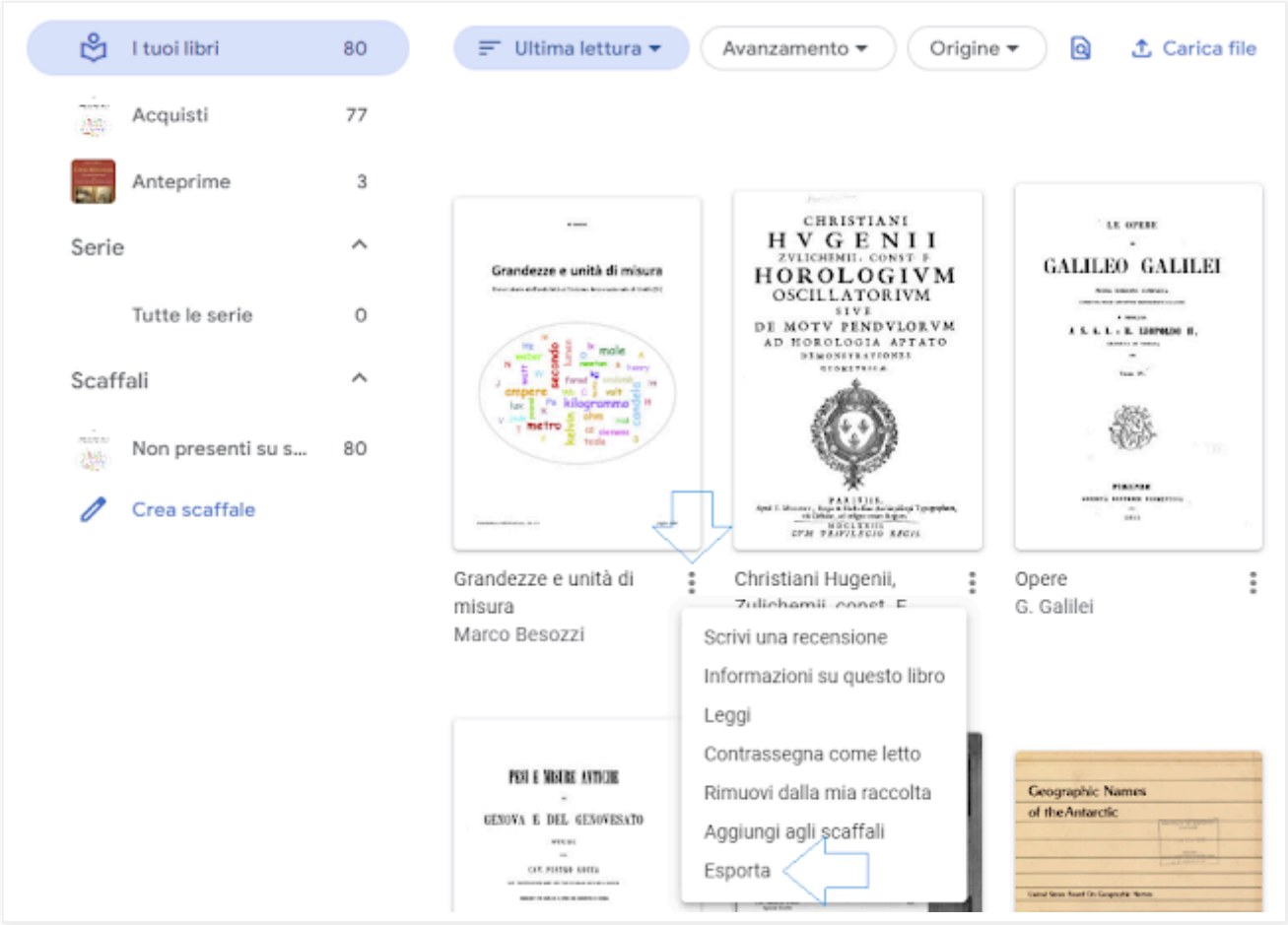
Qualcuno potrà obiettare che nell'ambito della statistica si tratta di aspetti marginali. Sarà anche vero: tuttavia sono convinto della necessità di richiamare l'attenzione su questi passaggi in quanto, essendo dati per ovvi, scontati e "intuitivi", senza il minimo di approfondimento che richiederebbero sono ancor oggi fonte di inesattezze tanto banali quanto fastidiose, che possono essere facilmente evitate seguendo le indicazioni delle fonti che ho riportato.

Aggiungo qualche parola in più in merito a grandezze e unità di misura. Discendente diretto del **sistema metrico decimale** originato dalla rivoluzione francese (fondato su metro e kilogrammo) e del successivo **sistema MKSA** (fondato su metro, kilogrammo, secondo, ampere e conosciuto anche come Sistema Giorgi in onore del proponente italiano Giovanni Giorgi), il **Sistema internazionale di unità (SI)**, accettato dalla Comunità Economica Europea (CEE) nel 1980 e divenuto legale in Italia nel 1982, il giorno 20 maggio 2019 ha visto cambiare le definizioni di chilogrammo, ampere, kelvin e mole [2]. Lo scopo di questo cambiamento è basare le *sette unità di misura fondamentali del SI* sulle leggi della fisica, dando loro un fondamento solido, immutabile nel tempo e nello spazio.

Ho riassunto brevemente la storia delle unità di misura, dall'antichità al cambiamento epocale del SI avvenuto il 20 maggio 2019, in un *ebook*, ovviamente gratuito, disponibile facendo click su questa immagine del logo del nuovo SI

Per scaricare il libro in formato *.pdf*, dopo avere aperto il libro con l'opzione *Leggi*:
→ fate click sull'iconcina in alto a sinistra per accedere a *I tuoi libri* dove trovate il libro;

→ fate click sui *tre puntini disposti verticalmente* in basso a destra sulla copertina del libro;
→ fate click su *Esporta* e infine selezionate *Esporta come PDF*.



---

[1] Vedere il post Scale nominali, scale ordinali e scale numeriche

[2] Bureau International des Poids et Mesures. *The International System of Units (SI)*. URL consultato il 3/10/2020: https://bit.ly/30tZOiq

[3] Bureau International des Poids et Mesures. *SI Brochure*. URL consultato il 3/10/2020: https://bit.ly/3le9aXR

[4] Vedere il post Arrotondare al giusto numero di cifre significative

[5] Vedere il post Come separare i decimali e come raggruppare le cifre

mercoledì 25 novembre 2020

Grafici a punti (dotplot)

I **grafici a punti (dotplot)** rappresentano una interessante alternativa ai *grafici a scatola con i baffi (boxplot)* [1] nei casi di distribuzioni univariate che includono pochi dati e per le quali non volete perdere il dettaglio dei singoli valori.

Possono inoltre essere sovrapposti sia ai boxplot sia ai *grafici a violino (violin plot)* [2] a testimonianza del fatto che si tratta di viste dei dati diverse ma tra loro integrate.

Per eseguire i due script che seguono è necessario scaricare dal **CRAN** il pacchetto **ggplot2** [3] e il pacchetto **gridExtra** [4]. Accertatevi inoltre di avere già installato il pacchetto **DAAG** che contiene i dati impiegati nello script, o in alternativa procedete seguendo le indicazioni riportate in [5].

Copiate questo primo script, incollatelo nella `Console di R` e premete `↵ Invio`.

```
# UN GRAFICO A PUNTI (DOTPLOT)
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
#
ggplot(ais, aes(x=sport, y=hg, fill=sport)) + geom_dotplot(method="dotdensity",
binaxis='y', stackdir="center", stackratio=1, dotsize=1, binwidth=0.2, show.legend=FALSE)
+ coord_cartesian(ylim=c(10, 20)) + labs(title="Concentrazione dell'emoglobina nel
sangue per sport praticato", x="Sport praticato", y="Emoglobina (g/dL)") +
theme_classic() # traccia dotplot dell'emoglobina per sport
#
```

Dopo avere caricato i pacchetti necessari, il grafico a punti (dotplot) viene realizzato impiegando una sola riga di codice che include cinque funzioni concatenate:
→ la funzione **ggplot()** che specifica i dati da impiegare (**ais**) e con la funzione **aes()** specifica di realizzare il grafico a punti separatamente per ciascuno sport praticato (**x=sport**) per la variabile emoglobina (**y=hg**);
→ la funzione **geom_dotplot()** che di fatto realizza il grafico con una serie piuttosto articolata di argomenti per il dettaglio dei quali rimando alla documentazione della funzione inclusa nel manuale di riferimento del pacchetto [3];
→ la funzione **coord_cartesian()** che viene impiegata per stabilire (**ylim=**) valore minimo e valore massimo pare rappresentare la distribuzione dei punti sull'asse delle y;
→ la funzione **labs()** che banalmente riporta titolo ed etichette degli assi;
→ la funzione **theme_classic()** che definisce "il look" del grafico.

In generale per realizzare un grafico a punti (dotplot) è necessario avere una variabile numerica [`valore`] che contiene i valori osservati più una seconda variabile qualitativa che contiene un criterio di classificazione [`fattore`] per aggregare i valori in sottoinsiemi, come qui schematicamente esemplificato

valore	fattore
4	A
2	B
5	B
4	A
3	C
5	A
2	C

```
7         B
...       ...
```

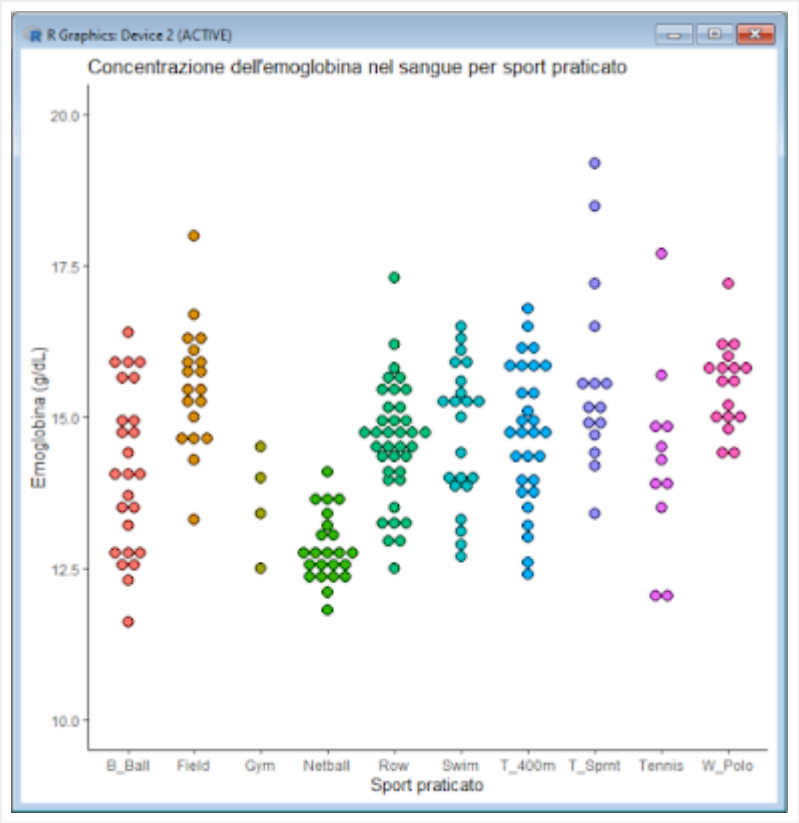
e dove nel caso specifico il `valore` è la concentrazione di emoglobina contenuta nella variabile **hg** (riportata sull'asse verticale **y**) e il `fattore` è lo sport praticato contenuto nella variabile **sport** (riportata sull'asse orizzontale **x**).

Se guardate alla struttura dell'oggetto **ais** digitando

str(ais)

notate che i dati impiegati per realizzare il grafico a punti sono strutturati proprio in questo modo.

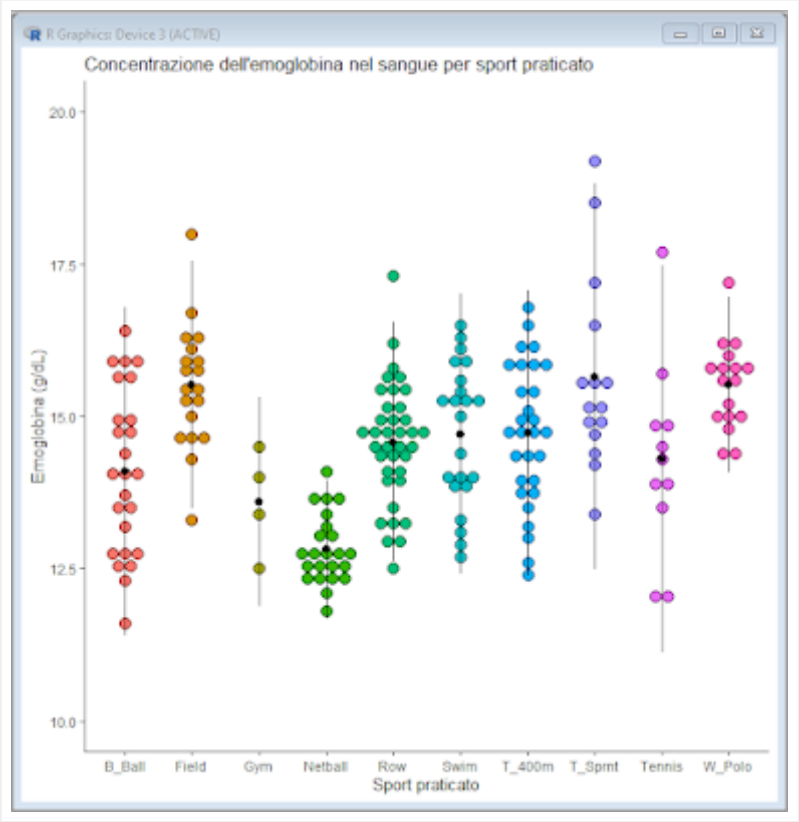
E questo è il grafico ottenuto con lo script.



Ora copiate e incollate nella `Console di R` queste due altre righe di codice che aprono una nuova finestra grafica nella quale viene rappresentato lo stesso grafico, questa volta aggiungendo (**+**) al codice riportato in precedenza la funzione **stat_summary()** che sovrappone ai punti la media e l'intervallo corrispondente a due (**mult=2**) deviazioni standard:

```
#
windows() # apre e inizializza una nuova finestra grafica
ggplot(ais, aes(x=sport, y=hg, fill=sport)) + geom_dotplot(method="dotdensity",
binaxis='y', stackdir="center", stackratio=1, dotsize=1, binwidth=0.2,
show.legend=FALSE) + coord_cartesian(ylim=c(10, 20)) + labs(title="Concentrazione
dell'emoglobina nel sangue per sport praticato", x="Sport praticato", y="Emoglobina
(g/dL)") + theme_classic() + stat_summary(fun.data=mean_sdl, fun.args = list(mult=2),
geom="pointrange", color="black", show.legend=FALSE) # aggiunge media +/- 2 deviazioni
standard
#
```

Questo è il grafico.

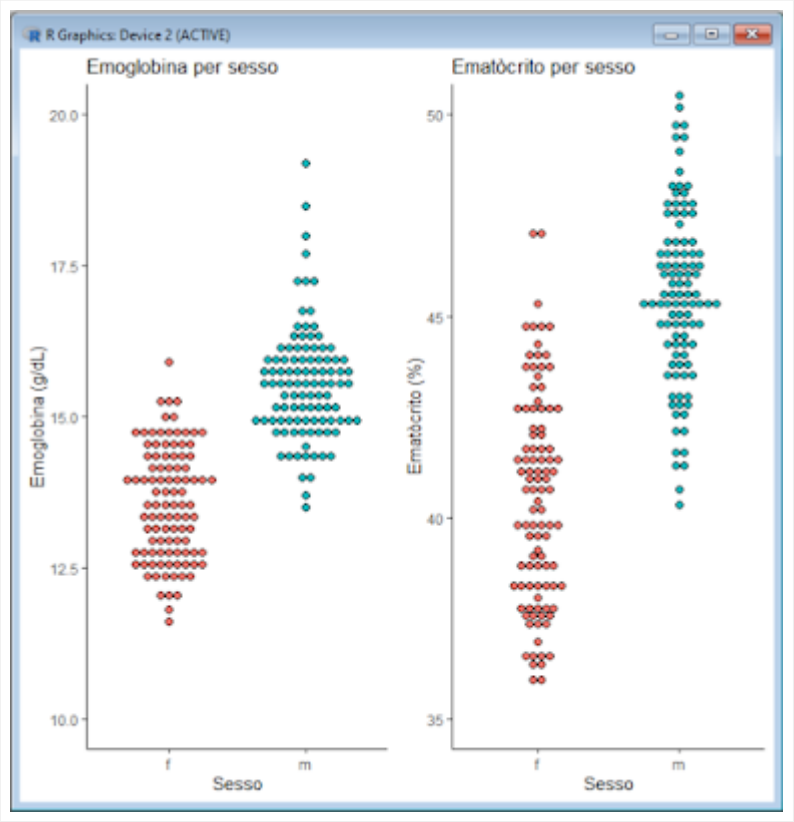


Copiate quest'altro script, incollatelo nella `Console di R` e premete `↵ Invio`.

```
# DUE GRAFICI A PUNTI (DOTPLOT) AFFIANCATI
#
library(DAAG) # carica il pacchetto DAAG incluso il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
#
plot1 <- ggplot(ais, aes(x=sex, y=hg, fill=sex)) + geom_dotplot(method="dotdensity", binaxis='y', stackdir="center", stackratio=1, dotsize=0.7, binwidth=0.2, show.legend=FALSE) + coord_cartesian(ylim=c(10, 20)) + labs(title="Emoglobina per sesso", x="Sesso", y="Emoglobina (g/dL)") + theme_classic() # dotplot dell'emoglobina per sesso
#
plot2 <- ggplot(ais, aes(x=sex, y=hc, fill=sex)) + geom_dotplot(method="dotdensity", binaxis='y', stackdir="center", stackratio=1, dotsize=1, binwidth=0.2, show.legend=FALSE) + coord_cartesian(ylim=c(35, 50)) + labs(title="Ematòcrito per sesso", x="Sesso", y="Ematòcrito (%)") + theme_classic() # dotplot dell'ematòcrito per sesso
#
library(gridExtra) # carica il pacchetto con la funzione necessaria per realizzare i grafici affiancati
grid.arrange(plot1, plot2, nrow = 1) # i due grafici sono affiancati orizzontalmente
#
```

Il codice di fatto è sempre lo stesso tranne che per due aspetti:
→ sono realizzati due grafici a punti - per la variabile **hg** e per la variabile **hc** [5] - separando i dati per sesso (**x=sex**);
→ per affiancare i due grafici, dell'emoglobina (**hg**) per sesso e dell'ematòcrito (**hc**) per sesso, viene impiegata la funzione **grid.arrange()** inclusa nel pacchetto **gridExtra** [3] che consente di superare i limiti della analoga funzione **par(mfrow=...)** del *pacchetto base* di **R** [6].

Questo è il risultato.



Per adattare gli script ai vostri dati dovete:
→ sostituire la prima riga con il codice necessario per importare i vostri dati;
→ nella funzione **ggplot()** sostituire **ais** con il nome dell'oggetto che contiene i vostri dati;
→ nella funzione **aes()** sostituire in **x** il nome della variabile (**sex**) con quello della variabile (fattore) che volete impiegare per raggruppare i vostri valori in sottoinsiemi;
→ nella funzione **aes()** sostituire in **y** il nome della variabile (**hg**) con quello della variabile numerica che contiene i vostri valori;
→ nella funzione **labs()** adattare opportunamente titolo, nome del fattore e nome della variabile.

Ricordate infine che:
→ potete visualizzare il kernel density plot dei dati sovrapponendo ai grafici a punti i grafici a violino [2];
→ potete salvare le immagini realizzate con **R** sotto forma di file `.bmp`, `.jpeg`, `.png`, `.pdf`, `.ps` per stamparle, archiviarle, inserirle in una pubblicazione, in un post o in un sito web, impiegando il codice riportato nel post Salvare i grafici di R in un file.

Se siete interessati a rappresentazioni grafiche più elaborate di quelle presenti nel pacchetto base di **R** potrebbe esservi utile approfondire le funzioni del pacchetto **ggplot2** [3].

---

[1] Vedere il post Grafici a scatola con i baffi (boxplot).

[2] Vedere il post Grafici a violino (violin plot).

[3] Il manuale di riferimento *Package 'ggplot2'* lo trovate facendo click sul link *Reference manual* nella pagina di documentazione del pacchetto. URL consultato il 2/10/2020: https://bit.ly/3ioEJfE

[4] Il manuale di riferimento *Package 'gridExtra'* lo trovate facendo click sul link *Reference manual* nella relativa pagina di documentazione del pacchetto. URL consultato il 2/10/2020: https://bit.ly/36qc9Ip

[5] Vedere il post Il set di dati ais e le indicazioni riportate alla pagina Dati.

[6] Alcuni esempi di impiego della funzione **par(mfrow=...)** sono riportati nel post Inserire più grafici nella stessa immagine.

domenica 7 febbraio 2021

Grafici a violino (violin plot)

I **grafici a violino (violin plot)** prendono lo spunto dai grafici a scatola con i baffi (boxplot) riportando per una distribuzione univariata, al posto delle classiche scatole, il profilo di densità dei valori osservati sotto forma di kernel density plot [1].

La forma tipica dei grafici, che dà loro il nome, deriva dal fatto che il kernel density plot dei dati è riportato simmetricamente da entrambi i lati della distribuzione. Mentre interessante, come vedremo tra poco, è che a un violin plot possono essere sovrapposti al bisogno un *grafico a scatola con i baffi (boxplot)* [2] oppure un *grafico a punti (dotplot)* [3], a testimonianza del fatto che si tratta di viste dei dati diverse ma tra loro integrate e complementari.

Come dati impieghiamo la concentrazione degli eritrociti (globuli rossi) nel sangue rilevata in 202 atleti australiani riportata nella colonna/variabile **rcc** della tabella **ais** inclusa nel pacchetto **DAAG**. Accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [4]. Se non volete installare il pacchetto e avete scaricato il file di dati `ais.csv` sostituite nella prima riga dello script

library(DAAG)

con

ais <- read.table("c:/Rdati/ais.csv", header=TRUE, sep=";", dec=",")

Dovete scaricare dal **CRAN** anche il pacchetto **ggplot2** che realizza la grafica [5].

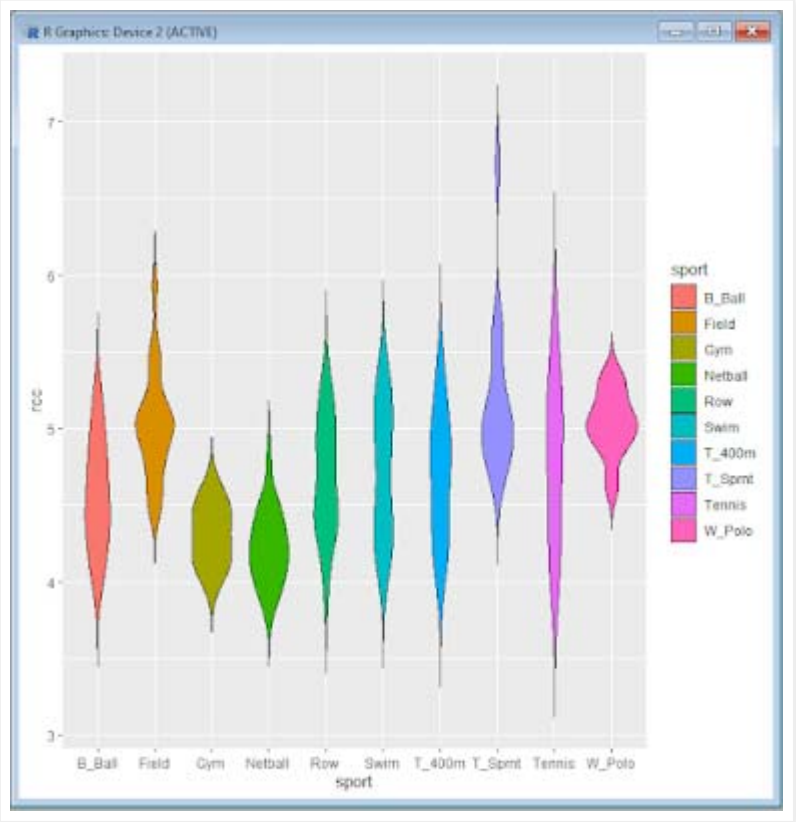
Il tema è stato suddiviso in sei script, corrispondenti ad altrettanti modi di rappresentazione, tra i quali scegliere quello preferito. Copiate questo primo script, incollatelo nella `Console di R` e premete `↵ Invio`.

```
# GRAFICI A VIOLINO (VIOLIN PLOT)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
windows() # apre una nuova finestra
#
# violin plot riportati per intero
ggplot(ais, aes(x=sport, y=rcc, fill=sport)) + geom_violin(trim=FALSE)
#
```

La seconda riga di codice con **library(ggplot2)** carica il pacchetto che contiene le funzioni necessarie per realizzare i grafici a violino (violin plot).

Dopo (terza riga) avere aperto e inizializzato con la funzione **windows()** una nuova finestra grafica, arriviamo finalmente all'unica riga di codice necessaria per realizzare il nostro primo grafico a violino mediante due funzioni concatenate:
→ la funzione **ggplot()** che specifica come primo argomento i dati da impiegare (**ais**) e come secondo argomento impiega la funzione **aes()** per realizzare il grafico per la variabile eritrociti (**y=rcc**) separatamente per ciascuno sport praticato (**x=sport**), riempiendo i grafici con un colore differente per ciascuno sport (**fill=sport**);
→ la funzione **geom_violin** che realizza il grafico riportando per intero il kernel density plot calcolato sui dati.

Questo è quindi il primo grafico:



Minimizzate la finestra ma non chiudetela: vi sarà utile riaprirla per confrontare questo primo grafico con il secondo e con i successivi grafici.

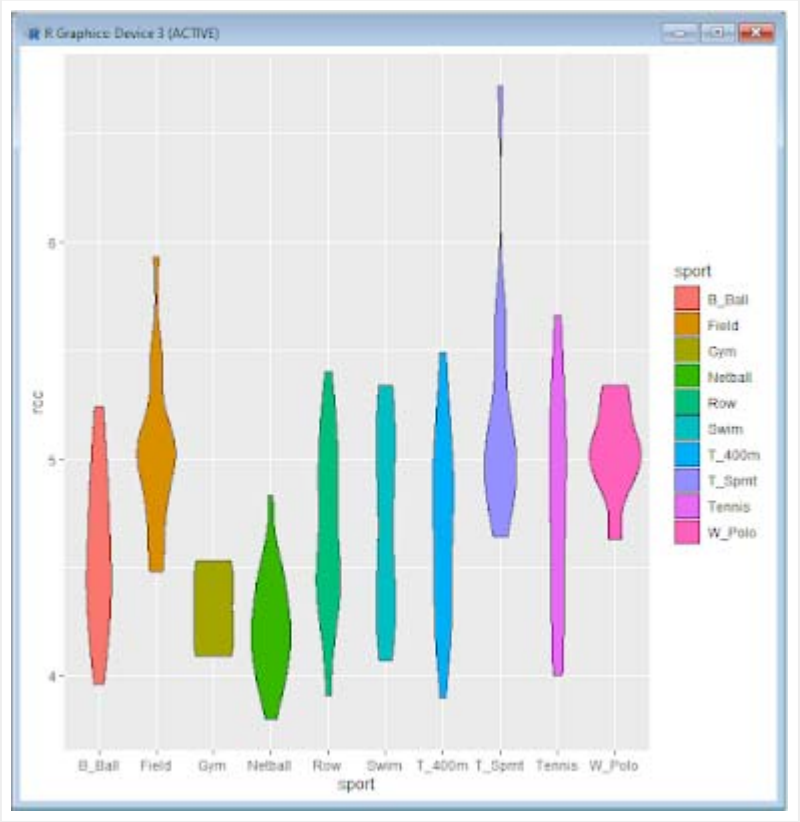
Ora copiate e incollate nella `Console di R` il secondo script e premete `↲ Invio`.

```
# GRAFICI A VIOLINO (VIOLIN PLOT)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
windows() # apre una nuova finestra
#
# violin plot troncati ai valori minimo e massimo osservati
ggplot(ais, aes(x=sport, y=rcc, fill=sport)) + geom_violin(trim=TRUE)
#
```

Ponendo nella funzione **geom_violin()** l'argomento **trim=TRUE** il kernel density plot/violin plot viene ora *troncato* in corrispondenza del valore minimo e del valore massimo osservati, o se preferite viene *tracciato solamente* all'interno del range/intervallo dei valori osservati.

Questo corrisponde ad una regola aurea in quanto in statistica (ma non solo in statistica) effettuare una estrapolazione, ovvero andare oltre i dati, quindi andare al di la dell'informazione in nostro possesso, significa addentrarsi in un terreno molto scivoloso.

Ed ecco il secondo grafico:



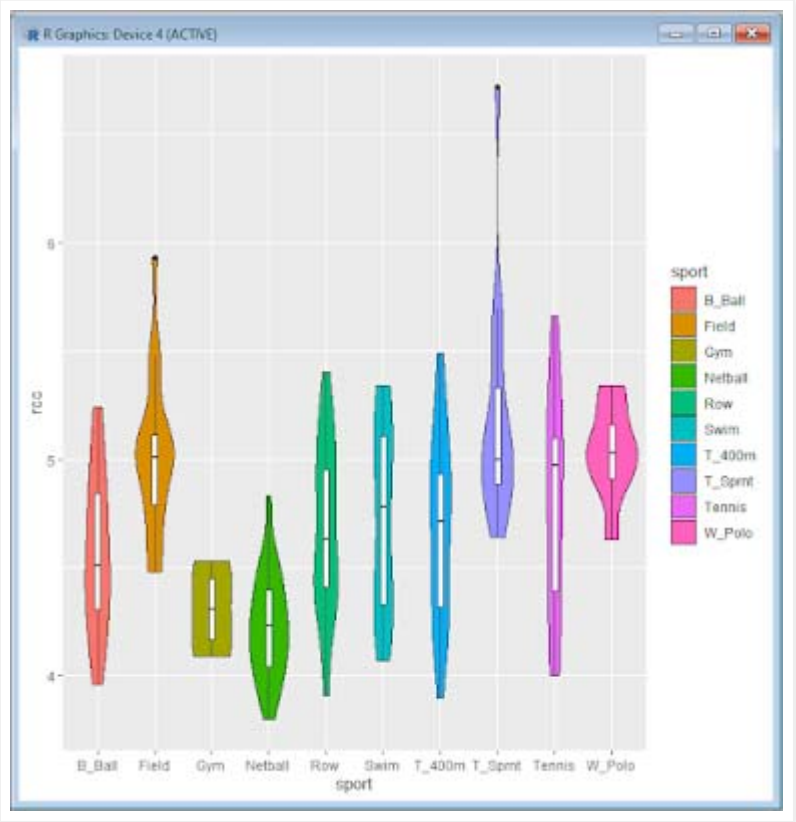
Ora copiate e incollate nella `Console di R` questo terzo script e premete `↵ Invio`.

```
# GRAFICI A VIOLINO (VIOLIN PLOT)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
windows() # apre una nuova finestra
#
# violin plot con sovrapposto grafico a scatola con i baffi
ggplot(ais, aes(x=sport, y=rcc, fill=sport)) + geom_violin(trim=TRUE) +
geom_boxplot(width=0.1, fill="white")
#
```

Qui è stata aggiunta la funzione **geom_boxplot()** che sovrappone al grafico a violino/violin plot un *grafico a scatola con i baffi (boxplot)* specificando di rappresentare le scatole dei grafici:
→ con **width="0.1"** che definisce la larghezza delle scatole/box;
→ con **fill="white"** che specifica il colore di riempimento delle scatole/box.

Da notare che sono stati impiegati gli stessi dati utilizzati nell'esempio che illustra i grafici a scatola con i baffi [2] proprio per consentirvi di effettuare un immediato confronto tra le due rappresentazioni grafiche.

Questo è il terzo grafico realizzato:

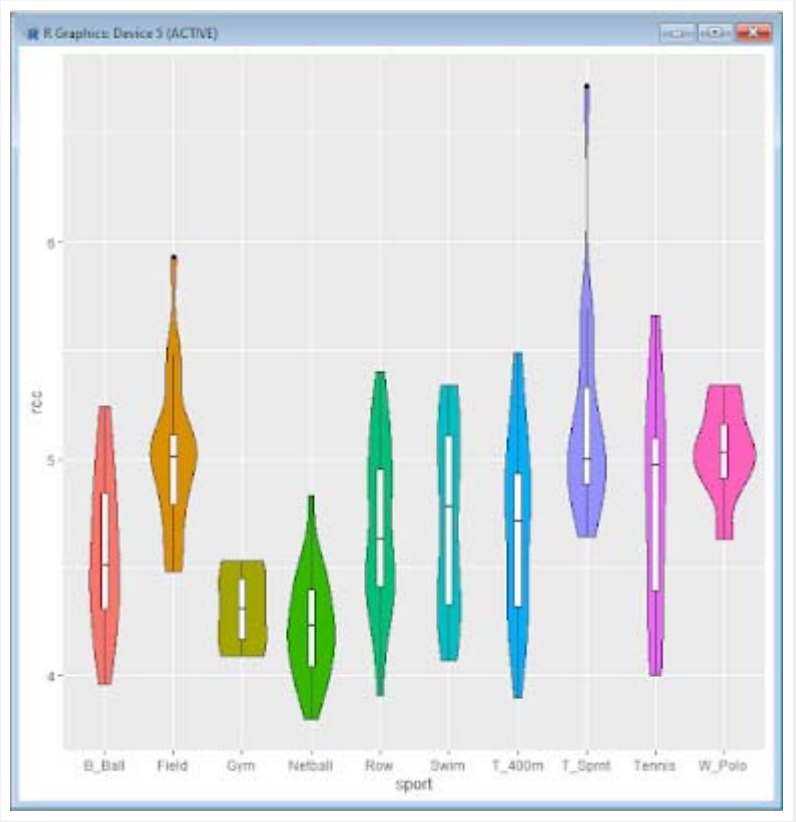


Copiate e incollate nella `Console di R` il quarto script e premete `↵ Invio`.

```
# GRAFICI A VIOLINO (VIOLIN PLOT)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
windows() # apre una nuova finestra
#
# violin plot senza legenda
ggplot(ais, aes(x=sport, y=rcc, fill=sport)) + geom_violin(trim=TRUE) +
geom_boxplot(width=0.1, fill="white") + theme(legend.position="none")
#
```

Questa volta aggiungendo **theme(legend.position="none")** abbiamo tolto, in quanto ridondante, la legenda generata automaticamente che compariva sulla destra.

Questo è conseguentemente il quarto grafico:

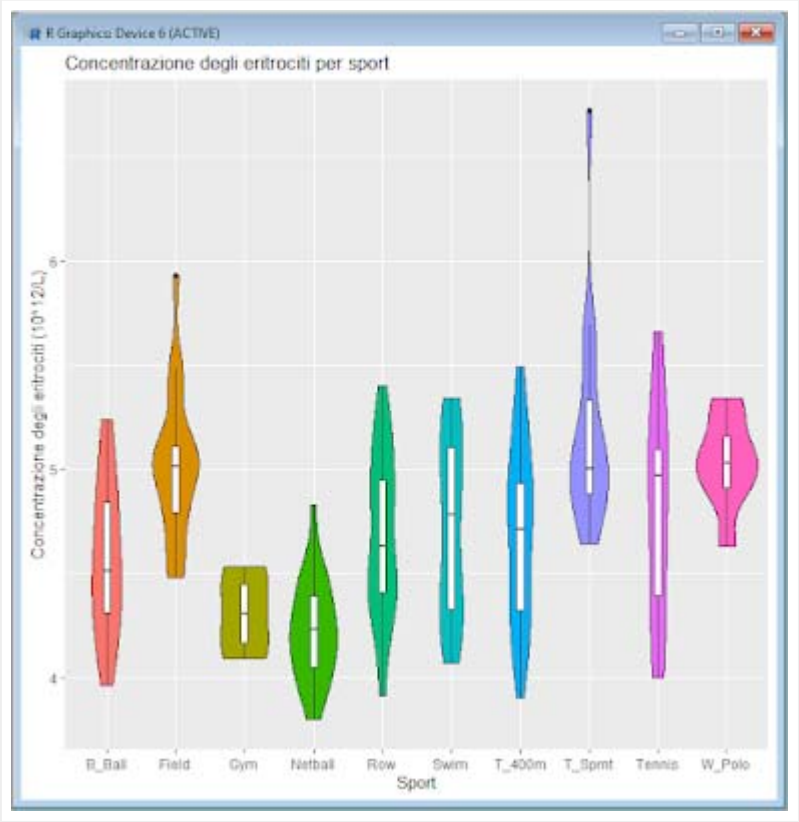


Copiate e incollate nella `Console di R` questo quinto script e premete ↵ `Invio`.

```
# GRAFICI A VIOLINO (VIOLIN PLOT)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
windows() # apre una nuova finestra
#
# violin plot con titolo ed etichette degli assi
ggplot(ais, aes(x=sport, y=rcc, fill=sport)) + geom_violin(trim=TRUE) +
geom_boxplot(width=0.1, fill="white") + theme(legend.position="none") +
labs(title="Concentrazione degli eritrociti per sport praticato", x="Sport", y =
"Concentrazione degli eritrociti (10^12/L)")
#
```

Ogni grafico che si rispetti prevede un titolo e la descrizione della variabili rappresentate, cosa che viene qui realizzata con la funzione **labs()** nella quale sono specificati titolo (**title=...**), etichetta dell'asse delle x (**x=...**) ed etichetta dell'asse delle y (**y=...**).

Questo è il relativo e quinto grafico:



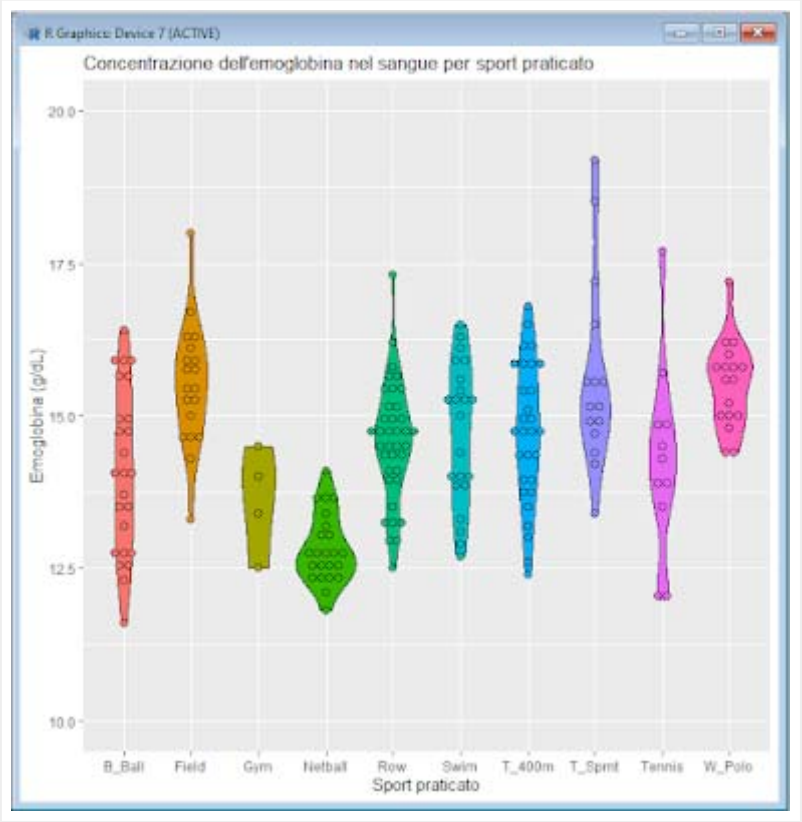
Per l'ultimo script copiate e incollate nella `Console di R` il codice che segue e premete ↵ `Invio`.

```
# GRAFICI A VIOLINO (VIOLIN PLOT)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(ggplot2) # carica il pacchetto necessario per la grafica
windows() # apre una nuova finestra
#
# violin plot con sovrapposto il grafico a punti
ggplot(ais, aes(x=sport, y=hg, fill=sport)) + geom_violin(trim=TRUE) +
geom_dotplot(method="dotdensity", binaxis='y', stackdir="center", stackratio=1,
dotsize=0.7, binwidth=0.2, show.legend=FALSE) + coord_cartesian(ylim=c(10,20)) +
theme(legend.position="none") + labs(title="Concentrazione dell'emoglobina nel sangue
per sport praticato", x="Sport praticato", y="Emoglobina (g/dL)")
#
```

Qui è stata aggiunta la funzione **geom_dotplot()** che sovrappone al grafico a violino/violin plot un *grafico a punti (dotplot)*. Inoltre i dati sono cambiati: abbiamo riportato, per ciascuno sport praticato, la concentrazione dell'emoglobina, la stessa impiegata per realizzare i grafici a punti [3], per consentirvi di effettuare un immediato confronto tra le due rappresentazioni grafiche.

Infine è stata aggiunta la funzione **coord_cartesian()** mediante la quale sono fissati con l'argomento **ylim=...** il limite inferiore (**10**) e il limite superiore (**20**) entro i quali rappresentare i valori.

E questo è quindi il sesto e ultimo grafico:



Potete facilmente adattare il codice per ottenere la rappresentazione che più vi interessa, aiutandovi con la documentazione del pacchetto [5]:
→ sostituendo la prima riga con il codice necessario per importare i vostri dati;
→ nella funzione **ggplot()** sostituendo **ais** con il nome dell'oggetto che contiene i vostri dati;
→ riportando in **x=...** il nome del *fattore* cioè della variabile qualitativa con la quale aggregare i dati in sottoinsiemi e in **y=...** il nome della vostra variabile numerica;
→ adattando opportunamente nella funzione **labs()** il titolo, il nome del fattore e il nome della variabile.

Per una guida rapida all'importazione in **R** dei vostri dati potete consultare i link:
→ importazione dei dati da un file .xls o .xlsx
→ importazione dei dati da un file .csv

Infine si rammenta che nel post Salvare i grafici di R in un file è riportato uno script che consente di trasformare i grafici in immagini e salvarli sotto forma di file `.bmp`, `.jpeg`, `.png`, `.pdf`, `.ps` per poterli stampare, archiviare, inserire in una pubblicazione, in un post o in un sito web.

---

[1] Per i dettagli vedere il post Kernel density plot.

[2] Vedere il post Grafici a scatola con i baffi (boxplot).

[3] Vedere il post Grafici a punti (dotplot).

[4] Potete scaricare questo e gli altri file di dati seguendo le indicazioni che trovate alla pagina Dati.

[5] Il manuale di riferimento *Package 'ggplot2'* lo potete scaricare facendo click sul link accanto alla voce *Reference manual* nella pagina di documentazione del pacchetto. URL consultato il 7/02/2021: https://bit.ly/3ioEJfE

sabato 22 maggio 2021

Analisi della varianza a un fattore

Come è stato sottolineato, a dispetto della "*... apparente incoerenza della nomenclatura - il fatto cioè che le tecniche riguardino confronti di medie piuttosto che varianze - ... il complesso delle tecniche chiamato analisi della varianza costituisce un potente metodo per analizzare il modo in cui il valore medio di una variabile è influenzato da classificazioni di vario genere dei dati*" [1].

Ancor più direttamente Wonnacott all'inizio del capitolo dedicato all'analisi della varianza riporta: "*... abbiamo compiuto inferenze relative alla media di una popolazione ... abbiamo esteso il ragionamento alla differenza fra due medie ... vogliamo confrontare ora* r *medie, usando un insieme di tecniche che vengono comunemente denominate «analisi della varianza»*" [2].

Le tecniche di **analisi della varianza** (**ANOVA**) sono ampiamente trattate in tutti i testi di statistica, ai quali si rimanda per i necessari approfondimenti. Vediamo in questo post la più semplice di queste tecniche, l'**analisi della varianza a un fattore** che:
→ è "*... una generalizzazione del test t [di Student] per dati non appaiati [per campioni indipendenti], adatta a un numero qualunque di gruppi ...*" [1];
→ è *"... equivalente al test t per dati non appaiati quando i gruppi sono ... due*" [1];
→ è basata "*... su due importanti ipotesi: (a) la normalità delle distribuzioni delle osservazioni ... e (b) la costanza delle varianze nei diversi gruppi*" [3].

I problemi che si presentano nella pratica quando gli assunti alla base del modello statistico dell'ANOVA non sono completamente soddisfatti - e i correttivi che possono eventualmente essere adottati - vanno al di la dei limiti di questo blog, ma sono ben approfonditi per esempio da Snedecor [4].

Per lo script impieghiamo i dati riportati da Wonnacott che riguardano "*... tre macchine (A, B e C), le quali, essendo azionate da uomini e a causa di altre ragioni inesplicabilil, danno luogo ad un prodotto orario soggetto a fluttuazioni casuali. Nella speranza di «mediare» e quindi di ridurre gli effetti di tali fluttuazioni, si effettua un campione casuale di 5 ore per ciascuna macchina, i cui risultati sono raccolti nella Tabella 10-1 insieme con le relative medie*" [5].

Tabella 10-1 *Campione dei prodotti da 3 macchine.*

Macchine o n. del campione	*Campione della macchina i*					$\bar{X}_i$
$i = 1$	48.4	49.7	48.7	48.5	47.7	48.6
$= 2$	56.1	56.3	56.9	57.6	55.1	56.4
$= 3$	52.1	51.1	51.6	52.1	51.1	51.6
						Media delle $\bar{X} = \bar{\bar{X}} = 52.2$

La domanda è: **le differenze tra le medie** di produzione, riportate nell'ultima colonna sulla destra, possono essere attribuite al caso? Se così non fosse, dovremmo pensare che esiste qualcosa in grado di influenzare la produzione media delle macchine.

I dati riportati in forma di tabella da Wonnacott per essere analizzati con **R** devono essere organizzati in un file di testo, sotto forma di righe (`record`) contenenti ciascuna due variabili (`campi`), una variabile qualitativa (`fattore`) che indica la *macchina*, e una variabile numerica che indica la *produzione* della macchina, e assumono quindi la forma seguente:

```
macchina;produzione
i1;48.4
i1;49.7
i1;48.7
i1;48.5
i1;47.7
i2;56.1
i2;56.3
i2;56.9
i2;57.6
i2;55.1
i3;52.1
i3;51.1
i3;51.6
i3;52.1
i3;51.1
```

Copiate le sedici righe riportate qui sopra aggiungendo un ↲ `Invio` al termine dell'ultima riga e salvatele in `C:\Rdati\` in un file di testo denominato `anova1.csv` (attenzione all'estensione `.csv` al momento del salvataggio del file).

In alternativa andate alla pagina Dati nella quale trovate diverse opzioni per scaricare i file di dati, quindi copiate il file `anova1.csv` nella cartella `C:\Rdati\`

Poi scaricate dal **CRAN** e installate il pacchetto **psych**, che ci consentirà di calcolare alcune statistiche riepilogative dei dati, e il pacchetto **ggplot2**, con il quale realizzeremo una rappresentazione grafica dei dati integrativa all'analisi statistica.
Infine copiate e incollate nella `Console di R` questo script e premete ↲ `Invio`:

```
# ANOVA analisi della varianza a un fattore
#
mydata <- read.table("c:/Rdati/anova1.csv", header=TRUE, sep=";", dec=".") # importa i dati
anova1 <- aov(produzione~macchina, data=mydata) # esegue l'analisi della varianza
summary(anova1) # mostra i risultati dell'analisi della varianza
#
library(psych) # carica il pacchetto per le statistiche riepilogative
describeBy(mydata$produzione, mydata$macchina) # statistiche della produzione per macchina
#
pairwise.t.test(mydata$produzione, mydata$macchina, p.adjust.method = "bonferroni") # confronto tra le (produzioni) medie delle macchine impiegando la correzione di Bonferroni per confronti multipli
#
library(ggplot2) # carica il pacchetto per la grafica
ggplot(mydata, aes(x=macchina, y=produzione, fill=macchina)) +
geom_dotplot(binaxis='y', stackdir='center', stackratio=1, dotsize=1, binwidth=0.4,
show.legend=FALSE) + coord_cartesian(ylim=c(40, 60)) + labs(title="Variabilità
osservata nella produzione tra le macchine", x="Macchina impiegata", y="Produzione
realizzata") + theme_classic() # dotplot della produzione per macchina
#
```

Lo script è suddiviso in quattro blocchi di codice. Nel primo, dopo avere importato i dati con la funzione **read.table()** (prima riga), viene eseguita l'analisi della varianza a un fattore mediante la funzione **aov()** (seconda riga) e sono riepilogati (terza riga) i risultati con la funzione **summary()**:

```
> summary(anova1) # mostra i risultati dell'analisi della varianza
            Df Sum Sq Mean Sq F value   Pr(>F)
macchina     2 154.80   77.40   141.6 4.51e-09 ***
Residuals   12   6.56    0.55
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

La varianza (Mean Sq) è calcolata dividendo la somma degli scarti quadratici (Sum Sq) per i gradi di libertà (Df). Il valore F (141.6) del rapporto fra varianze (F value) viene calcolato dividendo la varianza dovuta alle macchine (77.40) per la varianza residua (0.55).

La probabilità (Pr(>F)) è la probabilità di osservare per caso il valore F (141.6) e consente di rispondere alla domanda: le differenze tra le medie di produzione possono essere attribuite al caso? Essendo tale probabilità 4.51e-09 o se preferite 0.0000000451, quindi molto inferiore al valore 0.05 assunto in genere come valore soglia, concludiamo che le differenze tra le medie di produzione non sono attribuibili al caso: *le medie sono significativamente diverse, quindi esiste qualcosa in grado di influenzare la produzione [media] delle macchine*.

Da notare una cosa molto importante: l'ANOVA è un **test globale** e ci dice che tra le medie [nel nostro caso tra le medie di produzione delle tre macchine] esiste una differenza significativa, ma non consente di individuare la/e media/e causa della significatività. In altre parole non ci dice se la significatività sia dovuta alla differenza tra la prima macchina e la seconda, tra la prima macchina e la terza, tra la seconda macchina e la terza, o a una qualche combinazione di queste tre possibilità. Se è questo che interessa, è possibile ricorrere a test alternativi (vedi sotto).

Il secondo blocco di codice prevede di caricare il pacchetto **psych** e impiegare la funzione **describeBy()** per riportare le statistiche elementari separatamente per le tre macchine. Si tratta di un passo cruciale: e ci mostra che in effetti i dati, come richiesto dalle ipotesi soggiacenti al metodo statistico impiegato, riportate all'inizio del post, sono distribuiti in modo gaussiano.

```
> describeBy(mydata$produzione, mydata$macchina) # statistiche della produzione per
macchina

 Descriptive statistics by group
group: i1
   vars n mean   sd median trimmed mad  min  max range skew kurtosis   se
X1    1 5 48.6 0.72   48.5    48.6 0.3 47.7 49.7     2 0.32    -1.43 0.32
------------------------------------------------------------
group: i2
   vars n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
X1    1 5 56.4 0.93   56.3    56.4 0.89 55.1 57.6   2.5 -0.09    -1.68 0.42
------------------------------------------------------------
group: i3
   vars n mean  sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 5 51.6 0.5   51.6    51.6 0.74 51.1 52.1     1    0     -2.2 0.22
```

Il terzo blocco di codice consiste in una sola riga. Abbiamo già visto che l'analisi della varianza a un fattore è una generalizzazione del test t di Student per dati non appaiati, ed è equivalente al test t quando i gruppi sono due [6]. Tuttavia esistono delle correzioni del test t che consentono di impiegarlo anche nel caso del confronto di più di due gruppi [7], ed è proprio una di queste, la correzione di Bonferroni (**p.adjust.method = "bonferroni"**), che viene qui applicata ai dati mediante la funzione **pairwise.t.test()** per confrontarne i risultati con quelli dell'ANOVA

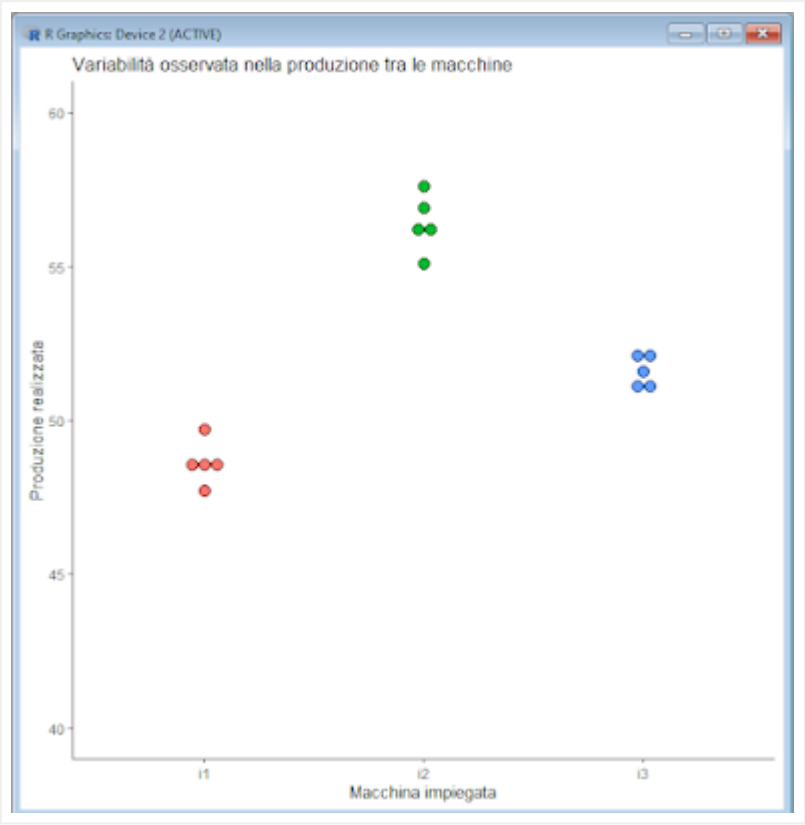
```
pairwise.t.test(mydata$produzione, mydata$macchina, p.adjust.method = "bonferroni") #
confronto tra le (produzioni) medie delle macchine impiegando la correzione di
Bonferroni per confronti multipli
```

```
        Pairwise comparisons using t tests with pooled SD

data:  mydata$produzione and mydata$macchina

   i1      i2
i2 3.4e-09 -
i3 1.0e-04 8.1e-07

P value adjustment method: bonferroni
```

Come si vede il test t effettuato per tutti e tre i confronti possibili tra macchine conferma i risultati dell'ANOVA riportando differenze sempre significative – essendo sempre inferiori a `0.05` le probabilità che le differenze tra medie qui osservate siano dovute al caso - ma ci indica anche dove risiedono tali differenze, riportando separatamente che `i1` <> `i2` (p=`3.4e-09`), `i1` <> `i3` (p=`1.0e-04`), `i2` <> `i3` (p=`8.1e-07`), cosa che l'ANOVA, essendo un test globale, non consente di specificare (vedi sopra).

Altre correzioni alternative, meno conservative della correzione di Bonferroni [9], possono essere impiegate con l'argomento **p.adjust.method =** : secondo Holm (1979) (**"holm"**), Hochberg (1988) (**"hochberg"**), Hommel (1988) (**"hommel"**), Benjamini & Hochberg (1995) (**"BH"** o **"fdr"**), Benjamini & Yekutieli (2001) (**"BY"**), e **"none"** per nessuna correzione.

L'ultimo blocco di codice prevede di caricare la libreria **ggplot2** per realizzare un grafico a punti (dotplot) [8] che consente di visualizzare i dati: il che è sempre un importante complemento all'analisi numerica.

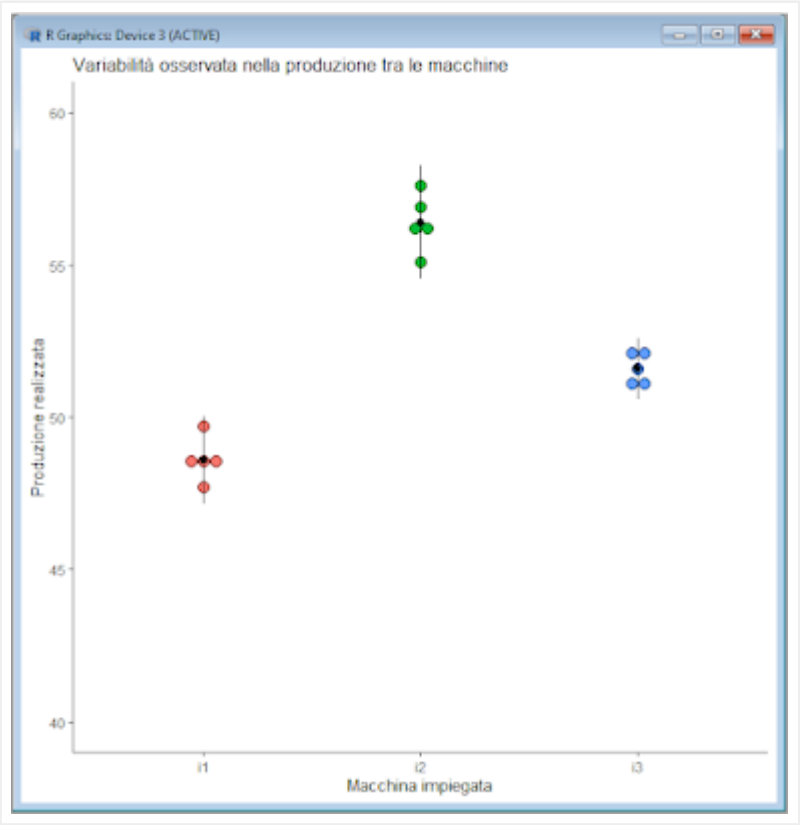


Se l'ANOVA consente di effettuare un confronto tra medie, a questo punto sembra logico sovrapporre ai dati raccolti la loro media e la loro deviazione standard.

Copiate e incollate nella `Console di R` queste due righe di codice che aprono una nuova finestra grafica nella quale viene rappresentato il grafico che sovrappone ai punti la media con l'intervallo

corrispondente a due deviazioni standard:

#
windows() # apre e inizializza una nuova finestra grafica
ggplot(mydata, aes(x=macchina, y=produzione, fill=macchina)) + geom_dotplot(binaxis='y', stackdir='center', stackratio=1, dotsize=1, binwidth=0.4, show.legend=FALSE) + coord_cartesian(ylim=c(40, 60)) + labs(title="Variabilità osservata nella produzione tra le macchine", x="Macchina impiegata", y="Produzione realizzata") + theme_classic() + stat_summary(fun.data=mean_sdl, fun.args = list(mult=2), geom="pointrange", color="black", show.legend=FALSE) # aggiunge media +/- 2 deviazioni standard
#



Da notare che nella funzione **stat_summary** l'argomento
mult=2
specifica di rappresentare attorno alla media l'intervallo uguale a 2 deviazioni standard, ma potete anche mettere
mult=1
per mostrare l'intervallo uguale a 1 deviazione standard, o qualsiasi altro valore per l'intervallo che desiderate rappresentare.

Come si vede i dati sono poco dispersi, le medie sono ben differenti, gli intervalli media ± 2 deviazioni standard non si sovrappongono: e questo conferma la significatività delle differenze tra le medie.

Dal punto di vista statistico è interessante notare l'importanza del **disegno sperimentale**. Quello qui adottato prevedeva di raccogliere i dati in modo tale che la variabilità osservata è quella dovuta al fattore *macchina+operatore*, pertanto non è possibile stabilire se le differenze (tra medie) osservate sono imputabili alla sola macchina, al solo operatore o a entrambi.

Impiegando un altro disegno sperimentale, e raccogliendo i dati diversamente, è possibile decomporre la variabilità osservata in variabilità dovuta al fattore *macchina* e variabilità dovuta al fattore *operatore*. lo facciamo nel post Analisi della varianza a due fattori, al termine del quale trovate anche alcune considerazioni generali sull'ANOVA.

---

[1] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, Milano, 1979, p. 188.

[2] Wonnacott TH, Wonnacott RJ. *Introduzione alla statistica*. Franco Angeli Editore, Milano, 1980, ISBN 88-204-0323-4, p. 237.

[3] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, Milano, 1979, pp. 195-196.

[4] Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6. *Chapter 15 - Failures in the assumptions*, pp. 274-297.

[5] Wonnacott TH, Wonnacott RJ. *Introduzione alla statistica*. Franco Angeli Editore, Milano, 1980, ISBN 88-204-0323-4, Tabella 10-1, p. 238.

[6] Vedere il post Test parametrici e non parametrici per due campioni indipendenti.

[7] Vedere il post Test parametrici e non parametrici per più campioni indipendenti per il significato di queste correzioni, che controbilanciano l'aumento della probabilità, derivante dall'esecuzione di confronti multipli, di considerare la differenza tra le medie di due campioni come significativa quando invece non è significativa.

[8] Si rimanda al post Grafici a punti (dotplot), inoltre nel post Grafici a violino (violin plot) trovate anche come sovrapporre ai punti un grafico a violino o un grafico a scatola con i baffi (boxplot).

[9] Un test statistico più conservativo a parità di condizioni produce come risultato un valore di p più alto quindi ci aspettiamo che a lungo andare fornisca un numero inferiore di differenze significative, mentre un test statistico meno conservativo a parità di condizioni produce come risultato un valore di p più basso quindi ci aspettiamo che a lungo andare fornisca un numero maggiore di differenze significative.

giovedì 3 giugno 2021

Calcolare la media mobile e la mediana mobile

Data una sequenza di valori numerici la **media mobile** è la media calcolata in corrispondenza di ciascun dato su un numero fisso di dati, determinato da una finestra di ampiezza prestabilita che scorre i dati e nella quale il dato più vecchio viene ogni volta sostituito con un nuovo dato.

Così ad esempio per una finestra con l'ampiezza di 5 dati, in corrispondenza dei primi 4 dati la media mobile non può essere calcolata, in corrispondenza del dato numero 5 viene calcolata la media dei dati da 1 a 5, in corrispondenza del dato numero 6 viene calcolata la media dei dati da 2 a 6, in corrispondenza del dato numero 7 viene calcolata la media dei dati da 3 a 7 e così via. La **mediana mobile** a sua volta è la mediana calcolata in corrispondenza di ciascun dato nello stesso modo.

Per calcolare la media mobile e la mediana mobile facciamo ricorso al pacchetto **zoo**, che deve essere scaricato dal **CRAN**.

Copiate e incollate questo script nella `Console di R` e premete `↲ Invio`:

```
# CALCOLARE LA MEDIA MOBILE E LA MEDIANA MOBILE
#
library(zoo) # carica il pacchetto
rileva_x <- seq(1:50) # numero progressivo della rilevazione in ascisse
valore_y <- c(5, 9, 12, 7, 8, 13, 26, 7, 8, 19, 10, 7, 5, 36, 12, 18, 31, 17, 16, 43, 7, 12, 9, 27, 15, 18, 24, 29, 32, 23, 29, 12, 7, 48, 34, 45, 19, 12, 16, 18, 22, 31, 38, 7, 11, 4, 38, 36 ,43, 15) # valore osservato corrispondente in ordinate
mydata <- data.frame(rileva_x, valore_y) # combina i vettori in una matrice
#
mydata$mediamob <- rollmean(valore_y, k=7, fill=NA, align='right') # calcola la media mobile (dato corrente + 6 precedenti) e la aggiunge in una nuova colonna
mydata$medianamob <- rollmedian(valore_y, k=7, fill=NA, align='right') # calcola la mediana mobile (dato corrente + 6 precedenti) e la aggiunge in una nuova colonna
mydata # mostra la matrice con i dati definitivi
#
plot(mydata$rileva_x, mydata$valore_y, xlim=c(0,50), ylim=c(0,60), type="l", lty=1, lwd=1, col="black", main="Valori osservati, media mobile e mediana mobile", xlab="Rilevazioni", ylab="Valori osservati") # grafico dei dati originali
lines(mydata$rileva_x, mydata$mediamob, xlim=c(0,50), ylim=c(0,60), type="l", lty=2, lwd=1, col="red") # sovrappone il grafico della media mobile
lines(mydata$rileva_x, mydata$medianamob, xlim=c(0,50), ylim=c(0,60), type="l", lty=2, lwd=1, col="blue") # sovrappone il grafico della mediana mobile
#
legend(0,60, legend=c("Valori osservati nelle rilevazioni", "Media mobile dei valori osservati", "Mediana mobile dei valori osservati"), col=c("black", "red", "blue"), lty=c(1,2,2), lwd=c(1,1,1), cex=0.8) # aggiunge la legenda
#
```

Dopo avere caricato il pacchetto **zoo** i dati impiegati come esempio sono generati in questo modo:
→ nel vettore **rileva_x** viene riportato il numero progressivo della rilevazione;
→ nel vettore **valore_y** viene riportato il valore numerico osservato in corrispondenza di ciascuna rilevazione, valore numerico sul quale calcoleremo la media mobile e la mediana mobile;
→ con la funzione **dataframe()** i due vettori sono combinati nella matrice **mydata** [1].

A questo punto alla matrice **mydata** sono aggiunte due colonne:
→ la colonna/vettore **mydata$mediamob** nella quale è riportata la media mobile calcolata in corrispondenza di ogni dato;

→ la colonna/vettore **mydata$medianamob** nella quale è riportata la mediana mobile calcolata in corrispondenza di ogni dato.

La media mobile e la mediana mobile sono calcolate rispettivamente con le funzioni **rollmean()** e **rollmedian()** che impiegano i seguenti argomenti:
→ la variabile (**valore_y**) sulla quale calcolare la media mobile;
→ l'ampiezza della finestra (**k=7**) che specifica che la media (o mediana) mobile va calcolata, in corrispondenza di ogni dato, su quel dato e sui sei dati che lo precedono;
→ **fill=NA** che specifica di riportare **NA** in corrispondenza dei valori non computabili;
→ **align='right'** che - in quanto con **k=7** la media e la mediana possono essere calcolate solamente a partire dal settimo dato - specifica di riportare i valori non computabili **NA** nelle prime sei posizioni.

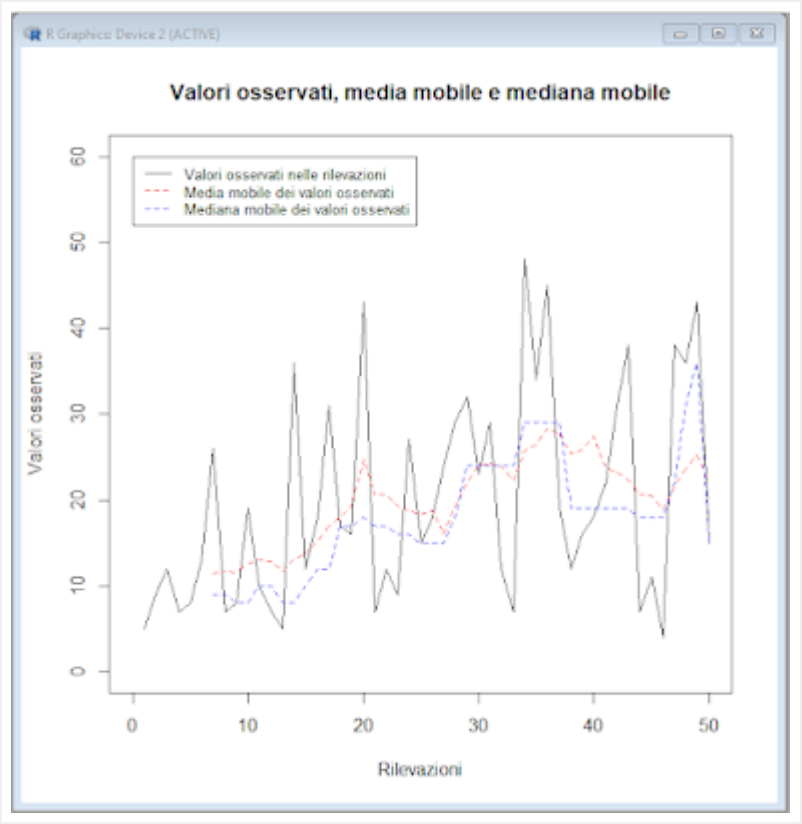
Per i dettagli delle funzioni impiegate digitare **help(*nomedellafunzione*)** nella `Console di R`. Al bisogno potete consultare il manuale del pacchetto **zoo** [2].

Non resta quindi che mostrare il contenuto della matrice **mydata** nella versione definitiva contenente il numero della rilevazione (`rileva_x`), il valore osservato (`valore_y`) e le rispettive media mobilea (`mediamob`) e mediana mobile (`medianamob`)

```
> mydata # mostra la matrice con i dati definitivi
   rileva_x valore_y mediamob medianamob
1         1        5       NA         NA
2         2        9       NA         NA
3         3       12       NA         NA
4         4        7       NA         NA
5         5        8       NA         NA
6         6       13       NA         NA
7         7       26 11.42857          9
8         8        7 11.71429          9
9         9        8 11.57143          8
10       10       19 12.57143          8
11       11       10 13.00000         10
12       12        7 12.85714         10
13       13        5 11.71429          8
14       14       36 13.14286          8
15       15       12 13.85714         10
16       16       18 15.28571         12
17       17       31 17.00000         12
18       18       17 18.00000         17
19       19       16 19.28571         17
20       20       43 24.71429         18
21       21        7 20.57143         17
22       22       12 20.57143         17
23       23        9 19.28571         16
24       24       27 18.71429         16
25       25       15 18.42857         15
26       26       18 18.71429         15
27       27       24 16.00000         15
28       28       29 19.14286         18
29       29       32 22.00000         24
30       30       23 24.00000         24
31       31       29 24.28571         24
32       32       12 23.85714         24
33       33        7 22.28571         24
34       34       48 25.71429         29
35       35       34 26.42857         29
```

```
36        36        45 28.28571          29
37        37        19 27.71429          29
38        38        12 25.28571          19
39        39        16 25.85714          19
40        40        18 27.42857          19
41        41        22 23.71429          19
42        42        31 23.28571          19
43        43        38 22.28571          19
44        44         7 20.57143          18
45        45        11 20.42857          18
46        46         4 18.71429          18
47        47        38 21.57143          22
48        48        36 23.57143          31
49        49        43 25.28571          36
50        50        15 22.00000          15
```

che sono impiegati per la successiva rappresentazione grafica



che viene realizzata con tre sole funzioni:
→ la funzione **plot()** che traccia il grafico del valore osservato **mydata$valore_y** in corrispondenza di ciascuna rilevazione **mydata$rileva_x** impiegando una linea (**type="l"**) continua (**lty=1**) di larghezza unitaria (**lwd=1**) e di colore nero (**col="black"**);
→ la funzione **lines()** che viene impiegata una prima volta per sovrapporre il grafico della media mobile con una linea tratteggiata (**lty=2)** di colore rosso (**col="red"**) e una seconda volta per sovrapporre il grafico della mediana mobile con una linea tratteggiata (**lty=2**) di colore blu (**col="blue"**)
→ la funzione **legend()** che riporta la legenda esplicativa dei grafici [3].

Ovviamente lo script può essere riadattato per rappresentare dati differenti ma anche per riportare solamente l'uno o l'altro dei grafici.

---

[1] Parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14
6	7	18	12
11	8	17	13
7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

[2] Vedere la voce *Reference manual* del pacchetto zoo: S3 Infrastructure for Regular and Irregular Time Series (Z's Ordered Observations). URL consultato il 02/01/2023.

[3] Per un approfondimento sull'impiego della funzione **legend()** rimando al post Aggiungere la legenda a un grafico.

lunedì 12 luglio 2021

Analisi della varianza a due fattori

Il complesso di tecniche statistiche note come **analisi della varianza** (**ANOVA**) è stato introdotto con l'analisi della varianza a un fattore con la quale avevamo analizzato i dati di produzione di tre macchine, azionate da operatori non meglio specificati, con la domanda: le differenze tra le medie di produzione delle tre macchine possono essere attribuite al caso? Se così non fosse, dovremmo pensare che esiste qualcosa in grado di influenzare la produzione media delle macchine.

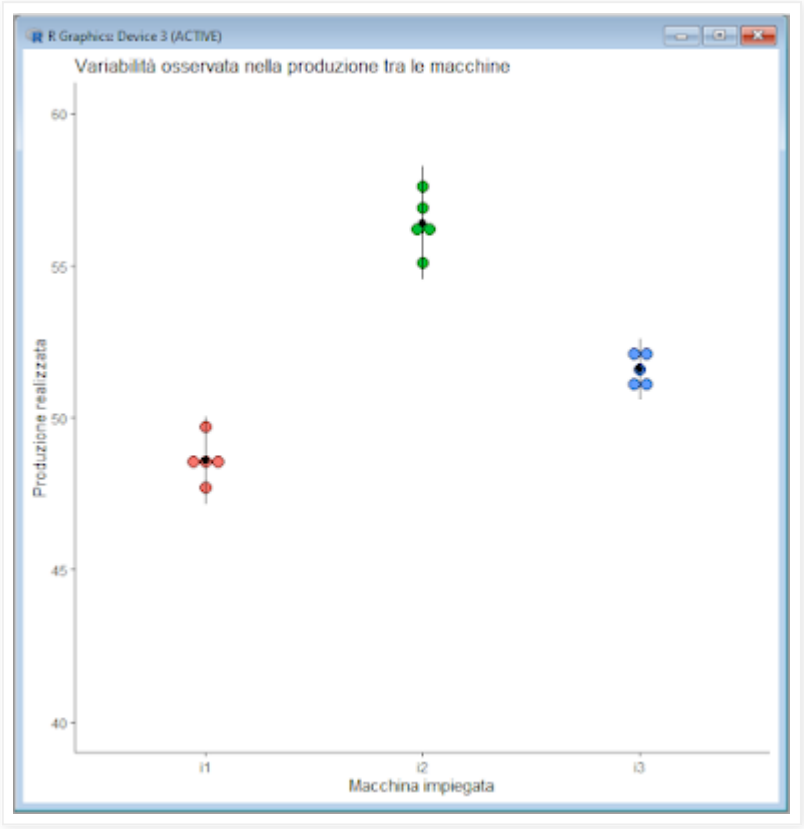
Il disegno sperimentale adottato aveva previsto di raccogliere i dati in modo tale che la variabilità osservata, misurata dalla differenza tra le medie di produzione delle `i` macchine, risultava essere quella determinata dal fattore *macchina+operatore*. Questi erano i dati riportati da Wonnacott [1] che abbiamo impiegato

Tabella 10-1 *Campione dei prodotti da 3 macchine.*

Macchine o n. del campione	*Campione della macchina i*					$\bar{X}_i$
$i = 1$	48.4	49.7	48.7	48.5	47.7	48.6
$= 2$	56.1	56.3	56.9	57.6	55.1	56.4
$= 3$	52.1	51.1	51.6	52.1	51.1	51.6

Media delle $\bar{X} = \overline{\bar{X}} = 52.2$

e questo è il grafico che li rappresentava (in nero la *media ± 2 deviazioni standard*).



Ci proponiamo ora di fare un passo ulteriore: decomporre la variabilità osservata in variabilità dovuta al fattore *macchina* e variabilità dovuta al fattore *operatore*. Per questo impieghiamo di nuovo dati illustrati da Wonnacott [2], ma raccolti diversamente, che riportano nelle righe la produzione delle macchine e nelle colonne gli operatori che le azionano. Da notare che le medie [della produzione] delle macchine sono identiche alle precedenti, ma i dati hanno una distribuzione di valori più ampia, un fatto molto importante e che si ripercuote sulle conclusioni che dai dati possiamo trarre.

Tabella 10-9 *Campioni della produzione* (X_{ij}) *di tre diverse macchine* (come nella Tabella 10-4, ma ordinate secondo l'operatore).

Operatore → / *Macchine* ↓	$j = 1$	2	3	4	5	*Media della macchina* $\bar{X}_{i.}$
$i = 1$	56.7	45.7	48.3	54.6	37.7	48.6
2	64.5	53.4	54.3	57.5	52.3	56.4
3	56.7	50.6	49.5	56.5	44.7	51.6
Media dell'operatore $\bar{X}_{.j}$	59.3	49.9	50.7	56.2	44.9	$\bar{\bar{X}} = 52.2$

Il **disegno sperimentale** adottato ci consente di confrontare tra loro sia le medie delle `i` macchine (ultima colonna), sia le medie dei `j` diversi operatori (ultima riga) e analizzare le **differenze tra le medie** rispondendo separatamente a due domande:
→ le differenze tra le *medie delle macchine* possono essere attribuite al caso?
→ le differenze tra le *medie degli operatori* possono essere attribuite al caso?

Se così non fosse (cioè se le differenze non possono essere attribuite al caso) dovremmo pensare che esiste qualcosa in grado di influenzare la produzione media delle macchine (ad esempio differenze strutturali, inadeguata manutenzione o altro) e/o qualcosa in grado di influenzare la produzione media degli operatori (ad esempio formazione dell'operatore, fenomeni di fatica o altro).

I dati riportati in forma di tabella da Wonnacott per essere analizzati con **R** devono essere organizzati in un file di testo, sotto forma di righe (`record`) contenenti ciascuna tre variabili (`campi`), una prima variabile qualitativa (`fattore`) che indica la `macchina`, una seconda variabile qualitativa (`fattore`) che indica l'`operatore` e una variabile numerica che indica la `produzione` della combinazione macchina/operatore, e assumono quindi la forma seguente:

```
macchina;operatore;produzione
i1;j1;56.7
i1;j2;45.7
i1;j3;48.3
i1;j4;54.6
i1;j5;37.7
i2;j1;64.5
i2;j2;53.4
i2;j3;54.3
i2;j4;57.5
i2;j5;52.3
i3;j1;56.7
i3;j2;50.6
i3;j3;49.5
```

i3;j4;56.5
i3;j5;44.7

Copiate le sedici righe riportate qui sopra aggiungendo un ↲ Invio al termine dell'ultima riga e salvatele in C:\Rdati\ in un file di testo denominato anova2.csv (attenzione all'estensione .csv al momento del salvataggio del file).

In alternativa andate alla pagina Dati nella quale trovate diverse opzioni per scaricare i file di dati, quindi copiate il file anova2.csv nella cartella C:\Rdati\

Poi, se ancora non l'avete fatto, scaricate dal **CRAN** e installate il pacchetto **psych**, che ci consentirà di calcolare alcune statistiche riepilogative dei dati, e il pacchetto **ggplot2**, con il quale realizzeremo una rappresentazione grafica dei dati integrativa all'analisi statistica.

Infine copiate e incollate nella Console di R questo script e premete ↲ Invio:

```
# ANOVA analisi della varianza a due fattori
#
mydata <- read.table("c:/Rdati/anova2.csv", header=TRUE, sep=";", dec=".") # importa i dati [1]
anova2 <- aov(produzione~macchina+operatore, data=mydata) # calcola la variabilità tra macchine e tra operatori
summary(anova2) # mostra i risultati
#
library(psych) # carica il pacchetto
describeBy(mydata$produzione, mydata$macchina) # statistiche della produzione per macchina
describeBy(mydata$produzione, mydata$operatore) # statistiche della produzione per operatore
#
pairwise.t.test(mydata$produzione, mydata$macchina, p.adjust.method = "bonferroni") # confronto tra le (produzioni) medie delle macchine impiegando la correzione di bonferroni per confronti multipli
pairwise.t.test(mydata$produzione, mydata$operatore, p.adjust.method = "bonferroni") # confronto tra le (produzioni) medie degli operatori impiegando la correzione di bonferroni per confronti multipli [2]
#
library(ggplot2) # carica il pacchetto per la grafica
#
plot1 <- ggplot(mydata, aes(x=macchina, y=produzione, fill=macchina)) + geom_dotplot(binaxis='y', stackdir='center', stackratio=1, dotsize=1, binwidth=0.4, show.legend=FALSE) + coord_cartesian(ylim=c(40, 70)) + labs(title="Variabilità tra le macchine", x="Macchina impiegata", y="Produzione realizzata") + theme_classic() # dotplot della produzione per macchina
#
plot2 <- ggplot(mydata, aes(x=operatore, y=produzione, fill=operatore)) + geom_dotplot(binaxis='y', stackdir='center', stackratio=1, dotsize=1, binwidth=0.4, show.legend=FALSE) + coord_cartesian(ylim=c(40, 70)) + labs(title="Variabilità tra gli operatori", x="Operatore", y="Produzione realizzata") + theme_classic() # dotplot della produzione per operatore
library(gridExtra) # carica il pacchetto
grid.arrange(plot1, plot2, nrow = 1) # i due dotplot sono mostrati affiancati orizzontalmente
#
```

Lo script è suddiviso in quattro blocchi di codice. Nel primo blocco, dopo avere importato i dati con la funzione **read.table()** (prima riga), viene eseguita l'analisi della varianza a un fattore mediante la

funzione **aov()** (seconda riga) e sono riepilogati (terza riga) i risultati con la funzione **summary()**:

```
            Df Sum Sq Mean Sq F value   Pr(>F)
macchina     2  154.8   77.40   13.10 0.002997 **
operatore    4  381.7   95.43   16.15 0.000673 ***
Residuals    8   47.3    5.91
---
Signif. Codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

La varianza (`Mean Sq`) è calcolata dividendo la somma degli scarti quadratici (`Sum Sq`) per i gradi di libertà (`Df`). Il valore F (`F value`) viene calcolato separatamente per la macchina e per l'operatore, dividendo la varianza dovuta alle macchine (`77.40`) per la varianza residua (`5.91`) ovvero dividendo la varianza dovuta agli operatori (`95.43`) per la varianza residua (`5.91`) .

La probabilità (`Pr(>F)`) è la probabilità di osservare per caso il valore F e consente di rispondere alla domanda: le differenze tra le medie di produzione possono essere attribuite al caso? Per le medie di produzione delle macchine abbiamo un valore `Pr(>F)= 0.002997` e per le medie di produzione degli operatori abbiamo un valore `Pr(>F)= 0.000673`. Poiché entrambi sono inferiori al valore `0.05` assunto in genere come valore soglia, concludiamo che le differenze osservate non sono attribuibili al caso o, se preferite, che sono significative: quindi *esiste qualcosa in grado di influenzare la produzione a livello delle macchine e/o a livello degli operatori*.

Il secondo blocco di codice prevede di caricare il pacchetto **psych** e impiegare la funzione **describeBy()** per riportare le statistiche elementari prima per le tre macchine `i` e poi per i cinque operatori `j`. Si tratta di un passo importante, che ci mostra che i dati, come richiesto dagli assunti alla base dell'ANOVA [3], sono distribuiti in modo gaussiano.

```
> describeBy(mydata$produzione, mydata$macchina) # statistiche della produzione per macchina

 Descriptive statistics by group
group: i1
   vars n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
X1    1 5 48.6 7.57   48.3    48.6 9.34 37.7 56.7    19 -0.26    -1.79 3.38
------------------------------------------------------------
group: i2
   vars n mean   sd median trimmed  mad  min  max range skew kurtosis  se
X1    1 5 56.4 4.93   54.3    56.4 2.97 52.3 64.5  12.2 0.72    -1.41 2.2
------------------------------------------------------------
group: i3
   vars n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
X1    1 5 51.6 5.08   50.6    51.6 8.75 44.7 56.7    12 -0.14    -1.93 2.27

> describeBy(mydata$produzione, mydata$operatore) # statistiche della produzione per operatore

 Descriptive statistics by group
group: j1
   vars n mean  sd median trimmed mad  min  max range skew kurtosis  se
X1    1 3 59.3 4.5   56.7    59.3   0 56.7 64.5   7.8 0.38    -2.33 2.6
------------------------------------------------------------
group: j2
   vars n mean  sd median trimmed  mad  min  max range  skew kurtosis   se
X1    1 3 49.9 3.9   50.6    49.9 4.15 45.7 53.4   7.7 -0.17    -2.33 2.25
------------------------------------------------------------
```

```
group: j3
   vars n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 3 50.7 3.17   49.5    50.7 1.78 48.3 54.3     6 0.32    -2.33 1.83
------------------------------------------------------------
group: j4
   vars n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 3 56.2 1.47   56.5    56.2 1.48 54.6 57.5   2.9 -0.2    -2.33 0.85
------------------------------------------------------------
group: j5
   vars n mean  sd median trimmed   mad  min  max range skew kurtosis   se
X1    1 3 44.9 7.3   44.7    44.9 10.38 37.7 52.3  14.6 0.03    -2.33 4.22
```

Il terzo blocco di codice consiste in due righe di codice, una per analizzare i dati delle macchine, e una analizzare i dati degli operatori. Abbiamo già visto che l'analisi della varianza a un fattore è una generalizzazione del test t di Student per dati non appaiati, ed è equivalente al test t quando i gruppi sono due [4]. Tuttavia esistono delle correzioni del test t che consentono di impiegarlo anche nel caso del confronto di più di due gruppi [5]. Queste correzioni controbilanciano l'aumento della probabilità, derivante dall'esecuzione di confronti multipli, di considerare la differenza tra le medie di due campioni come significativa quando invece non è significativa, ed è proprio una di queste, la correzione di Bonferroni (**p.adjust.method = "bonferroni"**), che viene qui applicata ai dati mediante la funzione **pairwise.t.test()**. Mentre l'ANOVA è un test globale, che ci dice che tra le medie esiste una qualche differenza, ma non ci consente di individuare la/e media/e responsabile/i di tale differenza, il test t viene effettuato per tutti i confronti possibili tra medie.

Il test t effettuato per tutti i confronti possibili tra `macchine` indica differenze sempre non significative - essendo sempre superiori a `0.05` le probabilità che le differenze tra medie qui osservate siano dovute al caso:

```
> pairwise.t.test(mydata$produzione, mydata$macchina, p.adjust.method = "bonferroni")
# confronto tra le (produzioni) medie delle macchine impiegando la correzione di
bonferroni per confronti multipli

        Pairwise comparisons using t tests with pooled SD

data:  mydata$produzione and mydata$macchina

   i1   i2
i2 0.18 -
i3 1.00 0.69

P value adjustment method: bonferroni
```

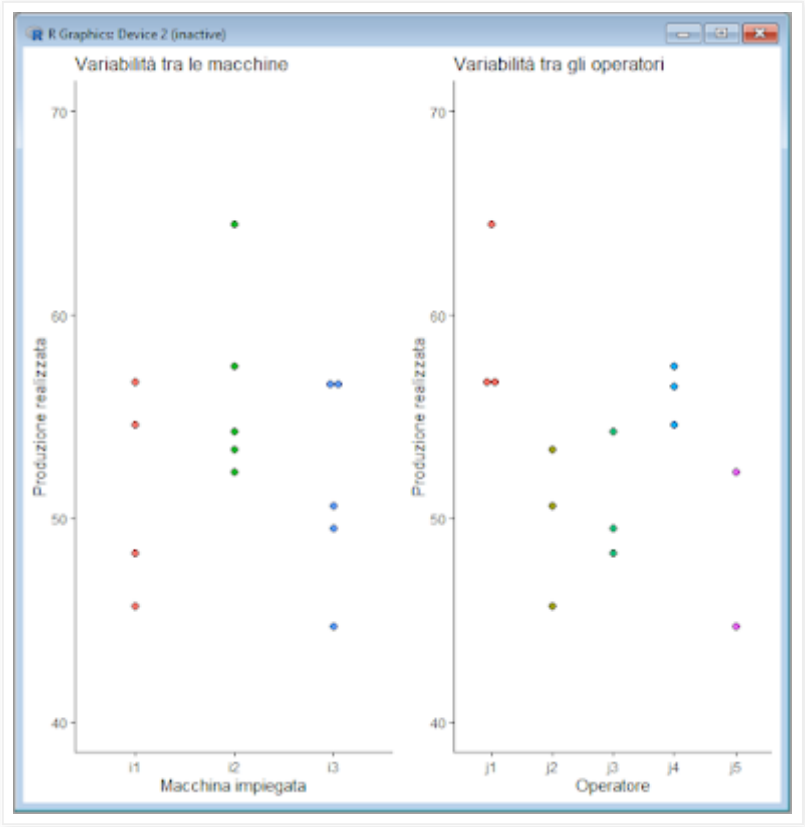
Il test t effettuato per tutti i confronti possibili tra `operatori` ci indica che la significatività dall'ANOVA è dovuta ad un'unica differenza tra tutti i confronti effettuati: `j1` risulta diverso da `j5` con un valore `p=0.029`, di poco inferiore al valore soglia `0.05` comunemente adottato:

```
>    pairwise.t.test(mydata$produzione,    mydata$operatore,    p.adjust.method    =
"bonferroni") # confronto tra le (produzioni) medie degli operatori impiegando la
correzione di bonferroni per confronti multipli

        Pairwise comparisons using t tests with pooled SD

data:  mydata$produzione and mydata$operatore

   j1    j2    j3    j4
j2 0.283 -     -     -
```

```
j3 0.411 1.000 -     -
j4 1.000 1.000 1.000 -
j5 0.029 1.000 1.000 0.117

P value adjustment method: bonferroni
```

Quindi il test t con la correzione di Bonferroni:
→ nel confronto tra le medie delle `macchine` indica differenze sempre non significative, contrariamente all'ANOVA;
→ nel confronto tra le medie degli `operatori` conferma il risultato dell'ANOVA specificando che la differenza è imputabile ad un unico caso, quello degli operatori `j1` e `j5`.

Il quarto e ultimo blocco di codice prevede di caricare la libreria **ggplot2** per realizzare un grafico a punti [6] che visualizza i dati della variabilità tra le macchine e un grafico a punti che visualizza i dati delle variabilità tra gli operatori, e combinarli in un'unica figura.



Se l'ANOVA consente di effettuare un confronto tra medie, a questo punto sembra logico sovrapporre ai dati raccolti la loro media e la loro deviazione standard.

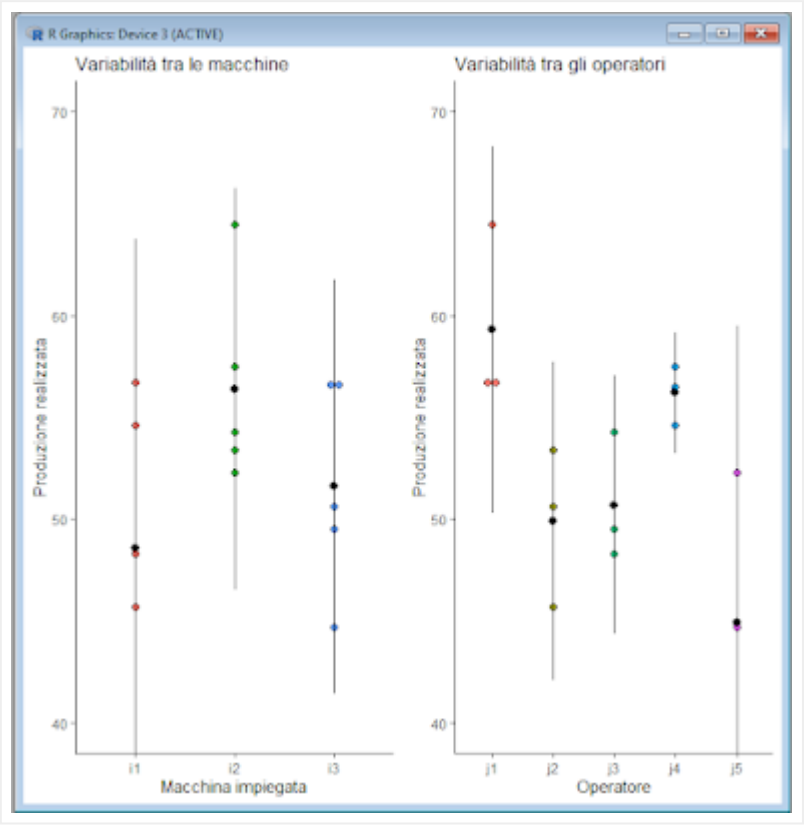
Copiate e incollate nella `Console di R` queste quattro righe di codice che aprono una nuova finestra grafica nella quale viene rappresentato il grafico che sovrappone ai punti la media con l'intervallo corrispondente a due deviazioni standard:

```
#
windows() # apre e inizializza una nuova finestra grafica
#
plot1 <- ggplot(mydata, aes(x=macchina, y=produzione, fill=macchina)) +
geom_dotplot(binaxis='y', stackdir='center', stackratio=1, dotsize=1,
binwidth=0.4, show.legend=FALSE) + coord_cartesian(ylim=c(40, 70)) +
labs(title="Variabilità tra le macchine", x="Macchina impiegata", y="Produzione
realizzata") + theme_classic() + stat_summary(fun.data = mean_sdl,
```

fun.args=list(mult=2), geom="pointrange", color="black", show.legend = FALSE) # dotplot della produzione per macchina
#
plot2 <- ggplot(mydata, aes(x=operatore, y=produzione, fill=operatore)) + geom_dotplot(binaxis='y', stackdir='center', stackratio=1, dotsize=1, binwidth=0.4, show.legend=FALSE) + coord_cartesian(ylim=c(40, 70)) + labs(title="Variabilità tra gli operatori", x="Operatore", y="Produzione realizzata") + theme_classic() + stat_summary(fun.data = mean_sdl, fun.args = list(mult=2), geom="pointrange", color="black", show.legend=FALSE) # dotplot della produzione per macchina
#
grid.arrange(plot1, plot2, nrow = 1) # i due dotplot sono mostrati affiancati orizzontalmente
#

Da notare che nella funzione **stat_summary()** l'argomento
mult=2
specifica di rappresentare attorno alla media l'intervallo uguale a 2 deviazioni standard, ma potete anche mettere
mult=1
per mostrare l'intervallo uguale a una deviazione standard, o qualsiasi altro valore per l'intervallo che desiderate rappresentare.

La rappresentazione grafica dei dati ci viene nuovamente in aiuto nell'interpretazione dei risultati. Come si vede in questo caso i dati sono molto dispersi, e gli intervalli *media ± 2 deviazioni standard* sono ampiamente sovrapposti: e questo corrobora la non significatività delle differenze tra le medie. Da notare la notevole differenza rispetto ai dati dell'ANOVA a un fattore riportati nella prima figura in alto. La maggior differenza rilevata graficamente è quella tra operatore `j1` e operatore `j5` ed è in linea con la debole (appena inferiore a 0.05) significatività statistica (`p=0.029`) di tale differenza.



La conclusione? Impiegando i dati tratti da un testo di statistica abbiamo sviluppato due esempi di analisi della varianza, a un fattore [7] e a due fattori, dai quali possiamo ricavare alcune considerazioni interessanti:
→ l'analisi della varianza (ANOVA) a dispetto del nome, è un insieme di tecniche per effettuare confronti multipli tra medie;
→ l'ANOVA fornisce come risultato un rapporto tra varianze (il test F), che è un test globale che ci dice che tra le medie esiste una qualche differenza, ma non specifica la/e media/e responsabile/i di tale differenza;
→ nel caso limite in cui il confronto tra medie è limitato a due campioni l'ANOVA equivale al test t di Student per dati non appaiati (campioni indipendenti);
→ i confronti multipli tra medie possono essere effettuati anche impiegando test alternativi come il test t con opportune correzioni, ad esempio con la correzione di Bonferroni, con il vantaggio, in questo caso, che la significatività viene valutata separatamente per ciascuna coppia di medie poste a confronto;
→ il valore soglia $p=0.05$ non è un dogma, e qualora si ritenga opportuno essere più prudenti (conservativi) nel giudicare la significatività di un test statistico si può adottare un valore soglia inferiore come per esempio $p=0.01$;
→ quando i dati sono poco dispersi e l'intervallo media ± 2 deviazioni standard non si sovrappone i risultati di ANOVA e test t (con le opportune correzioni) sono uguali in quanto l'informazione fornita dai dati è elevata, lascia poco adito a dubbi [sulla significatività delle differenze tra le medie] e la diversità degli assunti alla base delle tecniche statistiche impiegate non porta a conclusioni diverse;
→ quando i dati sono molto dispersi e l'intervallo media ± 2 deviazioni standard è ampiamente sovrapposto (vedere la seconda e la terza immagine riportate qui sopra), i risultati di ANOVA e test t (con le opportune correzioni) possono differire in quanto l'informazione fornita dai dati è scarsa, lascia adito a molti dubbi [sulla significatività delle differenze tra le medie] e la diversità degli assunti alla base delle tecniche statistiche impiegate può portare a conclusioni diverse;
→ la rappresentazione grafica dei dati rappresenta come sempre una importante integrazione ai test statistici.

In sintesi anche i risultati dell'ANOVA, come del resto tutti i risultati dell'analisi statistica, non devono essere interpretati in modo schematico e rigido, ma devono essere interpretati:
→ verificando che i dati analizzati rispettino gli assunti dell'ANOVA;
→ integrando i risultati dell'ANOVA con quelli di test alternativi per il confronto tra medie;
→ valutando criticamente le eventuali discrepanze tra i risultati dell'ANOVA e dei test alternativi;
→ adottando al bisogno requisiti di significatività più stringenti del tradizionale $p=0.05$;
→ integrando i risultati dell'ANOVA e dei test alternativi con l'esplorazione grafica dei dati;
→ traendo le conclusioni sulla base di una valutazione globale dei risultati ottenuti.

Il che ci ricorda che tutto sommato la statistica non dovrebbe essere altro che buonsenso, reso scientifico nella misura in cui si trasforma in un modello quantitativo e numerico, e adiuvato dalla rappresentazione grafica.

---

[1] Wonnacott TH, Wonnacott RJ. *Introduzione alla statistica*. Franco Angeli Editore, Milano, 1980, ISBN 88-204-0323-4, Tabella 10-1, p. 238.

[2] Wonnacott TH, Wonnacott RJ. *Introduzione alla statistica*. Franco Angeli Editore, Milano, 1980, ISBN 88-204-0323-4, Tabella 10-9, p. 255.

[3] L'analisi della varianza è basata su due ipotesi: (i) che i dati siano distribuiti in modo gaussiano e (ii) che la varianza sia la stessa nei diversi gruppi confrontati.

[4] Vedere il post Test parametrici e non parametrici per due campioni indipendenti.

[5] Vedere il post Test parametrici e non parametrici per più campioni indipendenti.

[6] Per i dettagli delle funzioni e degli argomenti impiegati per questa rappresentazione grafica si rimanda al post Grafici a punti (dotplot) mentre nel post Grafici a violino (violin plot) trovate anche come sovrapporre ai punti un grafico a violino o un grafico a scatola con i baffi (boxplot).

[7] Vedere il post Analisi della varianza a un fattore.

mercoledì 15 settembre 2021

Adattare i margini a un grafico

Può capitare di realizzare un grafico per il quale non sono adeguati i margini previsti di default per la finestra grafica di **R**. Vediamo il problema e una possibile soluzione.

Accertatevi innanzitutto di avere installato il pacchetto **DAAG** che contiene i dati che ci servono, o in alternativa procedete come indicato nel post Il set di dati ais nel quale trovate anche illustrato il significato dei dati impiegati [1].

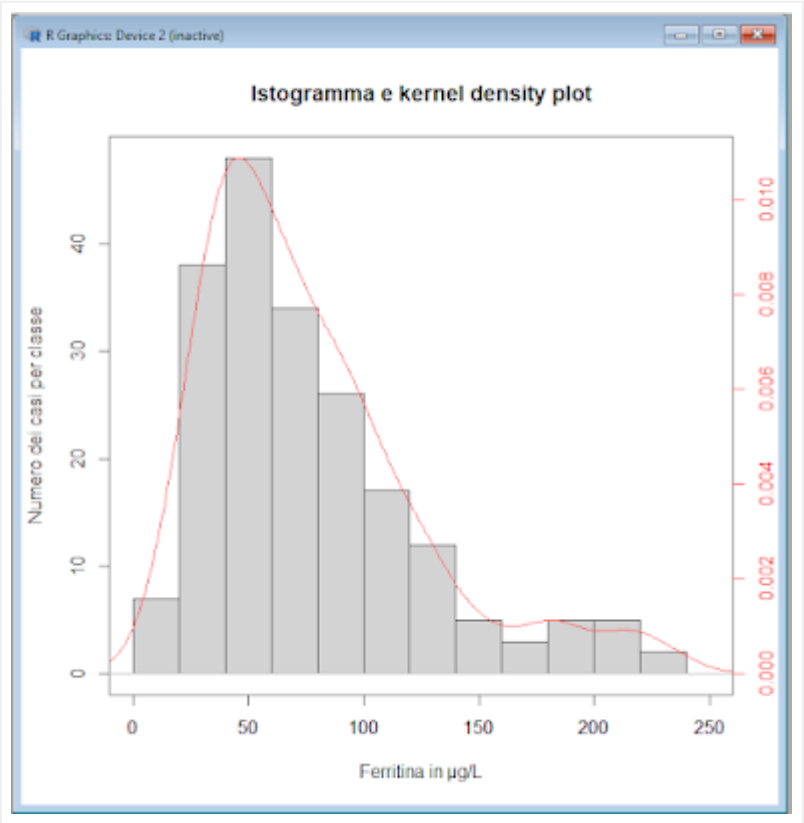
Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ADATTARE I MARGINI A UN GRAFICO
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
mydata <- ais[,c(5)] # salva in mydata i valori della variabile contenuta nella colonna 5 della tabella ais
#
par("mar") # mostra i margini di default della finestra grafica
#
# traccia istogramma e kernel density plot sovrapposti, con i margini di default
#
hist(mydata, main="Istogramma e kernel density plot", xlab="Ferritina in µg/L", ylab = "Numero dei casi per classe", xlim = c(0,250)) # traccia l'istogramma
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
plot(density(mydata), main="Kernel density plot", xlab="Ferritina in µg/L", ylab = "Frequenza", xlim = c(0,250), yaxt="n", col="red") # traccia il kernel density plot
axis(4, col.ticks="red", col.axis="red") # riporta l'asse delle y sulla destra
mtext("Stima kernel di densità", side=4, line=3) # aggiunge la legenda all'asse delle y sulla destra
#
# traccia istogramma e kernel density plot sovrapposti, con margine destro allargato
#
windows() # apre e inizializza una nuova finestra grafica
par(mar=c(5.1,4.1,4.1,5.1)) # aumenta il margine destro da 2.1 a 5.1
#
hist(mydata, main="Istogramma e kernel density plot", xlab="Ferritina in µg/L", ylab = "Numero dei casi per classe", xlim = c(0,250)) # traccia l'istogramma
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
plot(density(mydata), main="Kernel density plot", xlab="Ferritina in µg/L", ylab = "Frequenza", xlim = c(0,250), yaxt="n", col="red") # traccia il kernel density plot
axis(4, col.ticks="red", col.axis="red") # riporta l'asse delle y sulla destra
mtext("Stima kernel di densità", side=4, line=3) # aggiunge la legenda all'asse delle y sulla destra
#
par(mar=c(5.1,4.1,4.1,2.1)) # ripristina i margini di default
#
```

In questo caso il problema è causato dal desiderio di riportare sullo stesso grafico l'istogramma e il kernel density plot di una variabile. La scala sull'asse delle ascisse `x` è identica per entrambi, ma
→ nel caso dell'**istogramma** dobbiamo riportare sull'asse delle ordinate `y` il **numero di casi** per ciascuna delle classi in cui è suddiviso;
→ nel caso del **kernel density plot** dobbiamo riportare sull'asse delle ordinate `y` la **stima kernel di densità** espressa in termini di probabilità.

La soluzione più ovvia è di riportare due scale delle ordinate, una sulla sinistra riferita all'istogramma, e una sulla destra riferita al kernel density plot (o viceversa, non importa). Qui riportiamo il kernel density plot in rosso e i relativi valori sulla scale delle ordinate di destra nello stesso colore.

Con la prima parte dello script viene generato questo primo grafico con istogramma e kernel density plot sovrapposti



che impiega i margini di default della finestra grafica, i quali, richiamati nella terza riga di codice mediante la funzione **par("mar")**

```
> par("mar") # mostra i margini di default
[1] 5.1 4.1 4.1 2.1
```

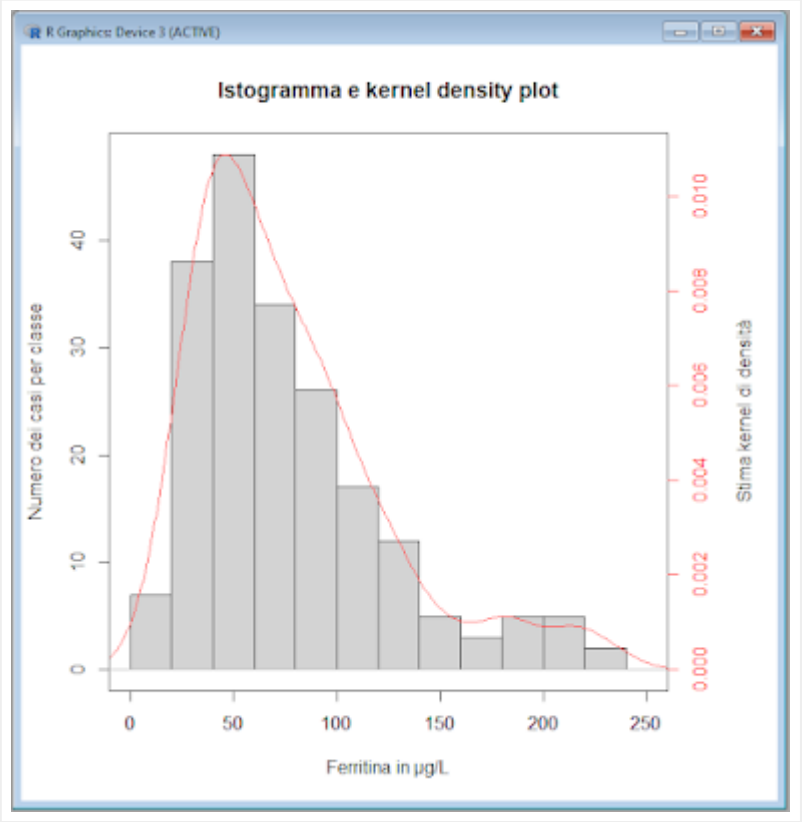
sono rispettivamente `5.1` per il margine inferiore, `4.1` per il margine sinistro, `4.1` per il margine superiore e `2.1` per il margine destro, espressi come numero di righe (per i dettagli digitare **help(par)** nella `Console di R`).

Come si vede la funzione **mtext()** riporta la legenda **"Stima kernel di densità"** dell'asse delle `y` sulla destra (**side=4**) nella terza riga di testo (**line=3**), ma questa non compare nella finestra grafica.

Nella seconda parte dello script il problema viene corretto impiegando lo stesso identico codice, ma facendolo precedere dalla funzione

par(mar=c(5.1,4.1,4.1,5.1))

che aumenta il margine destro dal valore di default di `2.1` a **5.1** riducendo lievemente la larghezza del grafico e lasciando così sulla destra spazio adeguato per la legenda dell'asse delle `y`, che nel grafico precedente non era visibile, e ottenendo questo secondo grafico [2].



Lo script si conclude con la funzione **par(mar=c(5.1,4.1,4.1,2.1))** che ripristina i valori di default dei margini (un passo non obbligatorio, ma altamente raccomandato).
La soluzione riportata può essere facilmente adattata ai diversi casi che si possono presentare nella pratica.

---

[1] Qui impieghiamo i valori - contenuti nella colonna 5 della tabella **ais** (vedere la seconda riga di codice dello script) - della variabile ferritina, una proteina che, pur con alcune limitazioni, fornisce una misura dei depositi di ferro presenti nell'organismo, necessari per garantire una adeguata produzione di emoglobina.

[2] Una banalità per chi non fosse ancora abituato a **R**: quando si esegue lo script, il secondo grafico viene sovrapposto al primo, quindi è necessario minimizzare la finestra del secondo grafico o afferrarla con il mouse e spostarla per vedere il grafico sottostante.

martedì 5 ottobre 2021

Loop e vettorializzazione

Questo blog ha come obiettivo quello di fornire gli strumenti per "*rompere il ghiaccio*" con **R**, lasciando poi agli eventuali interessi e all'iniziativa personale l'approfondimento del linguaggio di programmazione, che ciascuno potrà realizzare in base ai proprio bisogni. Tuttavia occasionalmente può essere interessante affrontare qualche argomento un poco più tecnico, come quello che vediamo oggi.

Il punto di partenza è rappresentato dal calcolo delle **statistiche elementari parametriche**. Per un campione che include n dati:
a) la **media**, la *misura di posizione dei dati*, viene calcolata come

$$\bar{x} = \sum_{i=1}^{i=n} x_i \,/\, n$$

b) la **deviazione standard**, la *misura di dispersione dei dati*, viene calcolata come

$$s = \sqrt{s^2}$$

essendo

$$s^2 = \sum_{i=1}^{i=n} (x_i - \bar{x})^2 \,/\, (n-1)$$

la **varianza** dei dati.

Invece di calcolare le statistiche elementari parametriche servendoci di qualcuna delle funzioni disponibili in **R** le vogliamo calcolare seguendo alla lettera la loro definizione matematica. Come si vede i calcoli sono elementari, e di fatto l'unico problema da risolvere in termini di programmazione è trovare un metodo per il **calcolo della sommatoria** Σ che ci serve per ottenere:
1) la somma di tutti i dati

$$\sum_{i=1}^{i=n} x_i$$

da dividere poi per il numero n dei dati al fine di calcolare la *media*;
2) la somma dei quadrati delle differenze tra ciascun dato e la media, ovvero la *devianza*

$$\sum_{i=1}^{i=n} (x_i - \bar{x})^2$$

da dividere poi per il numero dei dati meno uno per calcolare la *varianza* dalla quale otteniamo infine la *deviazione standard*.

Il primo metodo per il calcolo di una sommatoria che impieghiamo fa ricorso ad una **iterazione** (o **ciclo** o **loop**), una struttura di controllo che consente di eseguire ripetutamente

una o più istruzioni, fino al verificarsi di una condizione specificata.

Accertatevi innanzitutto di avere installato il pacchetto **psychTools** che contiene i dati che ci servono, o in alternativa procedete come indicato nel post Il set di dati galton nel quale trovate anche illustrato il significato dei dati impiegati. Quindi copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# STATISTICHE ELEMENTARI PARAMETRICHE con loop
#
library(psychTools) # carica il pacchetto che include il set di dati galton
x <- galton$parent # salva nel vettore x le altezze dei padri
#
n <- length(x) # calcola il numero dei dati
#
### (a)
#
sx <- 0 # inizializza la sommatoria dei dati
for (i in 1:n) {
   sx <- sx+x[i] # calcola la sommatoria dei dati
}
#
media <- sx/n # calcola la media
#
### (b)
#
devianza <- 0 # inizializza la sommatoria degli scarti quadratici (devianza)
#
for (i in 1:n) {
   devianza <- devianza+(x[i]-media)^2 # calcola la sommatoria degli scarti quadratici
}
#
varianza <- devianza/(n-1) # calcola la varianza
ds <- sqrt(varianza) # calcola la deviazione standard
#
### (c)
#
statistica <- c("numero dei dati =", "media =", "varianza =", "deviazione standard =") #
predispone i nomi delle statistiche calcolate
risultato <- c(n, media, varianza, ds) # predispone i risultati delle statistiche calcolate
print(data.frame(statistica, risultato), row.names=FALSE) # mostra la tabella dei risultati
#
```

Dopo avere caricato con la prima riga di codice il pacchetto che include i dati, con la seconda riga di codice al vettore **x** sono assegnati (**<-**) i valori della variabile **galton$parent** che vogliamo utilizzare (si rimanda nuovamente al post Il set di dati galton per la loro storia e il loro significato).

Ora che abbiamo in **x** i dati che dobbiamo elaborare impieghiamo (terza riga di codice) la funzione **length()** per calcolarne il numero **n**.

Il primo loop **(a)** ci serve per calcolare la sommatoria dei dati che chiamiamo **sx** e che come prima cosa (quarta riga) inizializziamo ponendola uguale a 0 (**sx <- 0**).

Per calcolare la sommatoria dei dati impieghiamo la funzione **for()** che esegue il codice compreso tra la parentesi graffa aperta **{** e la parentesi graffa chiusa **}**. Il loop **for (i in 1:n)** procede facendo variare l'indice **i** da **1** a **n** e, per ognuno dei valori assunti da **i**, somma in **sx** il valore del dato i-esimo **x[i]**. Alla prima iterazione verrà sommato in **sx** il valore del dato x_1, alla seconda iterazione

verrà aggiunto il valore del dato x_2, alla terza iterazione verrà aggiunto il valore del dato x_3, e così via fino al dato x_n, dopo il quale il loop si interrompe: a questo punto avremo in **sx** la sommatoria dei valori degli **n** dati. Questa prima parte si conclude con il calcolo della media dei dati ottenuta dividendo la sommatoria dei dati per il loro numero (**media <- sx/n**).

Il secondo loop **(b)** calcola la somma dei quadrati delle differenze tra ciascun dato e la media che chiamiamo **devianza** e che viene come prima cosa inizializzata ponendola uguale a 0 (**devianza <- 0**).

Quindi con un loop identico al precedente, facendo variare l'indice **i** da **1** a **n**, per ognuno dei valori assunti da **i** viene sommata nella variabile **devianza** la differenza tra ciascun dato e la media dei dati elevata al quadrato cioè **(x[i]-media)^2** ottenendo in tal modo nella variabile **devianza** la sommatoria desiderata, che divisa per **n - 1** ci consente di ottenere la varianza, la cui radice quadrata è infine la deviazione standard, come dalle formule riportate all'inizio.

La terza e ultima parte dello script **(c)** si limita a generare e a riportare nella `Console di R` una tabella con la sintesi dei risultati ottenuti:

```
            statistica  risultato
     numero dei dati = 928.000000
               media =  68.308190
            varianza =   3.194561
 deviazione standard =   1.787333
```

Il secondo metodo per il calcolo di una sommatoria impiega la **vettorializzazione**. Molte funzioni in **R** sono vettorializzate, il che significa che le operazioni specificate all'interno della funzione sono effettuate in parallelo su tutti gli elementi del vettore contenente i dati.

Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# STATISTICHE ELEMENTARI PARAMETRICHE con vettorializzazione
#
library(psychTools) # carica il pacchetto che include il set di dati galton
x <- galton$parent # salva la variabile nel vettore x
#
n <- length(x) # calcola il numero dei dati
#
### (a)
#
media <- sum(x)/n # calcola la media
#
### (b)
#
varianza <- sum((x-media)^2)/(n-1) # calcola la varianza
ds <- sqrt(varianza) # calcola la deviazione standard
#
### (c)
#
statistica <- c("numero dei dati =", "media =", "varianza =", "deviazione standard =") #
predispone i nomi delle statistiche calcolate
risultato <- c(n, media, varianza, ds) # predispone i risultati delle statistiche calcolate
print(data.frame(statistica, risultato), row.names=FALSE) # mostra la tabella dei risultati
#
```

Si vede chiaramente in che modo **R** realizza la vettorializzazione nella quinta riga di codice: la funzione **sum()** calcola la sommatoria dei quadrati delle differenze tra ciascun dato e la media (la devianza), necessaria a sua volta per il calcolo della varianza, impiegando direttamente come argomento il vettore **x** contenente gli n dati, ragion per cui l'espressione **(x-media)^2** si intende applicata, e viene applicata, a tutti gli n dati dati del vettore **x**. Confrontato con quello dello script precedente il codice della parte **(a)** e della parte **(b)** è molto più conciso e più facile da leggere, oltre ad essere più efficiente di quello impiegato da un linguaggio non vettorializzato, mentre la parte **(c)** dello script è semplicemente identica alla precedente (risultati inclusi).

Se volete infine controllare la correttezza dei calcoli effettuati con loop e con vettorializzazione potete confrontarne i risultati con quelli ottenuti con queste tre righe di codice che fanno direttamente ricorso alle funzioni statistiche standard della versione base di **R** (che si assumono fornire risultati corretti):

```
#
mean(galton$parent)
var(galton$parent)
sd(galton$parent)
#
```

In conclusione abbiamo visto attraverso un breve esempio come con **R** sia possibile effettuare la ripetizione ciclica di un calcolo, o più in generale di una qualsiasi operazione, sia ricorrendo alla classica **iterazione** (**loop**), sia impiegando funzioni che prevedono la **vettorializzazione** [1]. E questo ci ha consentito di illustrare due tecniche di programmazione semplici ma interessanti, che possono risultare utili nel corso dell'utilizzo di **R**.

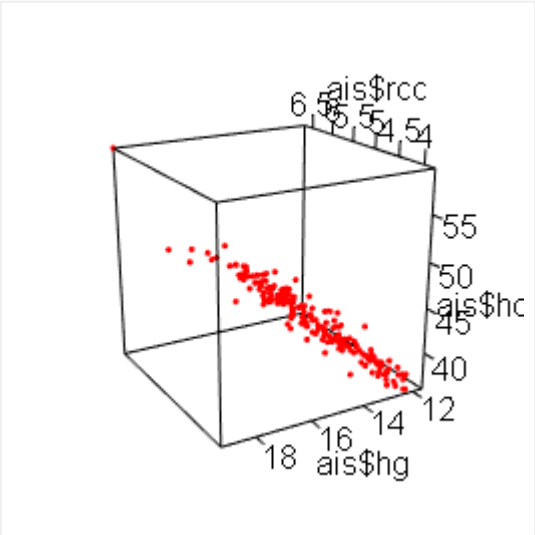
---

[1] R-bloggers. *Vectorization in R: Why?* Nell'articolo sono riportati anche i riferimenti ad alcune risorse sull'argomento. URL consultato il 20/09/2021: https://bit.ly/3nOsxuS

\bar{x}

lunedì 20 dicembre 2021

Grafici 3D



(l'animazione del grafico 3D è stata realizzata con l'ultimo script riportato in questo post,
per riavviarla riaprire la pagina web dal browser)

Con **R** è possibile realizzare grafici tridimensionali (grafici 3D) impiegando una sola riga di codice.

Per vedere come sia possibile farlo è necessario:
→ scaricare i quattro pacchetti aggiuntivi **scatterplot3d**, **plot3D**, **rgl** e **Rcmdr** [1];
→ installare il pacchetto **DAAG** che contiene nella tabella **ais** i dati impiegati come esempio [2].

Le tre variabili **x**, **y**, **z** utilizzate nelle rappresentazioni grafiche che seguono sono rispettivamente la concentrazione nel sangue degli eritrociti (**rcc**), espressa in milioni per microlitro di sangue (10^6/μL), la concentrazione nel sangue dell'emoglobina (**hg**), espressa in grammi per decilitro di sangue (g/dL) e il valore ematòcrito (**hc**), cioè la quota di sangue occupata dalla parte corpuscolata, espresso in percentuale (%) [3].

Ho riportato per i quattro pacchetti altrettanti script, limitati all'essenziale, che hanno il solo scopo di interessare a **R** per questo tipo di rappresentazione. Per i dettagli si rimanda ai manuali di riferimento dei pacchetti, che contengono la documentazione completa delle funzioni e dei rispettivi argomenti [4]. Ho poi aggiunto un quinto script che realizza un grafico 3D animato (quello riportato in apertura del post) per consentire di valutare anche questo aspetto di **R**.

Negli script le prime due righe di codice servono per caricare con **library(DAAG)** il pacchetto che contiene i dati (prima riga) e, di volta in volta, caricare il pacchetto di grafica (seconda riga) necessario per lo specifico grafico 3D. Questo viene infine realizzato con la funzione impiegata nell'ultima riga di codice (nel caso del grafico 3D animato nelle ultime due).

Poiché le variabili da rappresentare (**rcc**, **hg**, **hc**) sono contenute nella tabella **ais** (del pacchetto **DAAG**) il nome completo e univoco delle variabili è rappresentato dal prefisso **ais** più (**$**) il nome della variabile, pertanto in tre script trovate le variabili indicate come **ais$rcc**, **ais$hg**, **ais$c**.

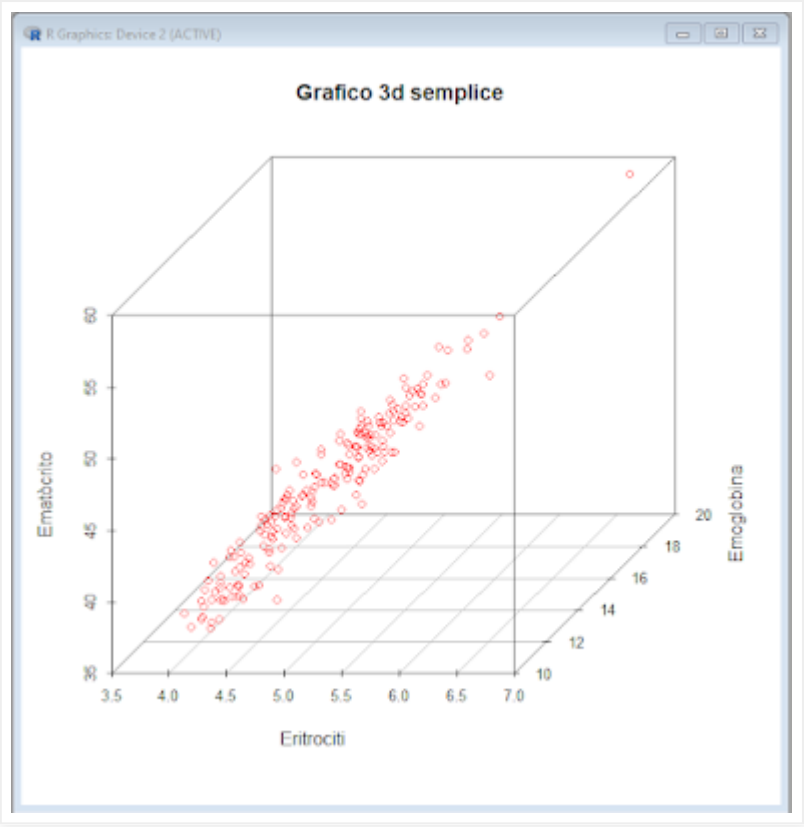
In altri due script è stata invece impiegata la funzione **attach()** che, una volta specificato come argomento il nome della tabella che contiene le variabili con **attach(ais)**, consente di impiegare i nomi originari delle variabili (quindi i nomi senza il prefisso **ais$**), che pertanto trovate riportati come **rcc**, **hg**, **hc**.

Copiate e incollate nella `Console di R` il primo script e premete `↵ Invio`:

```
# GRAFICO 3D SEMPLICE impiega il pacchetto scatterplot3d
#
library(DAAG) # carica il pacchetto che include il set di dati/tabella ais
library(scatterplot3d) # carica il pacchetto per la rappresentazione grafica 3D
#
# grafico 3D semplice
#
scatterplot3d(ais$rcc, y=ais$hg, z=ais$hc, color="red", type="p", main="Grafico 3d semplice", xlab="Eritrociti", ylab="Emoglobina", zlab="Ematòcrito")
#
```

Dopo avere caricato i due pacchetti necessari, il primo contenente i dati e il secondo contenente le funzioni grafiche, il grafico viene realizzato con la funzione **scatterplot3d()** specificando come argoment:
→ le variabili della tabella **ais** da rappresentare (**ais$rcc**, **ais$hg**, **ais$hc**);
→ il tipo (**type="p"**) di rappresentazione dei dati (**"p"** = punti, ma sono possibili le alternative **"h"** = punti con linee verticali che congiungono ciascun punto al piano x-y, e **"l"** = linee che congiungono i punti tra loro);
→ il colore dei punti (**color="red"**);
→ il titolo del grafico (**main="Grafico 3d semplice"**) e le etichette degli assi.

Questo è il grafico risultante:



Prima di eseguire ognuno dei successivi script uscite da **R** e poi rientrate (procedura raccomandata), oppure in alternativa chiudete la finestra contenente il grafico, quindi digitate nella `Console di R`

```
rm(list=ls(all=TRUE))
```

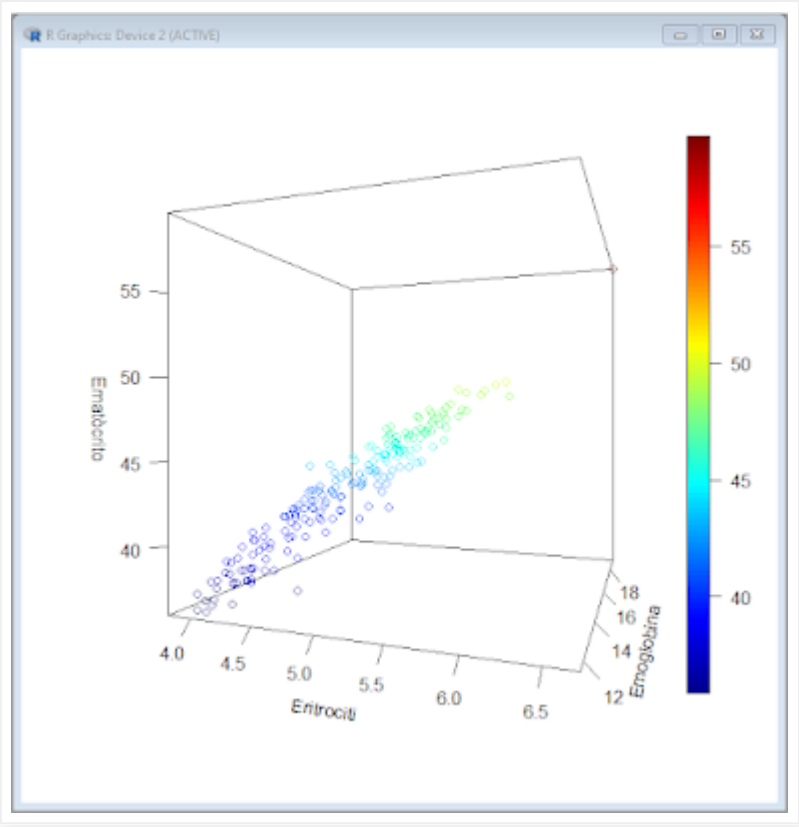
per effettuare la pulizia dell'area di lavoro [5].

Ora copiate e incollate nella `Console di R` questo secondo script e premete `↵ Invio`:

```
# GRAFICO 3D CON VISTA DA ANGOLI VARIABILI impiega il pacchetto plot3D
#
library(DAAG) # carica il pacchetto che include il set di dati/tabella ais
library(plot3D) # carica il pacchetto per la rappresentazione grafica 3D
#
# grafico 3D con vista da angoli variabili
#
scatter3D(ais$rcc, y=ais$hg, z=ais$hc, theta=20, phi=0, ticktype="detailed",
xlab="Eritrociti", ylab="Emoglobina", zlab="Ematòcrito")
#
```

Dopo avere caricato i due pacchetti necessari il grafico viene realizzo con la funzione **scatter3D()** specificando come argoment:
→ le variabili della tabella **ais** da rappresentare (**ais$rcc**, **ais$hg**, **ais$hc**);
→ l'angolo di rotazione del sistema di coordinate sull'asse verticale (**theta=20**);
→ l'angolo di rotazione del sistema di coordinate sull'asse orizzontale (**phi=0**);
→ la presenza sugli assi delle scale numeriche (**ticktype="detailed"**);
→ le etichette degli assi.

I punti sono riportati con una scala di colori che indica la loro distanza dal piano **x-y**, misurata lungo l'asse **z**. La scala colore/distanza è riportata nella legenda che compare sulla destra del grafico. Questo è il grafico:



Copiate e incollate nella `Console di R` il terzo script e premete `↵ Invio`:

```
# GRAFICO 3D INTERATTIVO CHE PUO' ESSERE RUOTATO CON IL MOUSE impiega il pacchetto rgl
#
library(DAAG) # carica il pacchetto che include il set di dati/tabella ais
library(rgl) # carica il pacchetto per la rappresentazione grafica 3D
#
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
# grafico 3D che può essere ruotato con il mouse
#
plot3d(rcc, y=hg, z=hc, type="p", col="red", size=3)
#
detach(ais) # termina l'impiego diretto dei nomi delle variabili
#
```

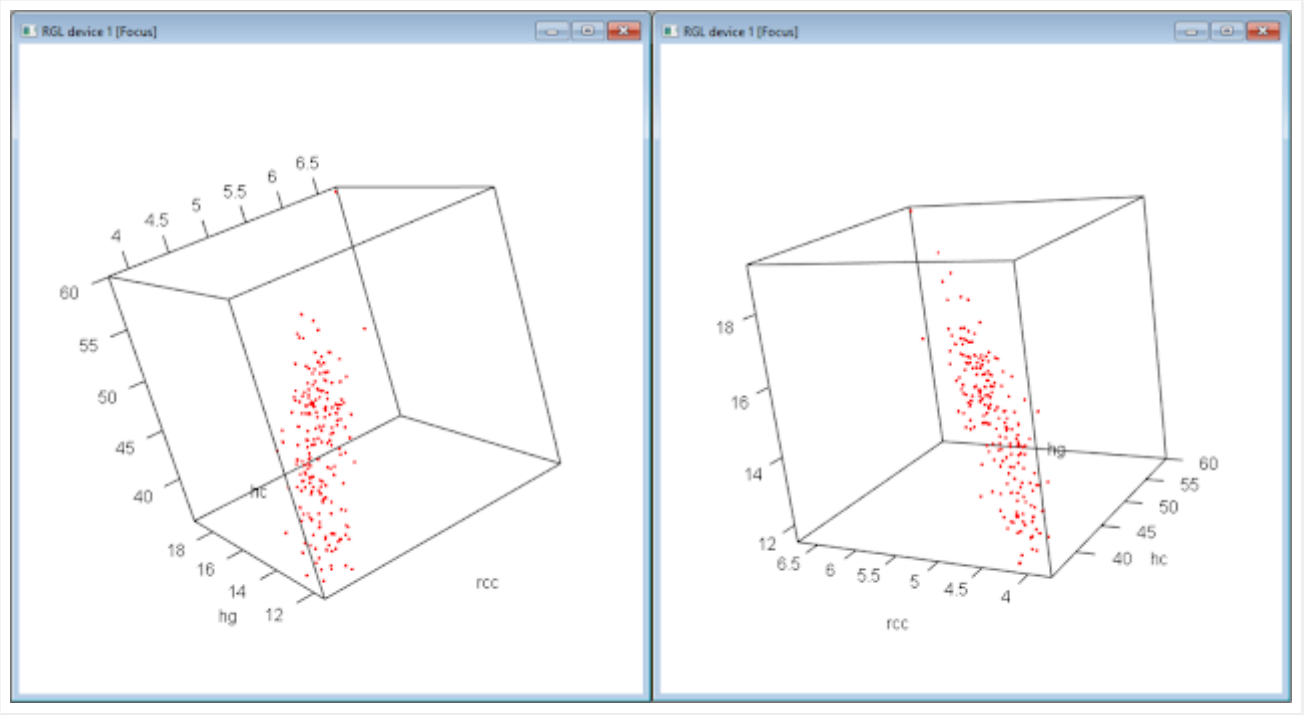
Dopo avere caricato i due pacchetti necessari, la funzione **attach(ais)** consente questa volta di realizzare il grafico nella riga successiva impiegando direttamente i nomi delle variabili contenute nella tabella **ais** senza specificare per ciascuna di esse la tabella nella quale sono incluse (notare quindi la differenza con i due script precedenti).

Il grafico è realizzato con la funzione **plot3d()** specificando come argomenti:
→ le variabili della tabella **ais** da rappresentare (**rcc**, **hg**, **hc**);
→ **type="p"** che consente di rappresentare i dati come punti;
→ **col="red"** che specifica il colore dei punti;
→ **size=3** che specifica la dimensione dei punti [6].

La funzione **detach()** tecnicamente, come riporta la documentazione "*Detach a database, i.e., remove it from the search() path of available R objects. Usually this is either a data.frame which has been attached or a package which was attached by library*", praticamente ripristina le condizioni precedenti al comando **attach()**.

Notare anche che, in questo e nei due script successivi, non essendo stati specificati gli argomenti **xlab**, **ylab** e **zlab**, **R** riporta automaticamente come etichette degli assi i nomi delle variabili.

Questa volta il grafico 3D è interattivo. Infatti se "afferrate" il grafico risultante facendo `click` su di esso e tenendo premuto il `tasto sinistro del mouse` senza rilasciarlo, potete ruotare il grafico a vostro piacimento muovendo il mouse. In questo modo sono state realizzate le due immagini che, qui mostrate affiancate, forniscono due delle possibili viste [della distribuzione] dei dati.



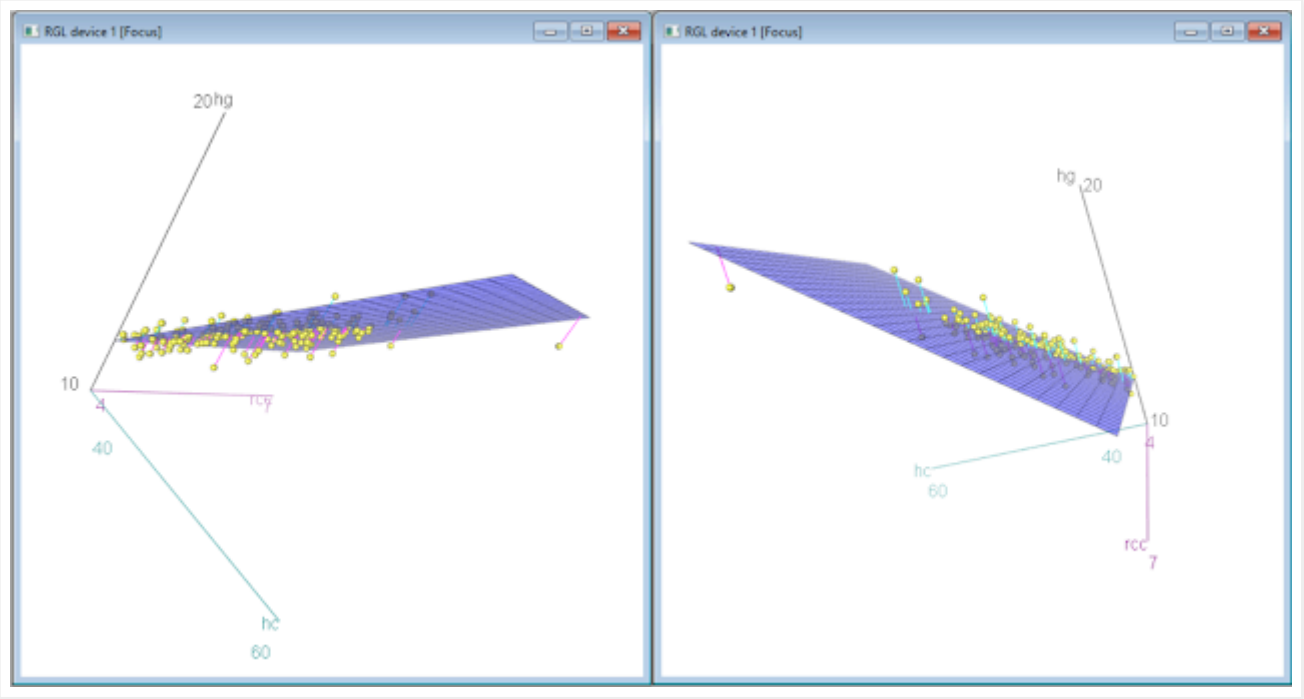
Copiate e incollate nella `Console di R` questo quarto script e premete ↲ `Invio`:

```
# GRAFICO 3D INTERATTIVO CHE PUO' ESSERE RUOTATO CON IL MOUSE impiega il pacchetto Rcmdr
#
library(DAAG) # carica il pacchetto che include il set di dati/tabella ais
library(Rcmdr) # carica il pacchetto per la rappresentazione grafica 3D
#
attach(ais) # consente di impiegare direttamente i nomi delle variabili
#
# grafico 3D che può essere ruotato con il mouse
#
scatter3d(rcc, y=hg, z=hc)
#
detach(ais) # termina l'impiego diretto dei nomi delle variabili
#
```

Dopo avere caricato i due pacchetti necessari, la funzione **attach(ais)** consente anche questa volta di realizzare il grafico impiegando direttamente i nomi delle variabili senza specificare per ciascuna di esse la tabella (**ais**) nella quale sono incluse (notare nuovamente la differenza con i primi due script).

Il grafico viene realizzato con la funzione **scatter3d()** specificando come argomento solamente le variabili della tabella **ais** da rappresentare (**rcc**, **hg**, **hc**), mentre per tutti gli altri argomenti, colori inclusi, sono stati lasciati i valori di default.

Come nel caso del grafico precedente si tratta di un grafico interattivo, e se lo "afferrate" [7] facendo `click` su di esso con il `tasto sinistro del mouse` e tenendolo premuto senza rilasciarlo, potete ruotarlo a vostro piacimento spostando il mouse. In questo modo sono state realizzate le immagini che, qui mostrate affiancate, forniscono due delle viste [della distribuzione] dei dati che è possibile ottenere.



Concludiamo con un grafico 3D animato, molto semplice, ma che apre in **R** un mondo, al quale si rimandano gli interessati [8].

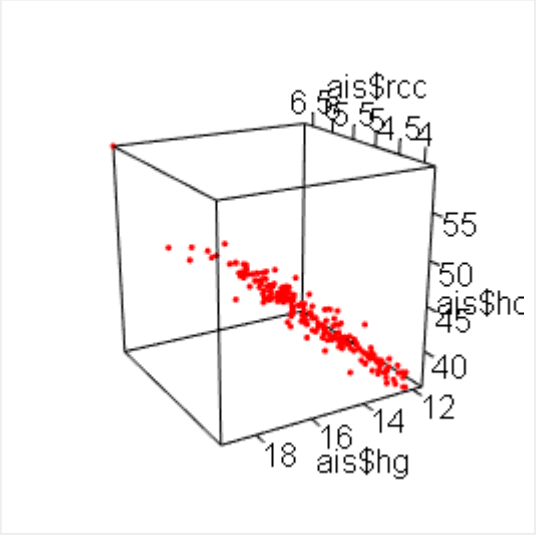
Copiate e incollate nella `Console di R` il quinto e ultimo script e premete `↵ Invio`:

```
# GRAFICO 3D ANIMATO impiega il pacchetto rgl
#
library(DAAG) # carica il pacchetto che include il set di dati/tabella ais
library(rgl) # carica il pacchetto per la rappresentazione grafica 3D
#
# grafico 3D animato
#
plot3d(ais$rcc, y=ais$hg, z=ais$hc, type="p", col="red", size=3)
play3d(spin3d(axis=c(0,0,1), rpm=10), duration=30)
#
```

Dopo avere caricato i due pacchetti necessari, il primo contenente i dati e il secondo contenente le funzioni grafiche, il grafico 3D viene realizzato con la funzione **plot3d()** del pacchetto **rgl** già impiegata nel terzo degli script riportati sopra (vedi).

Nella riga successiva con la funzione **play3d()** viene realizzata una animazione del grafico:
→ con i parametri specificati dalla funzione **spin3d()** che sono l'asse sul quale fare ruotare l'animazione, nel nostro caso l'asse delle **z** (**0,0,1**), e il numero di rotazioni al minuto (**rpm=10**).
→ delladurata di 30 secondi (**duration=30**)

Il grafico 3D animato risultante, che è quello riportato anche in apertura del post, è interessante non tanto per l'aspetto coreografico, ma piuttosto in quanto offre una serie di visuali che consentono di apprezzare appieno la relazione di linearità che intercorre tra le tre variabili rappresentate.



Nonostante gli script riportati sopra siano stati limitati all'essenziale [3], potete immediatamente riutilizzarli per una rappresentazione non banale dei vostri dati semplicemente sostituendo ai nomi delle variabili qui impiegati i nomi delle vostre variabili **x**, **y** e **z**.

Per una guida rapida all'importazione dei dati in **R** potete consultare i link:
→ importazione dei dati da un file `.csv`
→ importazione dei dati da un file `.xls` o `.xlsx`
→ gestione dei dati mancanti

Infine ricordo che nel post Salvare i grafici di R in un file è riportato uno script che consente di trasformare i grafici in immagini e salvarli sotto forma di file `.bmp`, `.jpeg`, `.png`, `.pdf`, `.ps` per poterli stampare, archiviare, inserire in una pubblicazione, in un post o in un sito web.

---

[1] *Pacchetti aggiuntivi* in quanto vanno ad aggiungersi al *pacchetto base* di **R**. Dopo avere scaricato il pacchetto **Rcmdr** chiudere e riavviare **R**, quindi digitare nella `Console di R` **library(Rcmdr)** e premere `↵ Invio` per scaricare dal **CRAN** gli ulteriori pacchetti aggiuntivi necessari al suo funzionamento, infine nuovamente chiudere e riavviare **R**.

[2] Se non volete installare il pacchetto potete in alternativa fare il download di un file contenente gli stessi dati, come riportato nel post Il set di dati ais e alla pagina Dati.

[3] La parte corpuscolata del sangue include eritrociti, leucociti e piastrine, ma dato che gli ultimi due in condizioni normali occupano un volume irrilevante, la quota di sangue occupata dalla parte corpuscolata (il valore ematòcrito) è di fatto quella occupata dagli eritrociti (globuli rossi).

[4] Per la documentazione completa delle funzioni e dei rispettivi argomenti vedere i manuali di riferimento dei pacchetti *scatterplot3d*, *plot3D*, *rgl* e *Rcmdr* sul repository della documentazione: *Available CRAN Packages By Name*. URL consultato il 10/11/2021: https://bit.ly/3zLwHWH

[5] Vedere il post Pulizia dell'area di lavoro e uscita dal programma

[6] Altre cose interessanti le trovate su *Impressive package for 3D and 4D graph - R software and data visualization*. URL consultato il 10/11/2021: https://bit.ly/3mKEeC1

[7] Attenzione perché il grafico potrebbe essere nascosto in parte o completamente dalla finestra di *R Commander* che si apre, al bisogno minimizzatela.

[8] Vedere ad esempio *Animated 3d chart with R* in*: The R Graph Gallery.* URL consultato il 10/11/2021: https://bit.ly/3BXySI4

venerdì 31 dicembre 2021

Regressione lineare multipla

La regressione lineare semplice (regressione bivariata) costruita sul piano cartesiano

$$y = a + b \cdot x$$

può essere estesa a uno spazio n-dimensionale: abbiamo così la **regressione lineare multipla** nella quale la variabile dipendente y viene fatta dipendere non più da una, bensì da k variabili indipendenti, indicate come x_1, x_2, x_3 … x_k secondo l'equazione

$$y = a + b_1 \cdot x_1 + b_2 \cdot x_2 + b_3 \cdot x_3 + \ldots + b_k \cdot x_k$$

Il risultato è rappresentato quindi da una intercetta a e da tanti coefficienti angolari b (indicati con b_1, b_2, b_3 ... b_k) quante sono le variabili indipendenti (x_1, x_2, x_3 ... x_k) che contribuiscono a determinare il valore della variabile dipendente y.

Come esempio impieghiamo il set di dati **ais** nel quale, tra i dati ematologici rilevati in un gruppo di atleti australiani, abbiamo tre grandezze:
→ la *concentrazione degli eritrociti* (globuli rossi) espressa in milioni per microlitro di sangue (10^6/µL) e contenuta nella variabile `rcc`;
→ il *valore ematòcrito* cioè la frazione del volume del sangue occupata dagli eritrociti espressa in percentuale (%) e contenuta nella variabile `hc`;
→ la *concentrazione dell'emoglobina* espressa in grammi per decilitro di sangue (g/dL) e contenuta nella variabile `hg`.

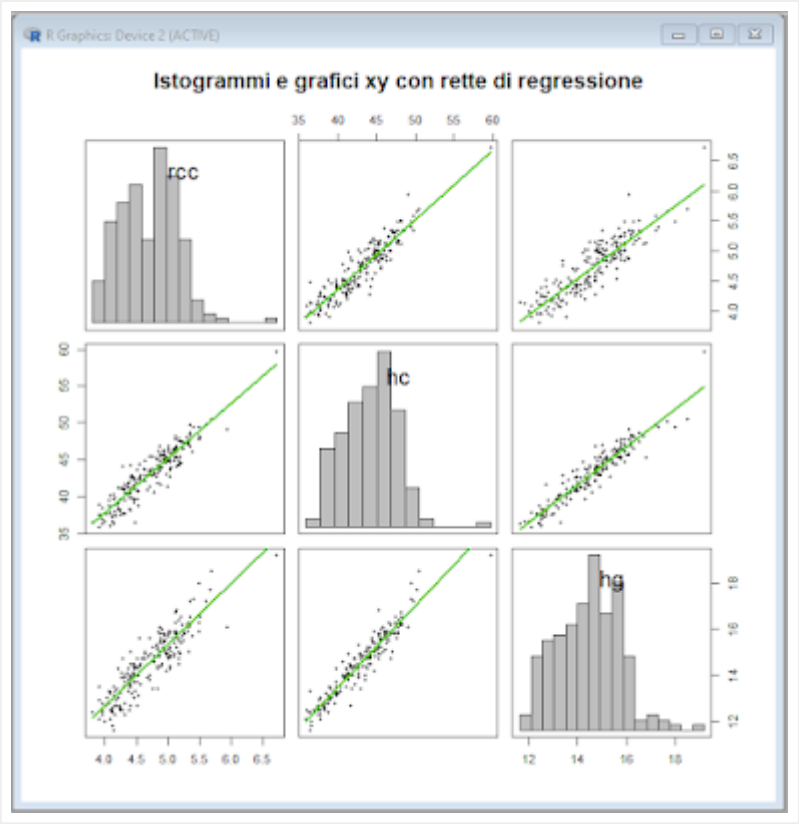
Le tre grandezze sono strettamente legate tra loro dal punto di vista biologico, in quanto gli eritrociti hanno ciascuno un certo volume e contengono ciascuno una data quantità di emoglobina: pertanto all'aumentare del numero di eritrociti aumenta la frazione percentuale del volume del sangue occupata dagli eritrociti (il valore ematòcrito) e aumenta la quantità di emoglobina. Mentre la diminuzione dell'ematòcrito e la diminuzione dell'emoglobina sono indice della diminuzione del numero degli eritrociti.

L'esempio è stato scomposto in una serie di script indipendenti - onde facilitare il riutilizzo del codice per analizzare i propri dati - che impiegano i pacchetti aggiuntivi **gvlma**, **car** e **rgl**, che devono essere scaricati dal **CRAN**. Accertatevi di avere installato inoltre il pacchetto **DAAG** che contiene i dati impiegati nello script, o in alternativa procedete come indicato nel post Il set di dati ais.

Iniziamo con una **analisi esplorativa preliminare** dei dati, copiate questo primo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# REGRESSIONE LINEARE MULTIPLA - analisi esplorativa preliminare dei dati
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(car) # carica il pacchetto per la grafica
mydata <- ais[,c(1,3,4)] # le colonne 1, 3, 4 contengono le variabili rcc, hc, hg
windows() # apre e inizializza una nuova finestra grafica
#
scatterplotMatrix(~rcc+hc+hg, col="black", pch=20, regLine=list(method=lm, lty=1,
lwd=2, col="chartreuse3"), smooth=FALSE, diagonal=list(method="histogram",
breaks="FD"), main="Istogrammi e grafici xy con rette di regressione", data=mydata) #
istogrammi e grafici xy con rette di regressione
#
```

Dopo avere caricato i pacchetti necessari, copiato nell'oggetto **mydata** le tre variabili che ci interessano, aperto e inizializzato una nuova finestra grafica, l'analisi preliminare dei dati viene effettuata con la funzione **scatterplotMatrix()** nella quale:
→ **~rcc+hc+hg** specifica le variabili da rappresentare, che sono la concentrazione degli eritrociti **rcc**, il valore ematòcrito **hc** e la concentrazone di emoglobia **hg**;
→ **col="..."** specifica il colore dei grafici;
→ **pch=...** specifica il tipo di simbolo da impiegare (un piccolo cerchio pieno);
→ per la retta di regressione **regLine** sono impiegati il modello lineare (**lm**), una linea continua (**lty=1**) con uno spessore doppio (**lwd=2**) e un colore (**col="chartreuse3"**), mentre l'argomento **smooth=FALSE** fa si che non venga aggiunta ai grafici la rappresentazione della funzione non lineare prevista di default nel pacchetto car;
→ l'argomento **diagonal=list()** indica la rappresentazione della variabile riportata nella diagonale, che qui è l'istogramma (**method="histogram"**) con il numero di classi (**breaks="FD"**) calcolato mediante la regola di Freedman-Diaconis.



Da notare che le espressioni che è possibile impiegare per l'argomento **diagonal=...** sono:
→ **list(method="boxplot")**;
→ **list(method="density", bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE)**;
→ **list(method="histogram")**;
→ **list(method="oned")**;
→ **list(method="qqplot")**.

Vista la relazione lineare molto buona che intercorre tra le variabili sembra ragionevole continuare ed effettuare il **calcolo della regressione lineare multipla**, copiate questo secondo script, incollatelo nella `Console di R` e premete `↲ Invio`:

```
# REGRESSIONE LINEARE MULTIPLA - calcolo della regressione lineare multipla
#
library(DAAG) # carica il pacchetto che include il set di dati ais
mydata <- ais[,c(1,3,4)] # le colonne 1, 3, 4 contengono le variabili rcc, hc, hg
reglin <- lm(rcc~hg+hc, data=mydata) # calcolo della regressione
#
```

coefficients(reglin) # intercetta e coefficienti angolari
confint(reglin, level=0.95) # intervalli di confidenza al 95%
#

La funzione **lm(...)** viene impiegata per calcolare la regressione lineare multipla specificando con l'espressione **rcc~hg+hc** la variabile dipendente (**rcc**), le due variabili indipendenti (**hg** e **hc**) e l'equazione

$$rcc = a + b_1 \cdot hg + b_2 \cdot hc$$

quindi possiamo ricavare con la funzione **coefficients()** i valori dell'intercetta (`a`), del coefficiente angolare di **hg** (b_1) e del coefficiente angolare di **hc** (b_2):

```
(Intercept)          hg          hc
-0.24269563  0.03283747  0.10403395
```

e riportare l'equazione della regressione lineare multipla come

$$rcc = -0.24269563 + 0.03283747 \cdot hg + 0.10403395 \cdot hc$$

Successivamente con la funzione **confint()** sono riportati gli intervalli di confidenza al 95% dell'intercetta e dei due coefficienti angolari:

```
                2.5 %     97.5 %
(Intercept) -0.53163317 0.04624192
hg          -0.02459817 0.09027311
hc           0.08267073 0.12539718
```

Da notare che gli intervalli di confidenza forniscono un test di significatività per intercetta e coefficienti angolari:
→ dato un intervallo di confidenza al `95%` che va da `-0.53163317` a `0.04624192` (che quindi include il valore `0`) l'intercetta non è significativamente diversa da `0`, cosa molto logica in quanto al diminuire fino ad azzerarsi dei valori di emoglobina ed ematòcrito deve necessariamente diminuire e azzerarsi il numero degli eritrociti;
→ con un intervallo di confidenza al `95%` che va da `-0.02459817` a `0.09027311` (che quindi include il valore `0`) il coefficiente angolare della concentrazione di emoglobina `hg` non è significativamente diversa da `0`, cosa meno logica in quanto non solo dal punto di vista biologico deve esistere una proporzionalità tra numero degli eritrociti e concentrazione dell'emoglobina, ma anche nell'analisi esplorativa preliminare dei dati riportata con lo script precedente risulta evidente la proporzionalità tra numero degli eritrociti e concentrazione dell'emoglobina che è una dato di fatto dal punto di vista biologico;
→ con un intervallo di confidenza al `95%` che va da `0.08267073` a `0.12539718` (che quindi non include il valore `0`) il coefficiente angolare del valore ematòcrito `hc` risulta significativamente diverso da `0`.

Come detto all'inizio, oltre ad effettuare una analisi esplorativa preliminare dei dati è opportuno sottoporre i risultati ad una valutazione critica a posteriori, che effettuiamo ora sotto forma di una **diagnostica numerica della regressione** lineare multipla, copiate il terzo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# REGRESSIONE LINEARE MULTIPLA - diagnostica numerica della regressione
#
library(gvlma) # carica il pacchetto per la diagnostica numerica
library(car) # carica il pacchetto per la grafica
library(DAAG) # carica il pacchetto che include il set di dati ais
mydata <- ais[,c(1,3,4)] # le colonne 1, 3, 4 contengono le variabili rcc, hc, hg
```

reglin <- lm(rcc~hg+hc, data=mydata) # calcolo della retta di regressione lineare multipla
#
summary(gvlma(reglin)) # test globale per l'assunto di linearità
outlierTest(reglin) # valore p di Bonferroni per la presenza di dati aberranti (outliers)
#

Ecco la prima parte del riepilogo dei risultati fornito dalla funzione **summary()**:

```
Call:
lm(formula = rcc ~ hg + hc, data = mydata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.55791 -0.11284 -0.00699  0.09950  0.53595

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.24270    0.14652  -1.656   0.0992 .
hg           0.03284    0.02913   1.127   0.2609
hc           0.10403    0.01083   9.603   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1744 on 199 degrees of freedom
Multiple R-squared:  0.8565,    Adjusted R-squared:  0.855
F-statistic: 593.8 on 2 and 199 DF,  p-value: < 2.2e-16
```

Il valore di probabilità `Pr(>|t|)` ottenuto con il test t di Student conferma (ovviamente) i risultati della significatività ricavati dagli intervalli di confidenza riportati sopra, per i quali valgono le considerazioni già fatte. E se la significatività statistica non è coerente con una realtà biologica assodata, dovrà ovviamente essere quest'ultima a prevalere: per cui assumiamo che l'equazione calcolata sia, fino a prova contraria, una descrizione sufficientemente adeguata della relazione di funzione che intercorre tra le tre variabili esaminate.

Che questo sia vero ce lo conferma la funzione **gvlma()**, acronimo di **g**lobal **v**alidation of **l**inear **m**odels **a**ssumptions, che fornisce questi risultati:
→ i dati sono distribuiti in modo lineare (`Global Stat`), assunto accettabile;
→ i dati non presentano asimmetria (`Skewness`), assunto accettabile;
→ i dati non presentano curtosi (`Skewness`), assunto accettabile;
→ nell'ambito dei valori osservati la varianza è costante (`Heteroscedasticity`), assunto accettabile.

```
ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = reglin)

                     Value p-value                   Decision
Global Stat        6.5801 0.15981    Assumptions acceptable.
Skewness           0.5149 0.47304    Assumptions acceptable.
Kurtosis           0.7559 0.38461    Assumptions acceptable.
Link Function      3.9844 0.04592 Assumptions NOT satisfied!
Heteroscedasticity 1.3249 0.24972    Assumptions acceptable.
```

L'assenza nei dati di asimmetria e di curtosi, e la loro omoschedasticità, sono condizioni sufficienti per garantire la loro adeguata descrizione mediante una regressione lineare. Il primo test (`Global Stat`) di valutazione globale degli assunti di linearità, che sostanzialmente comprende tutti e tre i test precedenti in forma compatta, li conferma. In questo contesto il test rimanente e l'unico discordante (`Link Function`) può essere ignorato.

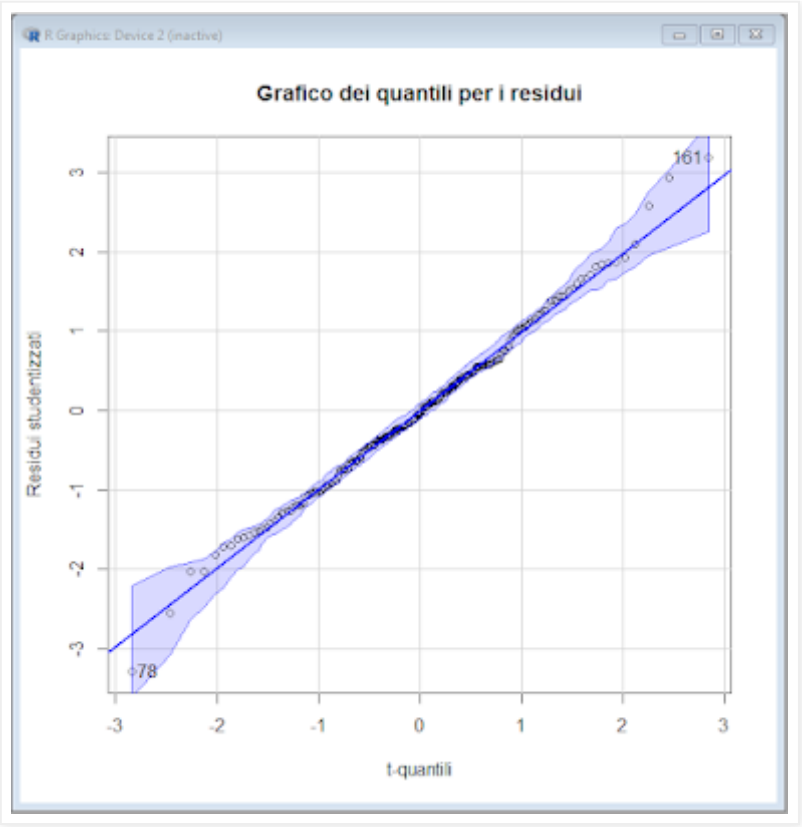
Con la funzione **outlierTest()** viene effettuato il *test di Bonferroni*, che non evidenzia dati anomali (*outliers*) ma segnala il dato `78` come quello che si discosta maggiormente dagli altri:

```
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferroni p
78 -3.28855          0.0011917      0.24073
```

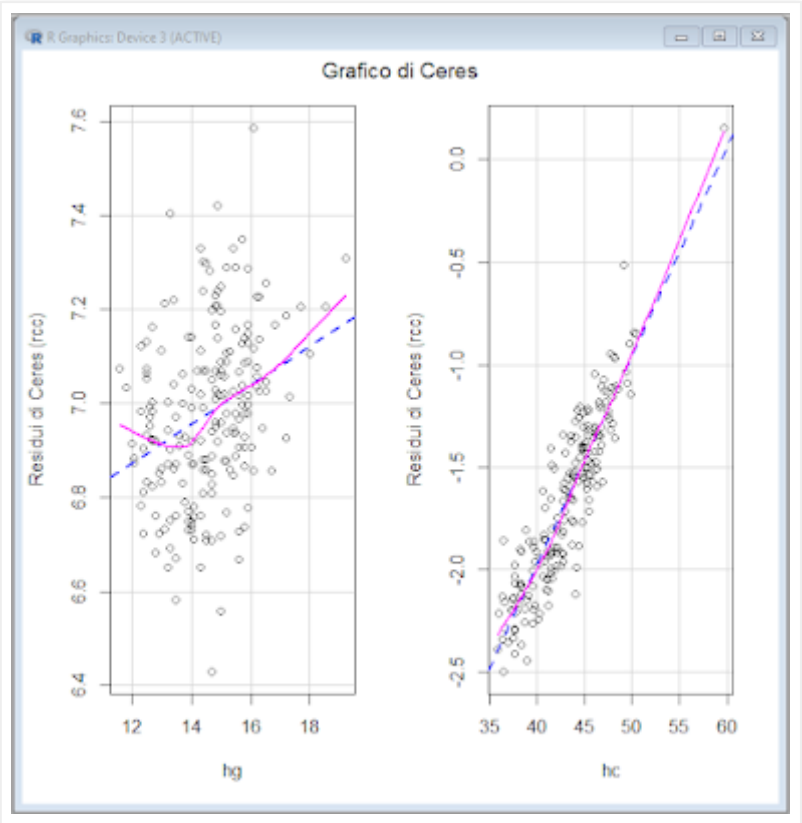
In tema di valutazione critica a posteriori, aggiungiamo ora alla diagnostica numerica anche una **diagnostica grafica della regressione** lineare multipla, per effettuarla copiate questo quarto script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# REGRESSIONE LINEARE MULTIPLA - diagnostica grafica della regressione
#
library(car) # carica il pacchetto per la grafica
library(DAAG) # carica il pacchetto che include il set di dati ais
mydata <- ais[,c(1,3,4)] # le colonne 1, 3, 4 contengono le variabili rcc, hc, hg
reglin <- lm(rcc~hg+hc, data=mydata) # calcolo della retta di regressione lineare multipla
#
windows() # apre e inizializza una nuova finestra grafica
qqPlot(reglin, main="Grafico dei quantili per i residui", xlab="t-quantili", ylab="Residui studentizzati") # grafico globale per linearità e dati aberranti
#
windows() # apre e inizializza una nuova finestra grafica
ceresPlots(reglin, ask=FALSE, main="Grafico di Ceres", ylab="Residui di Ceres (rcc)") # grafico di Ceres per la valutazione separata della linearità di hg e hc vs. rcc
#
windows() # apre e inizializza una nuova finestra grafica
plot(mydata$rcc, reglin$fitted.values, xlim=c(4,7), ylim=c(4,7)) # grafico valori originari vs valori stimati
abline(0, 1, lty=2) # linea di indentità y = x
#
```

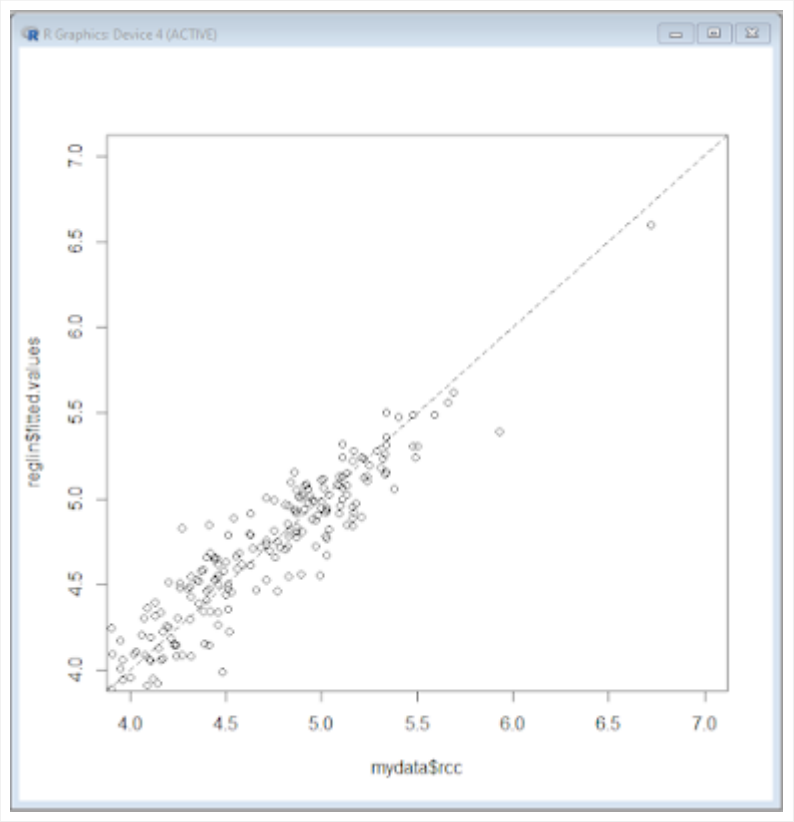
La funzione **qqPlot()** riporta una distribuzione dei residui lineare, evidenziando nel contempo i dati `78` e `161` come quelli che dei quali sarebbe utile controllare la validità, ovvero che sono situati in intervalli di valori per i quali potrebbe essere opportuno acquisire più dati.



La funzione **ceresPlot()** genera due grafici di Ceres separati per emoglobina ed ematòcrito, e consente di visualizzare una relazione lineare tra ematòcrito ed eritrociti migliore di quella che intercorre tra emoglobina ed eritrociti, confermando in un certo senso i valori di significatività riportati sopra.



La diagnostica grafica della regressione lineare multipla si conclude con questo grafico minimalista, ma che entra nell'essenza dei risultati.



Il grafico, generato con la funzione **plot()**, riporta in ascisse i valori originari della concentrazione degli eritrociti (**mydata$rcc**) e in ordinate i corrispondenti valori **reglin$fitted.values** ricalcolati da `hg` e `hc` impiegando l'equazione della regressione lineare multipla ove

reglin$fitted.values = `-0.24269563 + 0.03283747·hg + 0.10403395·hc`

La linea di identità `y = x` che viene riportata evidenzia, senza la necessità di ulteriori statistiche, che l'informazione acquisita misurando la concentrazione dell'emoglobina e l'ematòcrito può essere compressa in un unico valore, che fornisce una stima mediamente adeguata della concentrazione degli eritrociti.

Nota: con "*mediamente adeguata*" si intende che, data la misura diretta della concentrazione degli eritrociti riportata in ascisse, la sua misura indiretta - ottenuta misurando emoglobina ed ematòcrito ed applicando la regressione lineare multipla - comporta una perdita di informazione [aumento dell'incertezza della misura] che si riflette nella "*dispersione media*" dei valori sull'asse delle ordinate. Se non ci fosse perdita di informazione i punti sarebbero perfettamente allineati sulla retta `y = x`.

Ora, visto che con tre variabili la cosa è ancora fattibile, non resta che aggiungere un grafico tridimensionale (3D).

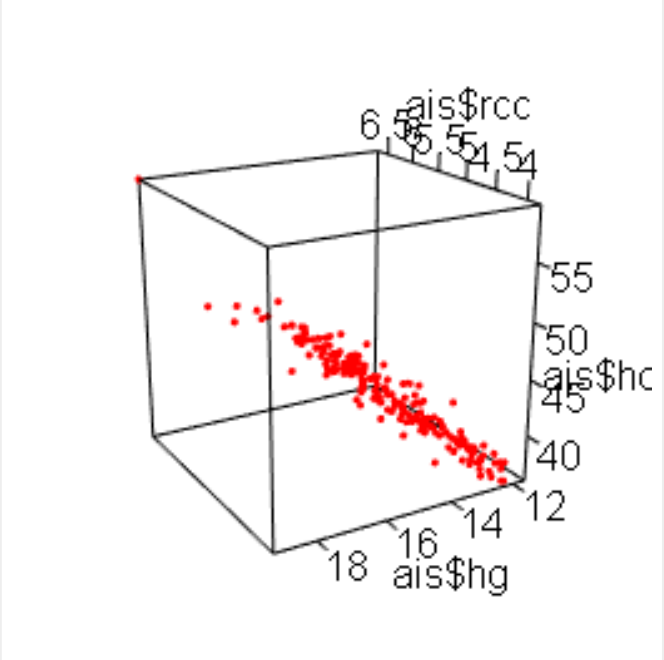
Copiate quest'ultimo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GRAFICO 3D ANIMATO con il pacchetto rgl
#
library(rgl) # carica il pacchetto per la grafica 3D
library(DAAG) # carica il pacchetto che include il set di dati ais
mydata <- ais[,c(1,3,4)] # le colonne 1, 3, 4 contengono le variabili rcc, hc, hg
windows() # apre e inizializza una nuova finestra grafica
#
# grafico 3D animato
#
```

plot3d(mydata$rcc, y=mydata$hg, z=mydata$hc, type="p", col="red", size=3) # realizza il grafico
play3d(spin3d(axis=c(0,0,1), rpm=10), duration=10) # anima il grafico
#

Dopo avere caricato il pacchetto necessario con il comando **library(rgl)** il grafico viene realizzato con la funzione **plot3d()**.

Quindi con la funzione **play3d()** viene realizzata una animazione della durata di 10 secondi (**duration=10**) con i parametri specificati dalla funzione **spin3d()** e cioè l'asse sul quale fare ruotare l'animazione, nel nostro caso l'asse delle **z** (**0,0,1**), e il numero di rotazioni al minuto (**rpm=10**).



Quando l'animazione che compare nella finestra "*RGL device*" aperta nella `Console di R` dallo script si ferma, se "afferrate" il grafico 3D facendo click con il tasto sinistro del mouse e tenendolo premuto senza rilasciarlo potete ruotarlo a vostro piacimento muovendo il mouse.

Anche se non aggiunge nulla di nuovo, il grafico [1] ci conforta documentando la linearità della relazione tra le tre variabili.

Conclusione: mettere in relazione tra di loro troppe variabili contemporaneamente può portare a risultati di difficile comprensione. Tuttavia una analisi esplorativa preliminare dei dati e una attenta valutazione critica a posteriori dei risultati possono aiutare a identificare i casi nei quali l'impiego della regressione lineare multipla può essere appropriato.

---

[1] Trovate altre modalità di rappresentazione sotto forma di grafici tridimensionali nel post Grafici 3D.

lunedì 10 gennaio 2022

Gestire le date del calendario

Se avete la necessità di gestire le **date del calendario**, dovete partire dal fatto che **R** assume essere date del calendario quelle contenute in oggetti della classe "`Date`" ed espresse in uno specifico formato.

Vediamo di capire questa cosa con lo script che segue, copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# GESTIRE LE DATE DEL CALENDARIO CON R - parte prima
#
# (1/5)
lemiedate <- c("13/05/1998", "16/12/1945", "23/09/2012") # i dati inseriti
lemiedate # il contenuto dell'oggetto
class(lemiedate) # la classe dell'oggetto
str(lemiedate) # la struttura dell'oggetto
#
# (2/5)
con_asDate <- as.Date(lemiedate, format="%d/%m/%Y") # i dati  trasformati
con_asDate # il contenuto dell'oggetto
class(con_asDate) # la classe dell'oggetto
str(con_asDate) # la struttura dell'oggetto
#
# (3/5)
con_format <- format(con_asDate, "%d/%m/%Y") # i dati trasformati
con_format # il contenuto dell'oggetto
class(con_format) # la classe dell'oggetto
str(con_format) # la struttura dell'oggetto
#
# (4/5)
as.Date("23/09/2012", format="%d/%m/%Y") # dati in formato gg/mm/aaaa
#
as.Date("09/23/2012", format="%m/%d/%Y") # dati in formato mm/gg/aaaa
#
as.Date("2012/09/23", format="%Y/%m/%d") # dati in formato aaaa/mm/gg
#
as.Date("23 settembre 2012", format="%d %B %Y") # dati in formato gg mese aaaa con spazio come separatore
#
as.Date("09-3-2012", format="%m-%d-%Y") # dati in formato mm-gg-aaaa con - come separatore
#
# (5/5)
library(lubridate)
dmy(c("13/05/1998", "16/12/1945", "23/09/2012"))
#
```

Questo è il risultato della prima parte dello script:

```
> # (1/5)
> lemiedate <- c("13/05/1998", "16/12/1945", "23/09/2012") # i dati inseriti
> lemiedate # il contenuto dell'oggetto
[1] "13/05/1998" "16/12/1945" "23/09/2012"
```

```
> class(lemiedate) # la classe dell'oggetto
[1] "character"
> str(lemiedate) # la struttura dell'oggetto
 chr [1:3] "13/05/1998" "16/12/1945" "23/09/2012"
> #
```

In pratica siamo partiti dal presupposto che
→ 13/05/1998
→ 16/12/1945
→ 23/09/2012
sono tre date del calendario che vogliamo gestire con **R**. Le abbiamo inserite nell'oggetto **lemiedate** con la funzione **c()**

```
> lemiedate <- c("13/05/1998", "16/12/1945", "23/09/2012") # i dati inseriti
```

le abbiamo ritrovate digitando il nome dell'oggetto

```
> lemiedate # il contenuto dell'oggetto
[1] "13/05/1998" "16/12/1945" "23/09/2012"
```

con la funzione **class()** che mostra la classe di un oggetto vediamo che **R** ha identificato le tre date come "character"

```
> class(lemiedate) # la classe dell'oggetto
[1] "character"
```

e con la funzione **str()** che mostra la struttura interna di un oggetto ne abbiamo riportato un ulteriore riepilogo sintetico

```
> str(lemiedate) # la struttura dell'oggetto
 chr [1:3] "13/05/1998" "16/12/1945" "23/09/2012"
```

nel quale "chr" sta di nuovo per "character", la classe dell'oggetto.

Quindi inserendo i dati con la funzione **c()** quelle che per noi erano tre date sono state identificate in **R** come "character", cioè come semplici stringhe o in altre parole come tre sequenze di caratteri alfanumerici: in pratica invece delle date avremmo potuto inserire "ciao Giovanni", "domani farà bel tempo", "viva la pappa col pomodoro”" o quant'altro.

Vediamo ora cosa accade con il successivo e secondo blocco di codice:

```
> # (2/5)
> con_asDate <- as.Date(lemiedate, format="%d/%m/%Y") # i dati  trasformati
> con_asDate # il contenuto dell'oggetto
[1] "1998-05-13" "1945-12-16" "2012-09-23"
> class(con_asDate) # la classe dell'oggetto
[1] "Date"
> str(con_asDate) # la struttura dell'oggetto
 Date[1:3], format: "1998-05-13" "1945-12-16" "2012-09-23"
> #
```

Questa volta impieghiamo la funzione **as.Date()** per inserire **lemiedate** (l'oggetto della classe "character" contenente tre stringhe alfanumeriche) in un nuovo oggetto denominato **con_asDate** (giusto per ricordarci come lo abbiamo generato)

```
> con_asDate <- as.Date(lemiedate, format="%d/%m/%Y") # i dati  trasformati
```

Dopo avere ritrovato quello che abbiamo inserito digitando il nome dell'oggetto

```
> con_asDate # il contenuto dell'oggetto
[1] "1998-05-13" "1945-12-16" "2012-09-23"
```

con la funzione **class()**

```
> class(con_asDate) # la classe dell'oggetto
[1] "Date"
```

e con la funzione **str()**

```
> str(con_asDate) # la struttura dell'oggetto
 Date[1:3], format: "1998-05-13" "1945-12-16" "2012-09-23"
> #
```

vediamo che **con_asDate** è un oggetto della classe "`Date`" nel quale in **R** per definizione sono contenute date del calendario

L'argomento **format="...."** impiegato nella funzione **as.Date()** specifica sia l'ordine sia il formato dei dati in ingresso con i seguenti codici:

Codice	Valore
%d	Giorno del mese (numero decimale)
%m	Mese (numero decimale)
%b	Mese (abbreviato)
%B	Mese (nome esteso)
%y	Anno (2 cifre)
%Y	Anno (4 cifre)

Se ora nella `Console di R` digitate **help(as.Date)** si apre una pagina web con la definizione della funzione in questione, che è definita come "*la funzione che converte le rappresentazioni [di date] sotto forma di caratteri [stringhe di testo] in oggetti della classe "`Date`" che rappresentano date di calendario*".

Come certamente avrete notato una volta importate in **R** con la funzione **asDate()** le date sono state convertite nel formato *aaaa-mm-gg*. A questo punto la domanda è d'obbligo: e se volessi tornare a convertirle nel formato *gg/mm/aaaa*? La cosa è possibile, e vediamo come con la terza parte di codice:

```
> # (3/5)
> con_format <- format(con_asDate, "%d/%m/%Y") # i dati trasformati
> con_format # il contenuto dell'oggetto
[1] "13/05/1998" "16/12/1945" "23/09/2012"
> class(con_format) # la classe dell'oggetto
[1] "character"
> str(con_format) # la struttura dell'oggetto
 chr [1:3] "13/05/1998" "16/12/1945" "23/09/2012"
> #
```

Con la funzione **format()** - da non confondere con l'argomento **format=** della funzione **asDate()** - possiamo convertire le date dal formato *aaaa-mm-gg* nel quale erano nell'oggetto **con_asDate** al formato *gg/mm/aaaa* specificato con l'argomento "**%d/%m/%Y**"

```
> con_format <- format(con_asDate, "%d/%m/%Y") # i dati trasformati
```

Questo può essere utile ad esempio se vogliamo visualizzare o stampare le date in questo formato. Ma in questo modo le date per **R** non sono più tali. Se esaminiamo il nuovo oggetto **con_format**

```
> con_format # il contenuto dell'oggetto
[1] "13/05/1998" "16/12/1945" "23/09/2012"
```

con la funzione **class()**

```
> class(con_format) # la classe dell'oggetto
[1] "character"
```

e con la funzione **str()**

```
> str(con_format) # la struttura dell'oggetto
 chr [1:3] "13/05/1998" "16/12/1945" "23/09/2012"
> #
```

vediamo che si tratta di un oggetto di classe "`character`": imponendo il formato *gg/mm/aaaa* le date tornano ad essere nuovamente delle stringhe ovvero generiche sequenze di caratteri alfanumerici esattamente come quelle da cui eravamo partiti.

Quindi concludendo: le date del calendario in **R** sono per definizione quelle contenute in un oggetto della classe "`Date`" ed espresse nel formato *aaaa-mm-gg*.

Vediamo ora alcuni altri esempi di date importate in **R** mediante la funzione **as.Date()**:

```
> # (4/5)
> as.Date("23/09/2012", format="%d/%m/%Y") # dati in formato gg/mm/aaaa
[1] "2012-09-23"
> #
> as.Date("09/23/2012", format="%m/%d/%Y") # dati in formato mm/gg/aaaa
[1] "2012-09-23"
> #
> as.Date("2012/09/23", format="%Y/%m/%d") # dati in formato aaaa/mm/gg
[1] "2012-09-23"
> #
> as.Date("23 settembre 2012", format="%d %B %Y") # dati in formato gg mese aaaa con
spazio come separatore
[1] "2012-09-23"
> #
> as.Date("09-3-2012", format="%m-%d-%Y") # dati in formato mm-gg-aaaa con - come
separatore
[1] "2012-09-03"
> #
```

I risultati confermano che se specificate opportunamente l'argomento **format=** dei dati in ingresso potete rappresentare sotto forma di stringhe alfanumeriche le vostre date predisposte:
→ nel formato *gg/mm/aaaa*
→ nel formato *mm/gg/aaaa*
→ nel formato *aaaa/gg/mm*
→ nel formato *gg mese aaaa* impiegando lo spazio come separatore
→ nel formato *mm-gg-aaaa* impiegando come separatore il tratto a metà altezza -
ovvero in altri modi purché sia correttamente e completamente specificato il formato dei dati in ingresso.

Se installate il pacchetto **lubridate** potete infine fruire di una sintassi più concisa come nell'esempio che segue, che porta ovviamente allo stesso risultato

```
> # (5/5)
> library(lubridate)
> dmy(c("13/05/1998", "16/12/1945", "23/09/2012"))
[1] "1998-05-13" "1945-12-16" "2012-09-23"
```

Vediamo ora cosa accade nel caso di **date importate da un file** in formato Excel (`.xls` o `.xlsx`) o, meglio ancora, in formato `.csv` (il formato dei dati da importare consigliato da **R**).

Andate alla pagina Dati nella quale trovate diverse opzioni per scaricare i file di dati, quindi scaricate e copiate nella cartella `C:\Rdati\` il file `PA.xls` e il file `PA.csv`

Nei file sono riportati i valori della pressione arteriosa sistolica e diastolica e della frequenza cardiaca registrati alle date del calendario riportate nella variabile "`Data`" (prima colonna).

	A	B	C	D
1	Data	Sistolica	Diastolica	Frequenza
2	07/05/2021	112	64	50
3	08/05/2021	118	69	49
4	08/05/2021	101	57	62
5	09/05/2021	111	65	56
6	09/05/2021	117	67	60
7	10/05/2021	110	69	59
8	10/05/2021	109	63	53
9	16/05/2021	115	66	49
10	17/09/2021	121	65	48
11	17/09/2021	92	58	63
12	17/09/2021	131	79	68
13	17/09/2021	116	68	57
14	29/09/2021	114	69	56

Se ancora non l'avete fatto, scaricate dal **CRAN** e installate il pacchetto **xlsx**, quindi copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

```
# GESTIRE LE DATE DEL CALENDARIO CON R - parte seconda
#
# importare le date da un file .xls
#
require(xlsx) # carica il pacchetto xlsx
PA_xls <- read.xlsx("C:/Rdati/PA.xls", sheetName="M") # importa i dati del file e del foglio specificati
PA_xls # mostra la tabella che contiene i dati
str(PA_xls) # la variabile Data viene riconosciuta come una data del calendario (classe "Date")
#
# importare le date da un file .csv
#
PA_csv <- read.table("C:/Rdati/PA.csv", header=TRUE, sep=",") # carica i dati del file specificato
PA_csv # mostra la tabella che contiene i dati
str(PA_csv) # la variabile Data viene riconosciuta come una semplice stringa di testo (classe "character")
#
PA_csv$Data <- as.Date(PA_csv$Data, format="%d/%m/%Y") # la stringa Data viene convertita con as.Date
str(PA_csv) # la variabile Data viene ora riconosciuta come una data del calendario
#
# elaborare le date (esempi)
#
Sys.Date()
PA_xls$Data[13] - PA_xls$Data[1]
PA_xls$Data[13] + 32
weekdays(PA_csv$Data[1])
#
```

Dopo avere caricato il pacchetto **xlsx**

```
> require(xlsx) # carica il pacchetto xlsx
Caricamento del pacchetto richiesto: xlsx
```

i dati sono importati dal foglio `M` del file `PA.xls`

```
> PA_xls <- read.xlsx("C:/Rdati/PA.xls", sheetName="M") # importa i dati del file e del foglio specificati
```

e richiamando il nome dell'oggetto (**PA_xls**) nel quale sono stati importati vediamo che i dati sono stati importati correttamente

```
> PA_xls # mostra la tabella che contiene i dati
         Data Sistolica Diastolica Frequenza
1  2021-05-07       112         64        50
2  2021-05-08       118         69        49
3  2021-05-08       101         57        62
4  2021-05-09       111         65        56
5  2021-05-09       117         67        60
6  2021-05-10       110         69        59
7  2021-05-10       109         63        53
8  2021-05-16       115         66        49
9  2021-09-17       121         65        48
10 2021-09-17        92         58        63
11 2021-09-17       131         79        68
```

```
12 2021-09-17       116         68        57
13 2021-09-29       114         69        56
```

La funzione **str()** ci conferma che anche la variabile `Data` è stata correttamente identificata come "`Date`" quindi il pacchetto **xlsx** ha trasferito i dati della prima colonna in modo tale che **R** li ha riconosciuti e li riporta come date del calendario.

```
> str(PA_xls) # la variabile Data viene riconosciuta come una data del calendario
(classe "Date")
'data.frame':   13 obs. of  4 variables:
 $ Data      : Date, format: "2021-05-07" "2021-05-08" ...
 $ Sistolica : num  112 118 101 111 117 110 109 115 121 92 ...
 $ Diastolica: num  64 69 57 65 67 69 63 66 65 58 ...
 $ Frequenza : num  50 49 62 56 60 59 53 49 48 63 ...
> #
```

Vediamo ora cosa accade quando si importano i dati da un file `.csv`

Dopo avere importato i dati nell'oggetto **PA_csv**

```
> # importare le date da un file .csv
> #
> PA_csv <- read.table("C:/Rdati/PA.csv", header=TRUE, sep=",") # carica i dati del
file specificato
```

e averli visualizzati richiamando il nome dell'oggetto nel quale sono stati importati

```
> PA_csv # mostra la tabella che contiene i dati
         Data Sistolica Diastolica Frequenza
1  07/05/2021       112         64        50
2  08/05/2021       118         69        49
3  08/05/2021       101         57        62
4  09/05/2021       111         65        56
5  09/05/2021       117         67        60
6  10/05/2021       110         69        59
7  10/05/2021       109         63        53
8  16/05/2021       115         66        49
9  17/09/2021       121         65        48
10 17/09/2021        92         58        63
11 17/09/2021       131         79        68
12 17/09/2021       116         68        57
13 29/09/2021       114         69        56
```

con la funzione **str()** vediamo che questa volta la variabile `Data` è stata identificata come una stringa / generica sequenza di caratteri alfanumerici (`chr`)

```
> str(PA_csv) # la variabile Data viene riconosciuta come una semplice stringa di
testo (classe "character")
'data.frame':   13 obs. of  4 variables:
 $ Data      : chr  "07/05/2021" "08/05/2021" "08/05/2021" "09/05/2021" ...
 $ Sistolica : int  112 118 101 111 117 110 109 115 121 92 ...
 $ Diastolica: int  64 69 57 65 67 69 63 66 65 58 ...
 $ Frequenza : int  50 49 62 56 60 59 53 49 48 63 ...
> #
```

Questo accade perché nel file `PA.csv` che aperto con un editor di testo così ci appare

Data,Sistolica,Diastolica,Frequenza
07/05/2021,112,64,50
08/05/2021,118,69,49
08/05/2021,101,57,62
09/05/2021,111,65,56
09/05/2021,117,67,60
10/05/2021,110,69,59
10/05/2021,109,63,53
16/05/2021,115,66,49
17/09/2021,121,65,48
17/09/2021,92,58,63
17/09/2021,131,79,68
17/09/2021,116,68,57
29/09/2021,114,69,56

non è riportata alcuna informazione che consenta ad **R** di identificare la variabile `Data` come una data del calendario (nel file `.xls` questa informazione c'è, anche se non potete visualizzarla trattandosi di un file binario).

Nel caso di dati importati da un file `.csv` - che di fatto è un semplice file di testo - è pertanto necessario impiegare la funzione **as.Date()** per convertire le date/stringhe in ingresso in date del calendario:

```
> PA_csv$Data <- as.Date(PA_csv$Data, format="%d/%m/%Y") # la stringa Data viene convertita con as.Date
> str(PA_csv) # la variabile Data viene ora riconosciuta come una data del calendario
'data.frame':   13 obs. of  4 variables:
 $ Data      : Date, format: "2021-05-07" "2021-05-08" ...
 $ Sistolica : int  112 118 101 111 117 110 109 115 121 92 ...
 $ Diastolica: int  64 69 57 65 67 69 63 66 65 58 ...
 $ Frequenza : int  50 49 62 56 60 59 53 49 48 63 ...
> #
```

Questa conversione è necessaria se vogliamo che **R** riconosca le operazioni che è possibile eseguire su questi oggetti, che saranno, appunto, solamente le operazioni applicabili alle date del calendario.

A questo punto la domanda: ma perché fare tanta fatica per arrivare fin qui? La risposta la trovate a conclusione dello script, laddove sono riportate alcune delle possibilità offerte da **R** nella gestione delle date che prima o poi possono tornare utili, come ad esempio:

→ la data attuale presente sul sistema

```
> Sys.Date()
[1] "2021-11-29"
```

→ la differenza in giorni tra due date

```
> PA_xls$Data[13] - PA_xls$Data[1]
Time difference of 145 days
```

→ la data corrispondente a una specifica differenza in giorni rispetto a una data di riferimento

```
> PA_xls$Data[13] + 32
[1] "2021-10-31"
```

→ il giorno della settimana corrispondente ad una specifica data

```
> weekdays(PA_csv$Data[1])
[1] "venerdì"
> #
```

In un prossimo post vedremo come gestire contemporaneamente data e ora.

---

mercoledì 2 febbraio 2022

Come separare i decimali e come raggruppare le cifre

Nel mondo anglosassone 3.142 è un numero con tre cifre decimali, mentre 302,073 sta per trecentoduemilasettantatre.

In Italia e in genere in Europa il primo, il valore (arrotondato) di pi greco (π), si scrive 3,142 mentre il secondo, la superficie dell'Italia espressa in km^2, si trova sovente riportato, anche in fonti ufficiali, come 302.073 [1].

Assumendo che il lettore, europeo o anglofono, non sappia di avere a che fare con pi-greco e con la superficie dell'Italia, quindi che interpreti il valore numerico *alla luce della sola informazione derivante dal proprio contesto linguistico*, il significato attribuito a questi numeri è il seguente:

	Significato in relazione al contesto linguistico	
	anglosassone	europeo
3,142	Tremila ...	Tre virgola ...
3.142	Tre virgola...	Tremila...
302,073	Trecentoduemila ...	Trecentodue virgola ...
302.073	Trecentodue virgola ...	Trecentoduemila ...

Il significato del punto (.) e della virgola (,) dipende dal contesto linguistico a causa del fatto che sono entrambi usati, a livello internazionale, sia come separatori decimali sia come simboli per il raggruppamento delle cifre. Ma si tratta di una prassi errata.

In realtà la situazione dovrebbe essere diversa, perché nella **XXII Conferenza Generale dei Pesi e delle Misure** del 2003, nella risoluzione 10, la **CGPM** [2] ha stabilito che:
→ "*il simbolo per il marcatore decimale **deve** essere il punto sulla linea [di base] o la virgola sulla linea [di base]*", ammettendo quindi, per ragioni pratiche/linguistiche, la coesistenza di questi due simboli;
→ "*i numeri **possono** essere suddivisi in gruppi di tre per facilitare la lettura; [ma] né punti né virgole devono essere inseriti negli spazi tra i gruppi, come stabilito nella risoluzione 7 della IX CGPM, 1948*".

Le regole dettate dalla CGPM, alla quale partecipa anche l'Italia in quanto aderente alla *Convenzione del Metro* [2], regole che risalgono addirittura al 1948, sono quindi molto semplici:
→ per **separare i decimali** devono essere impiegati o il punto (.) o la virgola (,) liberamente, in modo consono all'uso che si è andato consolidando nel tempo, al contesto linguistico e cuturale, o quant'altro;
→ **raggruppare le cifre** è facoltativo ma, nel caso in cui lo si faccia per rendere più agevole la lettura, può essere impiegato esclusivamente lo spazio, visto che per definizione "... *né punti né virgole devono essere inseriti negli spazi tra i gruppi...*".

Questo cambia tutto, perché se fossero applicate queste regole - che vengono ignorate anche nei Paesi come l'Italia che aderendo alla *Convenzione del Metro* del 20 maggio 1875 hanno successivamente adottato per legge [3] il **Sistema Internazionale di Unità (SI)** [4, 5] - allora il punto (.) e la virgola (,) sarebbero sempre correttamente intesi per quello che sono, cioè entrambi, per definizione, come separatori dei decimali e mai come simboli per il raggruppamento delle cifre, e la situazione diventerebbe questa:

	Significato in relazione al contesto linguistico	
	anglosassone	europeo
3,142 o 3.142	Tre virgola ...	Tre virgola ...
302 073	Trecentoduemila...	Trecentoduemila...
~~302,073~~	~~Trecentoduemila ...~~	~~Trecentodue virgola ...~~
~~302.073~~	~~Trecentodue virgola ...~~	~~Trecentoduemila ...~~

Alla luce di queste regole un anglosassone che leggesse un 3,142 lo interpreterebbe alla stessa stregua di un 3.142 e un europeo che leggesse un 3.142 lo interpreterebbe alla stessa stregua di un 3,142. A fronte di una diversa scelta nell'impiego del separatore dei decimali (ma "*de gustibus non disputandum est*") l'interpretazione sarebbe univoca. E ovviamente nessun dubbio potrebbe mai sorgere nella lettura del 302 073 come trecentoduemilasettantatre.

Per applicare questo approccio qualcuno potrebbe far notare che rimangono comunque da risolvere due problemi legati al comportamento dei programmi di videoscrittura, che:
→ nel caso in cui si impiega la giustificazione del testo, ampliano e riducono automaticamente gli spazi vuoti ai fini della miglior composizione della riga;
→ potrebbero andare a capo, passando alla riga successiva, proprio a livello dello spazio che separa un gruppo di cifre da quello che lo segue.

Ma questi due problemi sono già stati risolti. Oggi qualsiasi programma di videoscrittura consente di impiegare, oltre allo spazio normale, che si inserisce in un testo con la barra spaziatrice, anche il carattere **spazio unificatore**: questo spazio non cambia ampiezza e non consente di andare a capo, lasciando quindi il testo che lo include sempre correttamente impaginato.

Lo **spazio unificatore**:
→ su Windows può essere inserito tenendo premuti contemporaneamente i tasti <*ctrl*> e <*shift*> e premendo la *barra spaziatrice*, oppure tenendo premuto il tasto <*alt*> e digitando *255* sul tastierino numerico;
→ su Mac può essere inserito tenendo premuto il tasto <*alt*> e premendo la *barra spaziatrice*;
→ su Linux può essere inserito tenendo premuti contemporaneamente i tasti <*ctrl*> e <*shift*> e premendo la *barra spaziatrice* o tenendo premuto il tasto <*shift*> e premendo la *barra spaziatrice* o ricorrendo a una apposita configurazione.

Per separare i decimali **R** impiega di default e al proprio interno esclusivamente il punto (.) ma permette di importare e di esportare i dati impiegando come separatore dei decimali sia il punto (.) sia la virgola (,) [6].

Se dovete preparare un documento nel quale i numeri sono riportati utilizzando la notazione scientifica (di default in **R**) o utilizzando i multipli e sottomultipli raccomandati del SI [7], il problema dello spazio unificatore non si pone. Ma se volete esprimere i numeri in formato fisso [8] ricordate che:

→ *per facilitarne la lettura i numeri superiori a 999 possono* [facoltativamente] *essere suddivisi in gruppi di tre cifre* [ma] *impiegando* [esclusivamente] *lo spazio* [unificatore].

Attualmente in Italia i fogli elettronici, in barba alla razionalità e all'ufficialità dell'approccio definito dalla CGPM, impiegano di default il punto per raggruppare le cifre. Gli utilizzatori si adeguano (difficile che qualcuno si prende la briga di andare a riconfigurare le opzioni di formattazione dei numeri) e questo induce a perpetuare la vecchia e concettualmente sbagliata modalità di rappresentazione.

Temo che sarà necessario attendere molto per risolvere il problema, ammesso che qualcuno lo voglia prima o poi affrontare. Ma sarebbe il classico uovo di Colombo:
→ si sostituiscono i due simboli attualmente impiegati per raggruppare le cifre all'interno dei numeri (il punto e la virgola) con uno solo (lo spazio unificatore);
→ si attribuisce lo stesso significato, quello di separatori dei decimali, al punto e alla virgola;
semplificando, eliminando possibili errori di interpretazione e facilitando una più immediata e universale comprensione: che è poi la ragione per cui è stato creato il SI.

---

[1] Istat. *Principali dimensioni geostatistiche e grado di urbanizzazione del Paese. Superficie territoriale per zona altimetrica*. URL consultato il 27/12/2019: http://bit.ly/2Q0BT5K

[2] La *CGPM* (*Conférence Générale des Poids et Mesures*), in pratica il Parlamento dei Paesi che - come l'Italia - aderiscono alla *Convenzione del Metro*, viene convocata periodicamente, ogni quattro o sei anni, e discute e approva unità, terminologia e raccomandazioni del *Sistema internazionale di unità (SI)*.

[3] DECRETO DEL PRESIDENTE DELLA REPUBBLICA 12 agosto 1982, n. 802 *Attuazione della direttiva (CEE) n. 80/181 relativa alle unita' di misura*. GU Serie Generale n.302 del 03-11-1982 - Suppl. Ordinario (http://bit.ly/2v69IqD)

[4] Per una introduzione al SI, che riporta anche le indicazioni delle modifiche apportate al sistema dopo l'adesione ufficiale dell'Italia avvenuta nel 1982, vedere Grandezze e unità di misura. Breve storia dall'antichità al Sistema Internazionale di Unità (SI).

[5] Sul sito del *Bureau International des Poids et Mesures* - in una delle due sole lingue adottate, inglese e francese - si trova il documento ufficiale "*SI Brochure: The International System of Units (SI)*" ovvero "*Brochure sur le SI: Le Système international d'unités*". URL consultato il 10/02/2022: https://bit.ly/3uATFBb

[6] Come illustrato nel post Importazione dei dati da un file .csv e nel post Esportazione dei dati in un file .csv.

[7] Vedi Grandezze e unità di misura. Breve storia dall'antichità al Sistema Internazionale di Unità (SI), p. 47.

[8] Vedere il post Esprimere i numeri in formato fisso.

domenica 3 aprile 2022

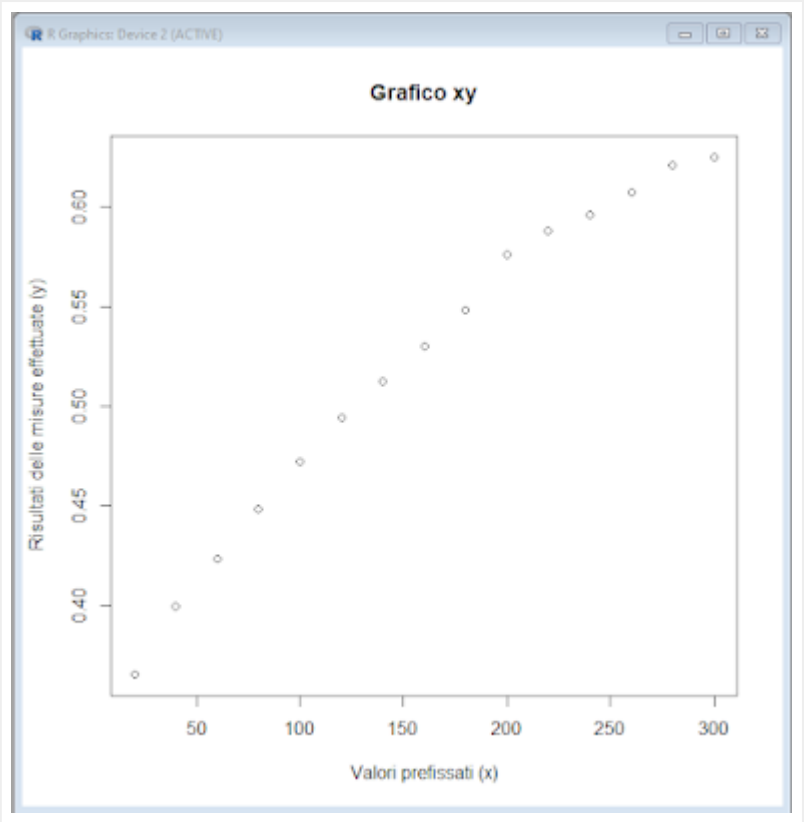
Regressione polinomiale di secondo grado

Supponiamo di avere, per ciascuno di una serie di valori prefissati `x` (riportati in ascisse), il risultato di altrettante misure `y` (riportate in ordinate) e riportiamo con la funzione **plot()** i dati sotto forma di un grafico `xy`.

Copiante il codice che segue nella `Console di R` e premete `↵ Invio`:

```
#
x <- c(20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300) # dati in ascisse
y <- c(0.365, 0.399, 0.423, 0.448, 0.472, 0.494, 0.512, 0.530, 0.548, 0.576, 0.588, 0.596, 0.607, 0.621, 0.625) # dati in ordinate
plot(y~x, col="black", pch=1, main="Grafico xy", xlab="Valori prefissati (x)", ylab="Risultati delle misure effettuate (y)") # traccia il grafico xy
#
```

Dalla ispezione del grafico risulta evidente una relazione non lineare.



Questo ci fa escludere la possibilità di descrivere la relazione di funzione che intercorre tra le due variabili mediante una retta. E pone il tema di quale sia la curva che potrebbe meglio descriverla.

La prima cosa da valutare è se esiste un modello deterministico di un evento fisico, chimico-fisico, biologico, econometrico o quant'altro, che si assume descriva adeguatamente la relazione di funzione che intercorre tra le due variabili: in tal caso non resta che applicare l'equazione fornita dal modello. In caso contrario è necessario ricorrere ad un modello empirico: ed è questo che consideriamo essere il caso nostro.

Qualora si debba ricorrere ad un modello empirico, la regola fondamentale è impiegare l'equazione più semplice compatibile con una adeguata descrizione dei dati. Nel nostro caso visto che è evidente

dall'ispezione del grafico che l'equazione di una retta

$$y = a + b \cdot x$$

calcolata mediante la classica regressione lineare con il metodo dei minimi quadrati sarebbe inadeguata a descrivere la relazione di funzione che lega la `y` alla `x`, possiamo pensare di ricorrere ad una equazione appena più complessa, come quella fornita da una **regressione polinomiale di secondo grado** nella forma

$$y = a + b \cdot x + c \cdot x^2$$

e calcolata anch'essa con il metodo dei minimi quadrati [1].

Il razionale di questa scelta è basato sul fatto che i nostri dati descrivono una curva con la concavità rivolta in basso e un raggio di curvatura omogeneo, senza massimi, senza minimi e senza punti di flesso - cioè senza punti di passaggio tra due curvature che vanno in senso opposto - tutti elementi compatibili con il ramo ascendente di una parabola descritta appunto da un polinomio di secondo grado.

Per capire se questa scelta genera una curva che approssima in modo adeguato i nostri dati, predisponiamo un breve script per calcolare e rappresentare una regressione polinomiale di secondo grado.

Copiate lo script nella `Console di R` e premete `↲ Invio`:

```
# REGRESSIONE POLINOMIALE di secondo grado
#
x <- c(20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300) # dati in ascisse
y <- c(0.365, 0.399, 0.423, 0.448, 0.472, 0.494, 0.512, 0.530, 0.548, 0.576, 0.588, 0.596, 0.607, 0.621, 0.625) # dati in ordinate
#
options(scipen=999) # esprime i numeri in formato fisso
#
# calcola la regressione
#
polisec <- lm(y~poly(x, degree=2, raw=TRUE)) # calcola la regressione
summary(polisec) # riporta le statistiche della regressione
#
# traccia il grafico
#
newx <- seq(min(x), max(x), (max(x)-min(x))/1000) # valori x in corrispondenza dei quali calcolare il valore del polinomio
newy <- predict(polisec, list(x=newx)) # calcola il valore corrispondente del polinomio
plot(y~x, col="black", pch=1, main="Regressione polinomiale di secondo grado", xlab="Valori prefissati (x)", ylab="Risultati delle misure effettuate (y)") # predispone il grafico e riporta i dati
lines(newx, newy, col="black", lty=2, lwd=1) # traccia il polinomio
#
options(scipen=0) # ripristina la notazione scientifica
#
```

Le prime due righe di codice servono semplicemente ad assegnare (**<-**) i valori inclusi nella funzione **c()** al vettore **x** e al vettore **y** che in questo modo conterranno i nostri dati.

Poi mediante la funzione **options()** e con l'argomento **scipen=999** facciamo si che i risultati numerici siano espressi in formato fisso, cosa che li renderà meglio leggibili (questo almeno a mio modo di vedere: se volete lasciare i risultati nella notazione scientifica eliminate questa riga di codice o, forse meglio, anteponete a questa riga di codice il simbolo #).

La quarta riga di codice assegna (**<-**) all'oggetto **polisec** mediante la funzione **lm()** i risultati del calcolo della regressione polinomiale **y~poly(x, …)** di secondo grado (**degree=2**). L'argomento **raw=TRUE** è impiegato perché non ci interessano i polinomi ortogonali (l'opzione di default è **raw=FALSE**).

Se siete interessati ad approfondimenti, ricordo che potete visualizzare ad esempio la struttura dell'oggetto **polisec** digitando **str(polisec)** e potete avere informazioni sulle funzioni impiegate nello script digitando **help**(*nomedellafunzione*).

Alla quinta riga di codice la funzione **summary(polisec)** consente di visualizzare le statistiche della regressione polinomiale:

```
Call:
lm(formula = y ~ poly(x, degree = 2, raw = TRUE))
Residuals:
       Min         1Q     Median         3Q        Max
-0.0043421 -0.0023941 -0.0000973  0.0013490  0.0074455
Coefficients:
                                    Estimate    Std. Error t value             Pr(>|t|)
(Intercept)                       0.3371692308  0.0031879149  105.77 < 0.0000000000000002 ***
poly(x, degree = 2, raw = TRUE)1  0.0015339463  0.0000458427   33.46    0.000000000000322 ***
poly(x, degree = 2, raw = TRUE)2 -0.0000018851  0.0000001393  -13.53    0.000000012542671 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.003579 on 12 degrees of freedom
Multiple R-squared:  0.9985,    Adjusted R-squared:  0.9982
F-statistic:  3879 on 2 and 12 DF,  p-value: < 0.00000000000000022
```

La voce `Residuals` riporta le statistiche delle differenze che, dopo l'approssimazione dei punti con il polinomio di secondo grado, residuano tra i punti stessi e la curva calcolata: differenze piccole come quelle qui osservate indicano un'ottima approssimazione della curva ai dati.

La voce `Coefficients` riporta dall'alto verso il basso il valore stimato (`Estimate`) dell'intercetta, del coefficiente di primo grado (`x`) e del coefficiente di secondo grado (x^2) e ci consente di scrivere l'equazione del polinomio che sarà quindi

$$y = 0.3371692308 + 0.0015339463 \cdot x - 0.0000018851 \cdot x^2$$

Per ciascun coefficiente viene calcolato il valore del *test t di Student* (`t value`) come rapporto tra il valore stimato (`Estimate`) e il suo errore standard (`Std. Error`) e viene infine riportato il valore `p` di probabilità di osservare per caso il valore di t riportato: se tale probabilità è sufficientemente bassa ci assumiamo il rischio di considerare significativo il coefficiente in questione. Nel nostro caso essendo i valori di `p` sempre molto piccoli (di gran lunga inferiori al valore `p = 0.05` generalmente considerato come valore soglia per la significatività) possiamo affermare che tutti e tre i coefficienti del polinomio di secondo grado sono significativi e lo rendono un'equazione in grado di approssimare adeguatamente i dati forniti.

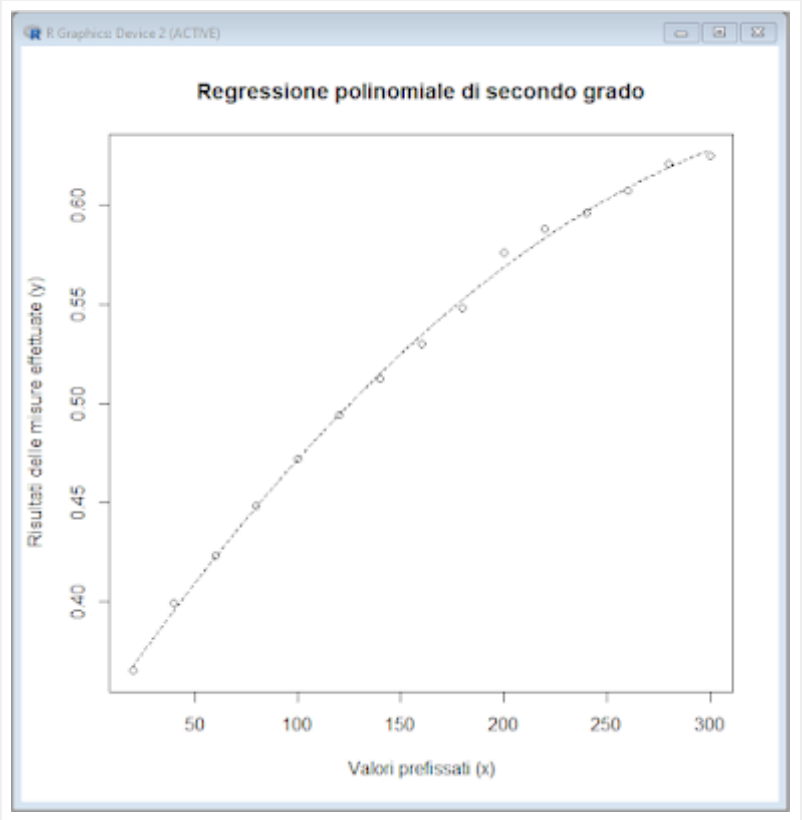
Il valore del *coefficiente di correlazione multipla* R^2 molto elevato (`Multiple R-squared: 0.9985`) da una misura della bontà dell'approssimazione, confermata dalla sua significatività con il *test F* (`p-value: < 0.00000000000000022`).

Infine in quattro passaggi viene tracciato il grafico che riporta i punti e traccia il polinomio che li approssima:

→ l'intervallo compreso tra il valore minimo **min(x)** della `x` e il valore massimo **max(x)** della `x` viene suddiviso per ottenere i valori `x` (salvati nell'oggetto **newx**) in corrispondenza dei quali calcolare il valore del polinomio;
→ i valori `y` del polinomio corrispondenti alle `x` contenute in **newx** sono calcolati e riportati nell'oggetto **newy**;
→ la funzione **plot()** viene impiegata per riportare sul grafico di dati contenuti in **x** e in **y**;
→ la funzione **lines()** viene impiegata per sovrapporre ai dati il tratto di polinomio calcolato e salvato in **newx** e in **newy**.

Potete riportare i dati modificando il simbolo impiegato (**pch=...**) e il suo colore (**col="..."**), come pure tracciare il polinomio modificando l'aspetto o stile della linea tracciata (**lty=...**), il suo colore (**col="..."**) e il suo spessore (**lwd=...**) [2].

Il grafico - complemento ai test statistici da cui non è possibile prescindere - conferma che la regressione polinomiale di secondo grado fornisce una eccellente approssimazione ai dati.



Da notare che il grafico è stato tracciato esclusivamente all'interno dei dati, quindi in un'ottica di interpolazione: e questo ci ricorda anche che, trattandosi di un modello empirico, è da escludere l'impiego del polinomio di secondo grado per la estrapolazione, cioè per trarre conclusioni al di fuori dei dati impiegati per calcolarlo.

A chiudere lo script è la funzione **options()** che con l'argomento **scipen=0** ripristina l'espressione dei numeri nella notazione scientifica [3].

Lo script può essere riutilizzato semplicemente riportando manualmente in **x** e in **y** i nuovi dati, o meglio ancora importandoli da un file [4].

Se siete interessati al tema trovate in un altro post come - in assenza di un'equazione basata su un modello teorico - situazioni più complesse, ma che si possono riscontrare nella pratica, possono essere affrontate impiegando per approssimare i punti la regressione polinomiale di terzo grado.

---

[1] Il **metodo dei minimi quadrati** prevede - ed è quello che abbiamo assunto essere il nostro caso - che la variabile in ascisse (`x`) abbia *valori prefissati*, che la variabile in ordinate (`y`) sia rappresentata da *misure* (affette da un errore casuale distribuito normalmente) effettuate corrispondenza delle `x` e *minimizza* la somma dei quadrati delle differenze `y - y'` ovvero delle differenze tra ciascun valore misurato `y` e il valore `y' = f(x)` corrispondente calcolato mediante la regressione (lineare o polinomiale). Esempi possono essere la risposta `y` di un sensore misurata in corrispondenza di una serie di concentrazioni date `x` di una sostanza chimica, la misurazione di un evento `y` in corrispondenza di una serie data di tempi `x`, e così via.

[2] Trovate le relative indicazioni:
→ nel post Cambiare i simboli dei punti di R
→ nel post Visualizzare i colori disponibili in R
→ nel post Cambiare gli stili delle linee di R.

[3] La *notazione scientifica* è una *notazione esponenziale*, cioè una notazione che consente di scrivere un numero `N` assegnato come prodotto di un opportuno numero `a` per la potenza di un altro numero b^k essendo `b` la base della notazione esponenziale. In particolare la notazione scientifica è una notazione esponenziale in base 10 (`b = 10`). Tuttavia "*notazione esponenziale*" non è sinonimo di "*notazione scientifica*" in quanto esistono notazioni esponenziali che impiegano una base diversa da quella impiegata dalla notazione scientifica: ne sono un esempio la notazione esponenziale in base 2 (`b = 2`) impiegata in informatica, e la notazione esponenziale in base `e` (`b = e`) - essendo il numero di Nepero `e` = 2.718 281 828 459... - impiegata in campo matematico e in campo scientifico.
[Precisazione aggiuntiva: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata non solo negli script ma anche nei file di dati e in tutto il testo di questo sito, quindi anche nell'espressione del numero `e` di Nepero. Per le norme che regolano la punteggiatura all'interno dei numeri rimando al post Come separare i decimali e come raggruppare le cifre].

[4] Alla pagina Indice nel capitolo **R - gestione dei dati** trovate i collegamenti ai post che contengono gli script per importare i dati da file esterni.

mercoledì 20 aprile 2022

Ricercare una possibile correlazione (tra più variabili)

La ricerca di una possibile correlazione, intesa come la tendenza di grandezze a variare congiuntamente in modo statisticamente significativo, richiede – oltre alla necessità ineludibile di spiegare per quale ragione l'una dipenda dall'altra – un approccio integrato che include il calcolo del coefficiente di correlazione, la regressione lineare e l'esecuzione dei test di normalità. Lo vediamo ora con tre script da tenere a portata di mano per un impiego immediato al bisogno.

Gli script consentono per la parte statistica di calcolare, nel caso del confronto tra più di due variabili:
→ il **coefficiente di correlazione lineare r di Pearson** (test parametrico)
→ il **coefficiente di correlazione per ranghi ρ (rho) di Spearman** (test non parametrico)
→ il **coefficiente di correlazione τ (tau) di Kendall** (test non parametrico)

Quale dei tre coefficienti impiegare per trarre dai propri dati le conclusioni in merito alla significatività della correlazione deve essere valutato di volta in volta alla luce delle seguenti considerazioni:
1) come suggerisce anche la sua denominazione completa e corretta, il coefficiente di correlazione lineare **r** di Pearson fornisce una misura accurata del grado di correlazione quando sono soddisfatti i seguenti due requisiti: (i) i dati hanno una distribuzione gaussiana [1] in quanto si tratta di un test parametrico; (ii) tra le due variabili a confronto esiste una relazione lineare;
2) i due coefficienti di correlazione non parametrici (**ρ** e **τ**) forniscono una misura della correlazione valida anche quando i dati non sono distribuiti in modo gaussiano e tra le due variabili a confronto non esiste una relazione lineare;
3) i test non parametrici sono più conservativi cioè tendono a fornire valori di `p` superiori (quindi valori di `p` meno significativi) rispetto al test parametrico; se si preferisce essere più prudenti nelle conclusioni, si possono applicare i test non parametrici anche quando i dati soddisfano i requisiti di gaussianità e di linearità che consentirebbero l'applicazione del test parametrico. Il coefficiente di correlazione **τ** di Kendall è il più conservativo.

Per eseguire gli script è necessario disporre dei pacchetti **car**, **corrgram**, **gclus**, **Hmisc**, **moments** [2], se non li avete dovete scaricarli e installarli dal **CRAN**. Installate infine il pacchetto **DAAG** che contiene nella tabella **ais** i dati impiegati come esempio.

Gli script prevedono tutti la stessa sequenza logica che include:
a) il calcolo dello specifico *coefficiente di correlazione* per tutte le coppie di variabili;
b) il calcolo della *significatività* dei coefficienti di correlazione;
c) i grafici xy con la *retta di regressione* per una valutazione grafica della linearità dei dati messi a confronto;
d) la rappresentazione dei dati sotto forma di *correlogrammi*, per meglio evidenziare le correlazioni;
e) i test per la valutazione della *normalità* (gaussianità) dei dati.

Vediamo per primo lo script per il calcolo del **coefficiente di correlazione lineare r di Pearson**, copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# COEFFICIENTE DI CORRELAZIONE LINEARE r (erre) di Pearson (parametrico)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
mydata <- ais[c(1:11)] # salva in mydata le colonne contenenti variabili numeriche
#
# coefficiente di correlazione lineare r di Pearson con valori p di significatività
#
library(Hmisc) # carica il pacchetto
rcorr(as.matrix(mydata), type="pearson") # mostra i valori di r di Pearson e i valori di p corrispondenti
#
# grafico xy per tutte le coppie di variabili
#
library(car) # carica il pacchetto
scatterplotMatrix(~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt,
regLine=list(method=lm, lty=1, lwd=2, col="red"), smooth=FALSE,
diagonal=list(method="density", bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE),
col="black", main="Grafici di dispersione", data=mydata) # traccia i grafici di dispersione xy che incrociano tutte le variabili
#
# correlogrammi
#
library(corrgram) # carica il pacchetto
windows() # apre una nuova finestra
corrgram(mydata, cor.method = "pearson", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.pie, text.panel=panel.txt, main="Correlogramma (r di Pearson)") # correlogramma con grafici xy e grafici a torta
#
windows() # apre una nuova finestra
corrgram(mydata, cor.method = "pearson", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.conf, text.panel=panel.txt, main="Correlogramma (r di Pearson)") # correlogramma con grafici xy e r di Pearson
#
windows() # apre una nuova finestra
corrgram(mydata, cor.method="pearson", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.ellipse, text.panel=panel.txt, diag.panel=panel.density,
main="Correlogramma (r di Pearson)") # correlogramma con grafici xy ed ellissi delle tendenze con curva loess
#
library(gclus) # carica il pacchetto
windows() # apre una nuova finestra
prs <- cor(mydata, method="pearson") # r di Pearson
mydata.col <- dmat.color(abs(prs)) # applica i colori
mydata.o <- order.single(abs(prs)) # dispone le variabili meglio correlate più vicine alla diagonale
cpairs(mydata, mydata.o, panel.colors=mydata.col, gap=.5, main="Gradiente di correlazione (r di Pearson)") # mostra il grafico
#
# test di gaussianità (asimmetria e curtosi)
#
library(moments) # carica il pacchetto
print(paste("numero dei dati =", n <- nrow(mydata)), quote=FALSE)
print(paste("deviazione standard del coefficiente di asimmetria (sa) =", sa <- sqrt(6/n)), quote=FALSE)
print(paste("deviazione standard del coefficiente di curtosi(sc) =", sc <- sqrt(24/n)), quote=FALSE)
```

```
#
ca <- skewness(mydata) # coefficiente di asimmetria (ca)
ca_sa <- round(abs(ca/sa), digits=2) # rapporto (ca/sa)
#
cc <- kurtosis(mydata)-3 # coefficiente di curtosi (cc)
cc_sc <- round(abs(cc/sc), digits=2) # rapporto (cc/sc)
#
ascurt <- data.frame(ca, ca_sa, cc, cc_sc) # costruisce la tabella
names(ascurt) <- c("ca", "ca/sa", "cc", "cc/sc") # assegna i nomi alle colonne
ascurt # mostra la tabella
#
```

Nel primo blocco di codice viene innanzitutto caricato con **library(DAAG)** il pacchetto che contiene nella tabella **ais** i dati utilizzati come esempio. Quindi all'oggetto **mydata** sono assegnate (**<-**) le variabili numeriche della tabella **ais** che si vogliono analizzare indicando le colonne che le contengono. Da notare che qui è possibile impiegare la forma **ais[c(1:11)]** in quanto le colonne sono contigue: in caso contrario è necessario specificare le colonne una per una nella forma **ais[c(1,2,3,4,5,6,8,9,10,11)]** nella quale, a titolo di esempio, è stata esclusa la colonna **7**.

Per importare i vostri dati è sufficiente sostituire **ais[c(1:11)]** con il nome della vostra tabella e **rcc**, **wcc**, **hc**, **hg** e così via con i nomi delle variabili contenute nella tabella.

Nel secondo blocco di codice, dopo avere caricato il pacchetto **Hmisc**, viene eseguita l'analisi statistica sull'oggetto **mydata** impiegando la funzione **rcorr()** e specificando l'argomento **type="pearson"**: in questo modo sono calcolati e riportati in forma di tabella i risultati per il coefficiente di correlazione lineare **r** di Pearson e i relativi valori di probabilità `p`.

Questi sono i valori di **r** calcolati

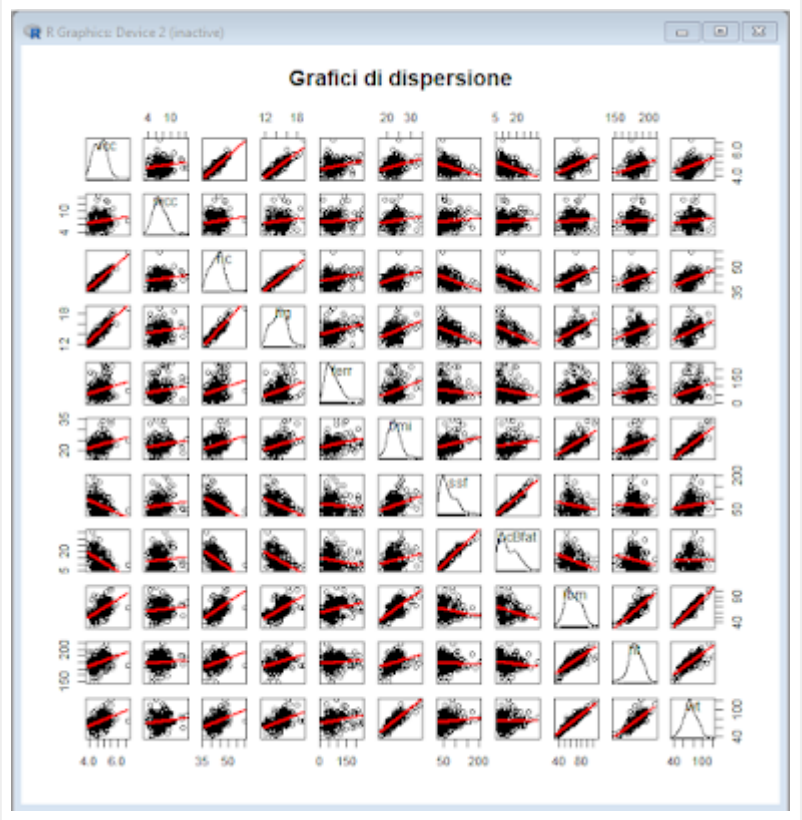
```
          rcc  wcc    hc    hg  ferr  bmi   ssf pcBfat   lbm    ht   wt
rcc      1.00 0.15  0.92  0.89  0.25 0.30 -0.40  -0.49  0.55  0.36 0.40
wcc      0.15 1.00  0.15  0.13  0.13 0.18  0.14   0.11  0.10  0.08 0.16
hc       0.92 0.15  1.00  0.95  0.26 0.32 -0.45  -0.53  0.58  0.37 0.42
hg       0.89 0.13  0.95  1.00  0.31 0.38 -0.44  -0.53  0.61  0.35 0.46
ferr     0.25 0.13  0.26  0.31  1.00 0.30 -0.11  -0.18  0.32  0.12 0.27
bmi      0.30 0.18  0.32  0.38  0.30 1.00  0.32   0.19  0.71  0.34 0.85
ssf     -0.40 0.14 -0.45 -0.44 -0.11 0.32  1.00   0.96 -0.21 -0.07 0.15
pcBfat -0.49 0.11 -0.53 -0.53 -0.18 0.19  0.96   1.00 -0.36 -0.19 0.00
lbm      0.55 0.10  0.58  0.61  0.32 0.71 -0.21  -0.36  1.00  0.80 0.93
ht       0.36 0.08  0.37  0.35  0.12 0.34 -0.07  -0.19  0.80  1.00 0.78
wt       0.40 0.16  0.42  0.46  0.27 0.85  0.15   0.00  0.93  0.78 1.00

n= 202
```

e questi i rispettivi valori di `p` tabulati (i valori `0.0000` sono da intendere come `p < 0.0001`) che consentono di stabilirne la significatività. Potete impiegare come valore soglia il classico `p = 0.05` ovvero essere un poco più esigenti (e prudenti) e porre la soglia per la significatività di **r** in corrispondenza del valore `p = 0.01`.

```
P
       rcc    wcc    hc     hg     ferr   bmi    ssf    pcBfat lbm    ht     wt
rcc           0.0367 0.0000 0.0000 0.0003 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
wcc    0.0367        0.0294 0.0559 0.0610 0.0118 0.0519 0.1262 0.1460 0.2773 0.0270
hc     0.0000 0.0294        0.0000 0.0002 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
hg     0.0000 0.0559 0.0000        0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
ferr   0.0003 0.0610 0.0002 0.0000        0.0000 0.1252 0.0090 0.0000 0.0805 0.0000
```

```
bmi     0.0000 0.0118 0.0000 0.0000 0.0000        0.0000 0.0075 0.0000 0.0000 0.0000
ssf     0.0000 0.0519 0.0000 0.0000 0.1252 0.0000        0.0000 0.0030 0.3136 0.0284
pcBfat 0.0000 0.1262 0.0000 0.0000 0.0090 0.0075 0.0000        0.0000 0.0074 0.9978
lbm     0.0000 0.1460 0.0000 0.0000 0.0000 0.0000 0.0030 0.0000        0.0000 0.0000
ht      0.0000 0.2773 0.0000 0.0000 0.0805 0.0000 0.3136 0.0074 0.0000        0.0000
wt      0.0000 0.0270 0.0000 0.0000 0.0000 0.0000 0.0284 0.9978 0.0000 0.0000
```

Con il terzo blocco di codice, dopo avere caricato il pacchetto **car**, con la funzione **scatterplotMatrix()** sono riportati in un'unica figura i grafici xy (grafici di dispersione) che incrociano tutte le variabili: questo consente una valutazione grafica preliminare dei dati, delle loro distribuzioni e delle loro possibili relazioni.

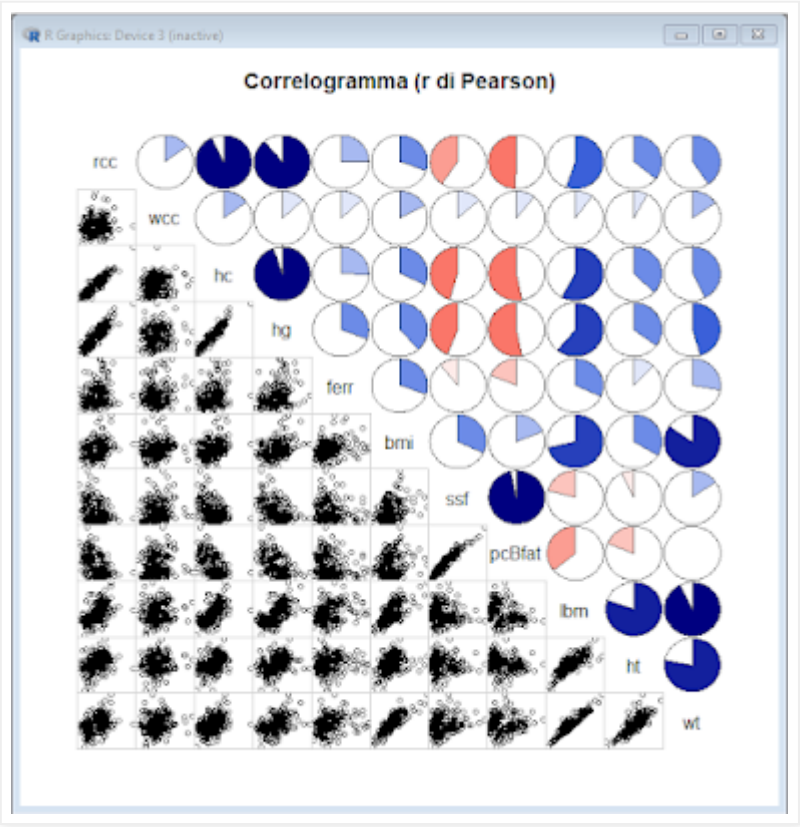


Sulla diagonale sono riportati i kernel density plot che consentono di valutare la distribuzione dei dati. In ciascun grafico xy è riportata a scopo orientativo la retta di regressione, la cui significatività peraltro deve essere valutata con gli altri strumenti che trovate più avanti.

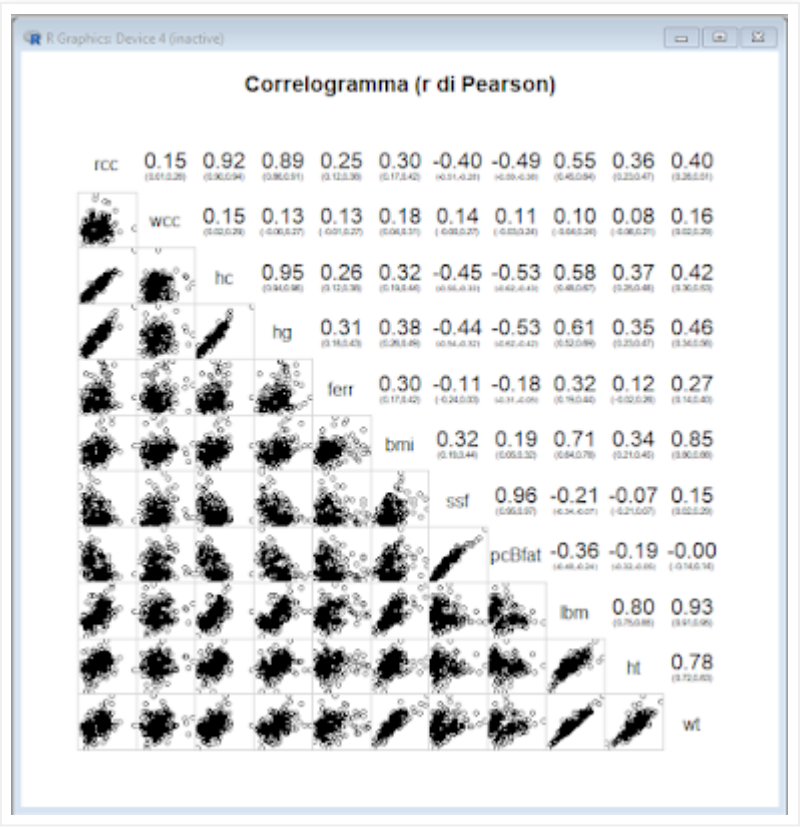
L'argomento **~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt** qui specifica tutte le variabili del set di dati **ais** allo scopo di non perdere nessuna potenziale correlazione. Se avete importato i dati da un vostro file non dimenticate di modificare questo argomento, riportando dopo la tilde (**~**) l'elenco delle variabili che volete rappresentare e collegando l'una all'altra con il segno **+**.

Nel quarto blocco di codice, sono realizzati innanzitutto mediante le funzioni del pacchetto **corrgram** tre correlogrammi che riportano come riferimento in basso a sinistra sotto alla diagonale i grafici xy dei dati (**lower.panel=panel.pts**).

Il primo correlogramma riporta in alto a destra i grafici a torta (**upper.panel=panel.pie**) che rappresentano i valori di **r** (**cor.method="pearson"**) con superficie colorata della torta e intensità del colore proporzionali al valore di **r** e con le gradazioni di blu e di rosso riferite rispettivamente a valori positivi e a valori negativi di **r**.

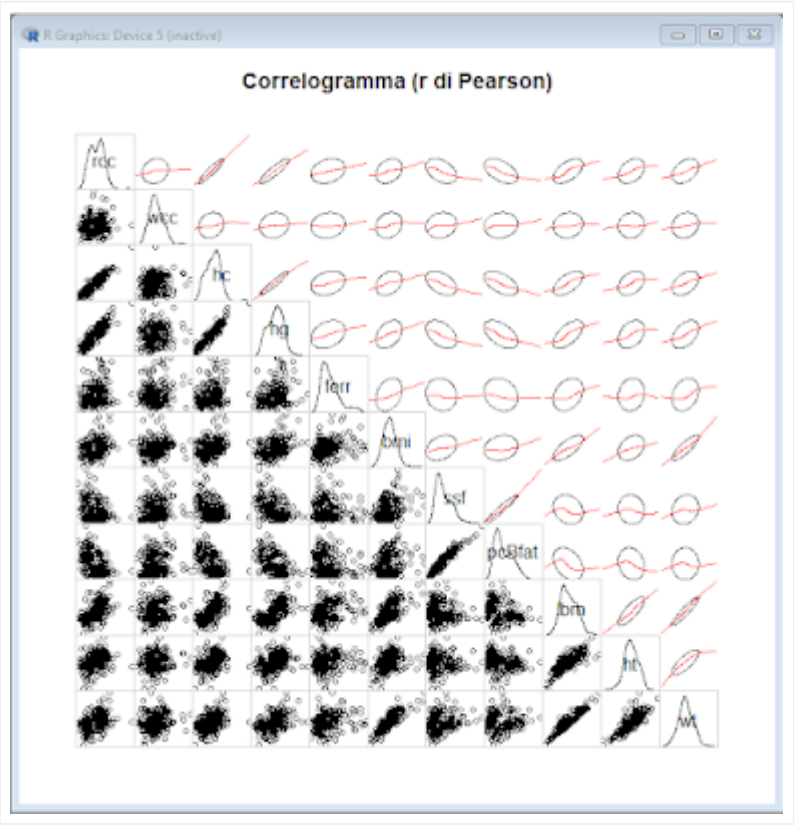


Il secondo correlogramma riporta in alto a destra i valori del coefficiente di correlazione lineare **r** di Pearson con i limiti di confidenza al 95% (**upper.panel=panel.conf**). Questi forniscono un test di significatività: il valore di **r** è significativo se i limiti di confidenza non includono lo `0` (zero), mentre non è significativo se includono lo `0`.

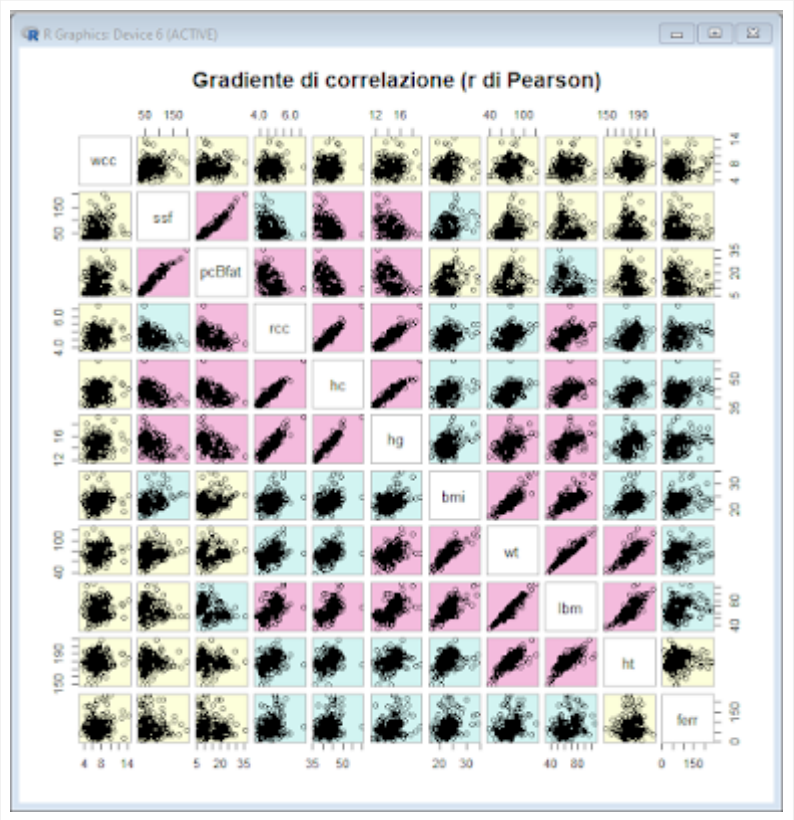


Il terzo correlogramma riporta in alto a destra le ellissi (**upper.panel=panel.ellipse**) i cui assi maggiori delineano la tendenza delle due variabili a variare congiuntamente, con le regressioni loess che aiutano a valutare la linearità della loro relazione. Inoltre riporta sulla diagonale i kernel density plot delle variabili rappresentate (**diag.panel=panel.density**) già visti nel primo grafico, che ci

consentono di evidenziare variabili distribuite in modo spiccatamente asimmetrico, come la variabile `ferr`, o che hanno una evidente distribuzione bimodale, come la variabile `rcc`: fatti dei quali si deve tenere conto.



Un quarto e ultimo correlogramma viene quindi realizzato mediante le funzioni del pacchetto **gclus**. Innanzitutto con la funzione **corr()** e con l'argomento **method="pearson"** sono calcolati e salvati (**<-**) in **prs** i valori del coefficiente di regressione lineare **r** di Pearson. Quindi sono predisposti i colori (**mydata.cor**) per evidenziare le variabili meglio correlate e viene stabilito l'ordine della variabili (**mydata.o**) in modo che le meglio correlate siano più vicine alla diagonale. Ordine e colore delle variabili sono quindi impiegati nella funzione **cpairs()** per organizzare i grafici all'interno del correlogramma, con i valori di **r** più significativi le relazioni lineari più evidenti disposti attorno alla diagonale.



Ai correlogrammi seguono i test di normalità (gaussianità). Viene mostrato il numero dei dati

```
> print(paste("numero dei dati =", n <- nrow(mydata)), quote=FALSE)
[1] numero dei dati = 202
```

viene calcolata la deviazione standard **sa** del coefficiente di asimmetria

```
> print(paste("deviazione standard del coefficiente di asimmetria (sa) =", sa <- sqrt(6/n)), quote=FALSE)
[1] deviazione standard del coefficiente di asimmetria (sa) = 0.172345496886428
```

e viene calcolata la deviazione standard **sc** del coefficiente di curtosi

```
> print(paste("deviazione standard del coefficiente di curtosi(sc) =", sc <- sqrt(24/n)), quote=FALSE)
[1] deviazione standard del coefficiente di curtosi(sc) = 0.344690993772856
```

Per ognuna della variabili con la funzione **skewness()** viene calcolato il coefficiente di asimmetria **ca** dal quale si ricava il rapporto **ca/sa**, sempre per ognuna delle variabili con la funzione **kurtosis()** viene calcolato il coefficiente di curtosi **cc** dal quale si ricava il rapporto **cc/sc**. Infine i valori calcolati sono assemblati mediante la funzione **data.frame()** in una tabella:

	ca	ca/sa	cc	cc/sc
rcc	0.4160046	2.41	0.662946166	1.92
wcc	0.8352336	4.85	1.449462762	4.21
hc	0.2752244	1.60	0.891229892	2.59
hg	0.1759351	1.02	0.008064219	0.02
ferr	1.2805839	7.43	1.420173468	4.12
bmi	0.9465155	5.49	2.183474974	6.33
ssf	1.1746685	6.82	1.365135223	3.96
pcBfat	0.7595480	4.41	-0.173299598	0.50
lbm	0.3585088	2.08	-0.240119844	0.70
ht	-0.1993028	1.16	0.528116566	1.53

```
wt       0.2406053  1.40  0.385610737  1.12
```

Se il rapporto **ca/sa** è superiore a `2.6` la asimmetria è significativa. Se il rapporto **cc/sc** è superiore a `2.6` la curtosi è significativa. Questo modo semplificato di effettuare il test di significatività è stato adottato per alleggerire il codice dello script, ma nulla impedisce di approfondire al bisogno lo studio delle variabili cui si è maggiormente interessati con altri test che trovate nei post.

Non ho consigli pratici in merito ai criteri da impiegare per la scelta tra l'uno o l'altro dei due test non parametrici, a parte il fatto che rispetto al **ρ** di Spearman il coefficiente di correlazione **τ** di Kendall è più robusto (meno influenzato da valori estremi) ed è più conservativo (a parità di condizioni fornisce valori di `p` maggiori quindi significatività inferiori).

Lo script per l'analisi della correlazione basata sul **coefficiente di correlazione per ranghi ρ (rho) di Spearman** ricalca il precedente, copiatelo e incollatelo nella `Console di R` e premete `↵ Invio`:

```
# COEFFICIENTE DI CORRELAZIONE PER RANGHI ρ (rho) di Spearman (non parametrico)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
mydata <- ais[c(1:11)] # salva in mydata solamente le colonne contenenti variabili numeriche
#
# coefficiente di correlazione per ranghi ρ di Spearman con valori p di significatività
#
library(Hmisc) # carica il pacchetto
rcorr(as.matrix(mydata), type="spearman") # mostra i valori di ρ di Sperman e i valori di p corrispondenti
#
# grafico xy per tutte le coppie di variabili
#
library(car) # carica il pacchetto
scatterplotMatrix(~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt,
regLine=list(method=lm, lty=1, lwd=2, col="red"), smooth=FALSE,
diagonal=list(method="density", bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE),
col="black", main="Grafici di dispersione", data=mydata) # traccia i grafici di dispersione xy che incrociano tutte le variabili
#
# correlogrammi
#
library(corrgram) # carica il pacchetto
windows() # apre una nuova finestra
corrgram(mydata, cor.method="spearman", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.pie, text.panel=panel.txt, main="Correlogramma (ρ di Spearman)") # correlogramma con grafici xy e grafici a torta
#
windows() # apre una nuova finestra
corrgram(mydata, cor.method="spearman", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.cor, text.panel=panel.txt, main="Correlogramma (ρ di Spearman)") # correlogramma con grafici xy e ρ di Spearman
#
windows() # apre una nuova finestra
corrgram(mydata, cor.method="spearman", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.ellipse, text.panel=panel.txt,
diag.panel=panel.density, main="Correlogramma (ρ di Spearman)") # correlogramma con grafici xy ed ellissi delle tendenze con curva loess
#
library(gclus) # carica il pacchetto
windows() # apre una nuova finestra
rho <- cor(mydata, method="spearman") # ρ di Spearman
mydata.col <- dmat.color(abs(rho)) # applica i colori
mydata.o <- order.single(abs(rho)) # dispone le variabili meglio correlate più vicine alla diagonale
cpairs(mydata, mydata.o, panel.colors=mydata.col, gap=.5, main="Gradiente di correlazione (ρ di Spearman)") # mostra il grafico
#
# test di gaussianità (asimmetria e curtosi)
#
library(moments) # carica il pacchetto
print(paste("numero dei dati =", n <- nrow(mydata)), quote=FALSE)
print(paste("deviazione standard del coefficiente di asimmetria (sa) =", sa <- sqrt(6/n)), quote=FALSE)
print(paste("deviazione standard del coefficiente di curtosi(sc) =", sc <- sqrt(24/n)), quote=FALSE)
```

```
#
ca <- skewness(mydata) # coefficiente di asimmetria (ca)
ca_sa <- round(abs(ca/sa), digits=2) # rapporto (ca/sa)
#
cc <- kurtosis(mydata)-3 # coefficiente di curtosi (cc)
cc_sc <- round(abs(cc/sc), digits=2) # rapporto (cc/sc)
#
ascurt <- data.frame(ca, ca_sa, cc, cc_sc) # costruisce la tabella
names(ascurt) <- c("ca", "ca/sa", "cc", "cc/sc") # assegna i nomi alle colonne
ascurt # mostra la tabella
#
```

Da notare che l'argomento **upper.panel=panel.conf** che riporta nel correlogramma i limiti di confidenza del coefficiente di correlazione può essere impiegato solamente con il test **r** di Pearson ed è stato qui sostituito con l'argomento **upper.panel=panel.cor**

L'analisi della correlazione per dati multivariati si conclude con lo script per l'analisi della correlazione basata sul **coefficiente di correlazione τ (tau) di Kendall**. Copiatelo e incollatelo nella `Console di R` e premete `↲ Invio`:

```
# COEFFICIENTE DI CORRELAZIONE PER RANGHI τ (tau) di Kendall (non parametrico)
#
library(DAAG) # carica il pacchetto che include il set di dati ais
mydata <- ais[c(1:11)] # salva in mydata solamente le colonne contenenti variabili numeriche
#
# coefficiente di correlazione τ di Kendall
#
cor(mydata, method="kendall") # mostra i valori di τ di Kendall
#
# grafico xy per tutte le coppie di variabili
#
library(car) # carica il pacchetto
scatterplotMatrix(~rcc+wcc+hc+hg+ferr+bmi+ssf+pcBfat+lbm+ht+wt,
regLine=list(method=lm, lty=1, lwd=2, col="red"), smooth=FALSE,
diagonal=list(method="density", bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE),
col="black", main="Grafici di dispersione", data=mydata) # traccia i grafici di dispersione xy
che incrociano tutte le variabili
#
# correlogrammi
#
library(corrgram) # carica il pacchetto
windows() # apre una nuova finestra
corrgram(mydata, cor.method="kendall", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.pie, text.panel=panel.txt, main="Correlogramma (τ di Kendall)") #
correlogramma con grafici xy e grafici a torta
#
windows() # apre una nuova finestra
corrgram(mydata, cor.method="kendall", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.cor, text.panel=panel.txt, main="Correlogramma (τ di Kendall)") #
correlogramma con grafici xy e τ di Kendall
#
windows() # apre una nuova finestra
corrgram(mydata, cor.method="kendall", order=FALSE, lower.panel=panel.pts,
upper.panel=panel.ellipse, text.panel=panel.txt,
diag.panel=panel.density, main="Correlogramma (τ di Kendall)") # correlogramma con
grafici xy ed ellissi delle tendenze con curva loess
#
library(gclus) # carica il pacchetto
windows() # apre una nuova finestra
tau <- cor(mydata, method="kendall") # τ di Kendall
mydata.col <- dmat.color(abs(tau)) # applica i colori
mydata.o <- order.single(abs(tau)) # dispone le variabili meglio correlate più vicine alla
diagonale
cpairs(mydata, mydata.o, panel.colors=mydata.col, gap=.5, main="Gradiente di
correlazione (τ di Kendall)") # mostra il grafico
#
# test di gaussianità (asimmetria e curtosi)
#
library(moments) # carica il pacchetto
print(paste("numero dei dati =", n <- nrow(mydata)), quote=FALSE)
print(paste("deviazione standard del coefficiente di asimmetria (sa) =", sa <- sqrt(6/n)),
quote=FALSE)
print(paste("deviazione standard del coefficiente di curtosi(sc) =", sc <- sqrt(24/n)),
quote=FALSE)
#
ca <- skewness(mydata) # coefficiente di asimmetria (ca)
```

ca_sa <- round(abs(ca/sa), digits=2) # rapporto (ca/sa)
#
cc <- kurtosis(mydata)-3 # coefficiente di curtosi (cc)
cc_sc <- round(abs(cc/sc), digits=2) # rapporto (cc/sc)
#
ascurt <- data.frame(ca, ca_sa, cc, cc_sc) # costruisce la tabella
names(ascurt) <- c("ca", "ca/sa", "cc", "cc/sc") # assegna i nomi alle colonne
ascurt # mostra la tabella
#

Lo script ricalca il precedente. Anche in questo caso l'argomento **upper.panel=panel.conf** che riporta nel correlogramma i limiti di confidenza del coefficiente di correlazione e che può essere impiegato solamente con il test **r** di Pearson viene sostituito con l'argomento **upper.panel=panel.cor**

Lo sola differenza sostanziale, a parte ovviamente il differente coeffiiente di correlazione, è la sostituzione della funzione **rcorr()**, che non prevede il calcolo del test **τ** di Kendall, con la funzione **cor()**. Il fatto che questa non riporti i valori di `p` di significatività del **τ** di Kendall si risolve ricorrendo di volta in volta al confronto diretto tra le due variabili che interessano [3].

La conclusione?

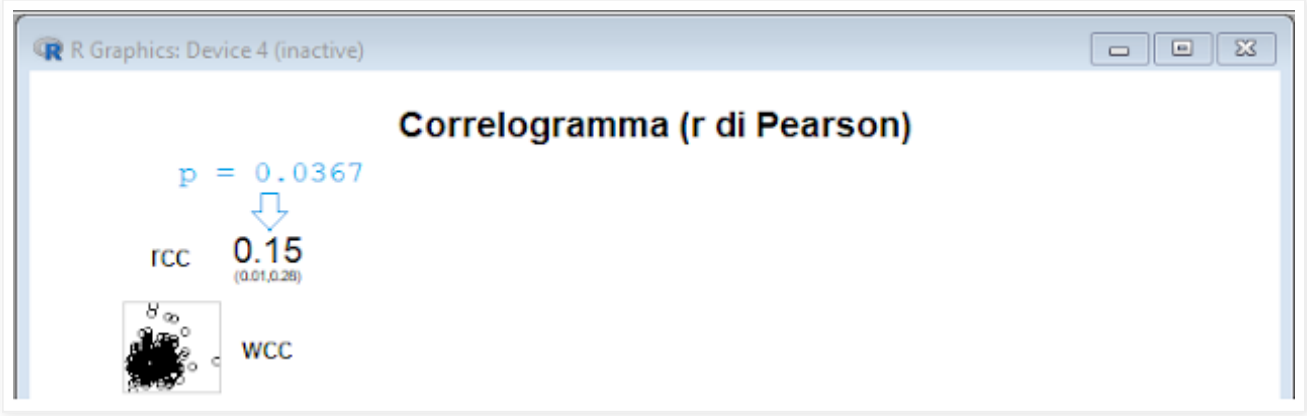
Non è vero che ricercare una correlazione equivale semplicemente a calcolare il coefficiente di correlazione lineare **r** *e la sua significatività*.

Integrando i risultati:
→ del coefficiente di correlazione e della sua significatività
→ della valutazione grafica della linearità
→ dei test di normalità (gaussianità) dei dati
abbiamo le basi per decidere se sia opportuno basare le nostre conclusioni sul più classico e tradizionale test **r** di Pearson, test parametrico basato sull'assunto di gaussianità e di linearità dei dati, o ricorrere in alternativa a un test non parametrico, il coefficiente di correlazione per ranghi **ρ** (rho) di Spearman oppure il coefficiente di correlazione **τ** (tau) di Kendall, che forniscono una misura della correlazione valida anche quando i dati non soddisfano i requisiti di normalità (gaussianità) e di linearità necessari per applicare **r**.

Ad esempio per le variabili `hc` e `hg` (per il significato delle variabili vedere il post Il set di dati ais) abbiamo che:
→ il valore del coefficiente di correlazione lineare **r** di Pearson è elevato ed è significativo;
→ i grafici mostrano una evidente relazione lineare tra le due variabili;
→ i test di normalità (gaussianità) confermano per entrambe le variabili una distribuzione gaussiana;
pertanto possiamo basare le nostre conclusioni in merito alla correlazione sul coefficiente di correlazione lineare **r** di Pearson.

Nel caso della correlazione tra le variabili `bmi` e `ssf` abbiamo che:
→ i valori del coefficiente di correlazione **ρ** (rho) di Spearman e del coefficiente di correlazione **τ** (tau) di Kendall sono elevati e sono significativi;
→ i grafici ci mostrano tra le due variabili una relazione praticamente casuale (difficile poterla considerare "lineare");
→ i test di normalità (gaussianità) confermano per entrambe le variabili una distribuzione non gaussiana;
pertanto **r** non è applicabile e per le nostre conclusioni in merito alla correlazione dobbiamo ricorrere a uno dei due test non parametrici.

Ma non è tutto. Nel confronto che abbiamo fatto tra le variabili `rcc` e `wcc`



abbiamo un esempio tipico di una situazione nella quale un **r** significativo (`p = 0.0367`) è fuorviante e sarebbe sbagliato basarsi solamente su di esso senza considerare il contesto nel quale è stato generato, infatti:
→ una delle due condizioni per applicare il *coefficiente di correlazione lineare* **r** *di Pearson* è che i dati siano distribuiti in modo gaussiano mentre i test mostrano che la variabile `rcc` ha una asimmetria significativa e la variabile `wcc` ha asimmeria e curtosi significative, quindi i dati non sono distribuiti in modo gaussiano;
→ l'altra delle due condizioni per applicare il coefficiente di correlazione lineare **r** di Pearson è che i dati seguano un andamento lineare. Essendo `r = 0.15` abbiamo che **R^2** – il **coefficiente di determinazione**, che rappresenta la variabilità spiegata dalla regressione lineare – è uguale a `0.0225` il che significa che solamente il `2.2%` della variabilità osservata è spiegata dalla regressione lineare, troppo poco per assumere che la relazione tra `rcc` e `wcc` possa essere rappresentata mediante una retta;
→ il grafico xy di `rcc` e `wcc` mostra una distribuzione dei punti in un'area sostanzialmente circolare, nella quale non vi è alcuna tendenza che possa essere ragionevolmente approssimata mediante una retta;
→ queste osservazioni confermano che il coefficiente di correlazione **r** non è applicabile e indicano la necessità di ricorrere a un coefficiente di correlazione non parametrico – il *coefficiente di correlazione per ranghi* **ρ** (rho) *di Spearman* o il *coefficiente di correlazione* **τ** (tau) *di Kendall* – che è in grado di fornire una misura della correlazione valida anche quando i dati non soddisfano i requisiti di normalità (gaussianità) e di linearità previsti per **r**.

C'è infine dell'altro che coinvolge tutti i coefficienti di correlazione in quanto:
→ per valori identici del coefficiente di correlazione, tanto maggiore è il numero di dati sui quali un valore è stato calcolato, tanto maggiore è la sua significatività (cioè tanto minore è il valore di `p`) – in altre parole è possibile rendere significativo un coefficiente di correlazione aumentando opportunamente il numero dei dati senza che il coefficiente cambi valore;
→ il coefficiente di correlazione misura la forza del legame tra due variabili ponendola tra `0` e `1` e l'esperienza insegna che valori inferiori a `0.5` corrispondono a correlazioni che non hanno rilevanza pratica – qui abbiamo `0.15` per **r**, `0.17` per **ρ**, `0.11` per **τ**, cioè valori ormai prossimi allo `0`, in sostanza valori inadeguati;
→ a prescindere dal fatto che **r** in questo caso non deve essere impiegato per le ragioni riportate sopra, questi valori inadeguati del coefficiente di correlazione risultano paradossalmente significativi sia per **r** (`p = 0.0367`), sia per **ρ** (`p = 0.01697`), sia per **τ** (`p = 0.02104`) in virtù del numero elevato (`202`) di dati sui quali è stato calcolato;
→ è ragionevole concludere che i coefficienti di correlazione **r**, **ρ** e **τ** in questo caso sono falsamente significativi:
a) a causa della distorsione indotta sui risultati dalla numerosità campionaria;
b) perché `p` è appena al di sotto della soglia del 5% (`p = 0.05`) e potremmo optare per una più prudente soglia dell'1% (`p = 0.01`) che li renderebbe tutti e tre non significativi;
c) perché dal grafico non si riesce proprio a immaginare un qualsivoglia tipo di relazione che colleghi le due variabili;

d) in quanto (soprattutto) sappiamo che nei soggetti sani (come sono gli atleti australiani i cui valori ematologici e biometrici sono riportati nel set di dati **ais** qui impiegato) i meccanismi di produzione dei globuli rossi (`rcc`) e dei globuli bianchi (`wcc`) del sangue sono tra loro indipendenti e quindi non in relazione l'uno con l'altro.

Quest'ultima osservazione ci riporta ancora ai princìpi generali soggiacenti alla correlazione ricordandoci che:
→ uno studio di correlazione ha un senso solamente quando sia supportato da un modello di un evento – fisico, chimico-fisico, biologico, fisiologico, econometrico o quant'altro – che si assume spieghi e descriva adeguatamente la relazione di funzione che intercorre tra le variabili;
→ è indispensabile verificare che gli assunti alla base del modello di correlazione impiegato siano rispettati;
→ esistono alternative al coefficiente di correlazione lineare **r** che è necessario utilizzare quando detti assunti non sono rispettati;
→ correlazione statisticamente significativa non significa causazione [4], l'assunzione implicita di un rapporto di causa-effetto in seguito ad un coefficiente di correlazione significativo è una trappola subdola e ricorrente nella statistica [5].

Nonostante questi limiti, in **R** sono disponibili gli strumenti che, impiegati in modo appropriato, possono fornire un valido aiuto quando si vogliono analizzare i dati alla ricerca di una possibile correlazione, che in ogni caso va interpretata con attenzione, cautela e spirito critico.

---

[1] Cerco per quanto possibile di evitare l'aggettivo "*normale*" in quanto viene impiegato con troppi differenti significati: la distribuzione cui ci si riferisce è a rigore la "*distribuzione gaussiana*".

[2] Trovate i relativi manuali in: *Available CRAN Packages By Name*. URL consultato il 20/04/2022: https://goo.gl/hLC9BB

[3] Vedere il post Ricercare una possibile correlazione (tra due variabili).

[4] "*Correlation is not causation*". In: Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 103.

[5] Vedere quanto ho già sottolineato nel post Correlazione e causazione e quanto ci ricorda Tyler Vigen nel suo Spurious correlations.

sabato 23 aprile 2022

Ricercare una possibile correlazione (tra due variabili)

La ricerca di una possibile correlazione, intesa come la tendenza di due grandezze a variare congiuntamente in modo statisticamente significativo, richiede – oltre alla necessità ineludibile di spiegare per quale ragione l'una dipenda dall'altra – un approccio integrato che include il calcolo del coefficiente di correlazione, la regressione lineare e l'esecuzione dei test di normalità. Lo vediamo ora con tre script da tenere a portata di mano per un impiego immediato al bisogno.

Gli script consentono per la parte statistica di calcolare:
→ il **coefficiente di correlazione lineare r di Pearson** (test parametrico)
→ il **coefficiente di correlazione per ranghi ρ (rho) di Spearman** (test non parametrico)
→ il **coefficiente di correlazione τ (tau) di Kendall** (test non parametrico)

Quale dei tre coefficienti impiegare per trarre dai propri dati le conclusioni in merito alla significatività della correlazione deve essere valutato di volta in volta alla luce delle seguenti considerazioni:
1) come suggerisce anche la sua denominazione completa e corretta, il coefficiente di correlazione lineare **r** di Pearson fornisce una misura accurata del grado di correlazione quando sono soddisfatti i seguenti due requisiti: (i) i dati hanno una distribuzione gaussiana [1] in quanto si tratta di un test parametrico; (ii) tra le due variabili a confronto esiste una relazione lineare;
2) i due coefficienti di correlazione non parametrici (**ρ** e **τ**) forniscono una misura della correlazione valida anche quando i dati non sono distribuiti in modo gaussiano e tra le due variabili a confronto non esiste una relazione lineare;
3) i test non parametrici sono più conservativi cioè tendono a fornire valori di `p` superiori (quindi valori di `p` meno significativi) rispetto al test parametrico; se si preferisce essere più prudenti nelle conclusioni, si possono applicare i test non parametrici anche quando i dati soddisfano i requisiti di gaussianità e di linearità che consentirebbero l'applicazione del test parametrico. Il coefficiente di correlazione **τ** di Kendall è il più conservativo.

Per eseguire gli script è necessario disporre dei pacchetti **mblm** e **moments** [2], se non li avete dovete scaricarli e installarli dal **CRAN**. Gli script prevedono tutti la stessa sequenza logica che include:
a) il calcolo dei tre *coefficienti di correlazione* per una immediata comparazione dei loro risultati;
b) il calcolo della *significatività* dei coefficienti di correlazione;
c) un grafico xy con la *retta di regressione*;
d) i test per la valutazione della *normalità* (gaussianità) dei dati.

Con il primo script viene calcolato il **coefficiente di correlazione lineare r di Pearson** (test parametrico) e viene rappresentata la regressione lineare parametrica. Copiate e incollate nella `Console di R` lo script e premete `↵ Invio`:

```
# Coefficiente di correlazione lineare r di Pearson e regressione lineare parametrica
#
x <- c(18, 46, 89, 72, 96, 63, 23, 58, 36, 82, 88) # variabile in ascisse
y <- c(21, 41, 94, 64, 91, 71, 19, 66, 42, 89, 79) # variabile in ordinate
#
# coefficienti di correlazione
#
options(scipen=999) # predispone espressione dei numeri in formato fisso
cor.test(x, y, method="pearson") # r di Pearson
cor.test(x, y, method="spearman") # ρ (rho) di Spearman
cor.test(x, y, method="kendall") # τ (tau) di Kendall
#
# test di gaussianità
#
library(moments) # carica il pacchetto
agostino.test(x) # test di D'Agostino per il coefficiente di asimmetria di x
anscombe.test(x) # test di Anscombe-Glynn per il coefficiente di curtosi
#
agostino.test(y) # test di D'Agostino per il coefficiente di asimmetria di y
anscombe.test(y) # test di Anscombe-Glynn per il coefficiente di curtosi
#
shapiro.test(x) # test di Shapiro-Wilk
shapiro.test(y)
#
# grafico xy
#
plot(x, y, xlim = c(min(x), max(x)), ylim = c(min(y), max(y)), main="Regressione lineare
(parametrica)")
#
# aggiunge al grafico la regressione lineare (parametrica) con r di Pearson
#
reglin <- lm(y~x) # calcola la regressione lineare
a <- reglin$coefficients[1] # intercetta a
b <- reglin$coefficients[2] # coefficiente angolare b
r <- cor(x, y, method="pearson") # coefficiente di correlazione r
#
lines(c(min(x), max(x)), c(a+b*min(x), a+b*max(x)), col="black", lty=2, lwd=1) # traccia
la retta di regressione
#
legend(min(x), max(y), legend=c(paste("Intercetta a =", round(a, digits=0)),
paste("Coefficiente angolare b =", round(b, digits=3)), paste("Equazione: y =", round(a,
digits=0), ifelse(sign(b) == 1, "+", "-"), round(b, digits=3), "· x"), paste("r di Pearson =",
round(r, digits=3)))) # aggiunge al grafico la legenda
#
options(scipen=0) # ripristina espressione dei numeri in formato esponenziale
#
```

Dopo avere definito nelle prime due righe di codice i dati da analizzare come **x** e come **y**, il che consente di sostituirli agevolmente con i propri, e dopo avere predisposto l'espressione dei numeri in formato fisso con **options(scipen=999)**, mediante la funzione **cor.test()** sono subito calcolati i tre coefficienti di correlazione.

```
> cor.test(x, y, method="pearson") # r di Pearson

        Pearson's product-moment correlation
```

```
data:  x and y
t = 11.83, df = 9, p-value = 0.0000008695
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8827108 0.9922369
sample estimates:
      cor
0.9693169

> cor.test(x, y, method="spearman") # ρ (rho) di Spearman

        Spearman's rank correlation rho

data:  x and y
S = 14, p-value < 0.00000000000000022
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.9363636

> cor.test(x, y, method="kendall") # τ (tau) di Kendall

        Kendall's rank correlation tau

data:  x and y
T = 49, p-value = 0.0003334
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.7818182
```

I coefficienti **r** di Pearson (`p = 0.0000008695`), **ρ** (rho) di Spearman (`p < 0.00000000000000022`) e **τ** (tau) di Kendall (`p = 0.0003334`) sono sempre significativi. Quest'ultimo con suo il valore inferiore di `p` conferma di essere il più conservativo dei tre.

Il test di asimmetria

```
> agostino.test(x) # test di D'Agostino per il coefficiente di asimmetria

        D'Agostino skewness test

data:  x
skew = -0.30031, z = -0.55045, p-value = 0.582
alternative hypothesis: data have a skewness
```

e il test di curtosi

```
> anscombe.test(x) # test di Anscombe-Glynn per il coefficiente di curtosi

        Anscombe-Glynn kurtosis test

data:  x
kurt = 1.7415, z = -1.1572, p-value = 0.2472
alternative hypothesis: kurtosis is not equal to 3
```

confermano una distribuzione normale della variabile `x`.

Di nuovo il test di asimmetria

```
> agostino.test(y) # test di D'Agostino per il coefficiente di asimmetria

        D'Agostino skewness test

data:  y
skew = -0.38458, z = -0.70302, p-value = 0.482
alternative hypothesis: data have a skewness
```

e il test di curtosi

```
> anscombe.test(y) # test di Anscombe-Glynn per il coefficiente di curtosi

        Anscombe-Glynn kurtosis test

data:  y
kurt = 1.7976, z = -1.0283, p-value = 0.3038
alternative hypothesis: kurtosis is not equal to 3
```

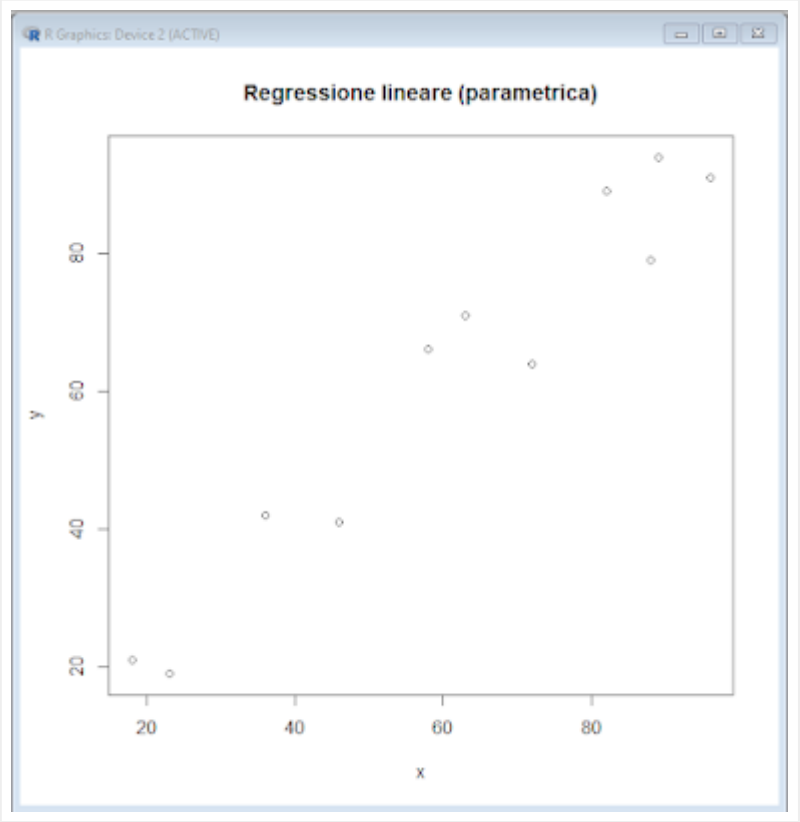
confermano una distribuzione gaussiana della variabile `y`.

Anche un test globale per la normalità

```
> shapiro.test(x) # test di Shapiro-Wilk

        Shapiro-Wilk normality test

data:  x
W = 0.93397, p-value = 0.4524

> shapiro.test(y)

        Shapiro-Wilk normality test

data:  y
W = 0.91105, p-value = 0.2509
```

conferma che le due variabili sono distribuite in modo gaussiano.

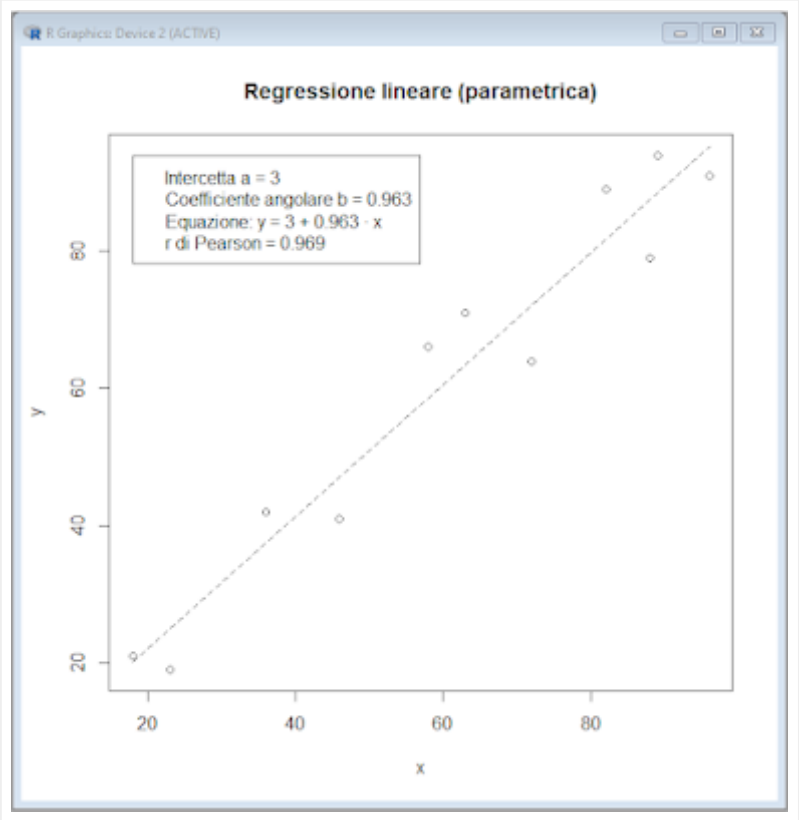
Integriamo i test statistici con un semplice grafico xy dei dati realizzato mediante la funzione **plot()**. Da notare che gli argomenti che definiscono limite inferiore e limite superiore dell'asse delle `x` (**xlim**) e dell'asse delle `y` (**ylim**) sono parametrizzati, nel senso che ricavano valore minimo (**min(x)** e **min(y)**) e valore massimo (**max(x)** e **max(y)**) impiegati nel grafico dai dati in ingresso, in modo che cambiando i dati le scale degli assi siano automaticamente riadattate. Ma nulla impedisce di sostituire tali valori di volta in volta con valori numerici fissi che possono fornire una miglior rappresentazione del grafico (in questo caso ad esempio sarebbe più appropriato dimensionare entrambi gli assi tra da `0` a `100`).



A questo punto sappiamo di avere:
→ due variabili che i test statistici ci indicano avere una distribuzione gaussiana;
→ un grafico che ci indica che tra le due sussiste, nell'ambito dei valori osservati, una relazione lineare.

Pertanto abbiamo quanto ci serve per essere autorizzati a:
→ riportare la correlazione tra le due variabili sotto forma del **coefficiente di correlazione lineare r di Pearson**;
→ descrivere la relazione di funzione che collega le due variabili mediante l'**equazione della retta di regressione**.

L'equazione della retta viene calcolata con il tradizionale metodo parametrico (dei minimi quadrati) come **lm(y~x)** [3] e il coefficiente di correlazione **r** viene calcolato come **cor(x, y, method="pearson")**. L'intercetta **a** e il coefficiente angolare **b** della retta di regressione sono impiegati per riportarla mediante la funzione **lines()** nel grafico, che viene infine completato aggiungendo mediante la funzione **legend()** una legenda che riporta l'equazione della retta e il coefficiente di correlazione **r**.



Lo script si chiude ripristinando l'espressione dei numeri in formato esponenziale con **options(scipen=0)**.

Vediamo ora cosa accade fare quando i dati non sono distribuiti normalmente e quindi viene impiegato il **coefficiente di correlazione per ranghi ρ (rho) di Spearman** (test non parametrico) e viene rappresentata la retta di regressione lineare non parametrica.

Copiate e incollate nella `Console di R` lo script, che ricalca il precedente, e premete ↵ `Invio`:

```
# Coefficiente di correlazione per ranghi ρ (rho) di Spearman e regressione lineare non parametrica di Siegel
#
x <- c(18, 16, 19, 12, 17, 13, 15, 58, 64, 36, 82, 88) # variabile in ascisse
y <- c(21, 14, 15, 11, 20, 18, 19, 46, 38, 42, 89, 79) # variabile in ordinate
#
# coefficienti di correlazione
#
options(scipen=999) # predispone espressione dei numeri in formato fisso
cor.test(x, y, method="pearson") # r di Pearson
cor.test(x, y, method="spearman") # ρ (rho) di Spearman
cor.test(x, y, method="kendall") # τ (tau) di Kendall
#
# test di gaussianità
#
library(moments) # carica il pacchetto
agostino.test(x) # test di D'Agostino per il coefficiente di asimmetria di x
anscombe.test(x) # test di Anscombe-Glynn per il coefficiente di curtosi
#
agostino.test(y) # test di D'Agostino per il coefficiente di asimmetria di y
anscombe.test(y) # test di Anscombe-Glynn per il coefficiente di curtosi
#
shapiro.test(x) # test di Shapiro-Wilk
shapiro.test(y)
#
# grafico xy
#
plot(x, y, xlim = c(min(x), max(x)), ylim = c(min(y), max(y)), main="Regressione lineare di Siegel (non parametrica)")
#
# aggiunge al grafico la regressione lineare non parametrica con ρ (rho) di Spearman
#
library(mblm) # carica il pacchetto per la regressione di Siegel
reglin <- mblm(y ~ x, repeated=TRUE) # calcola la regressione lineare non parametrica di Siegel
a <- reglin$coefficients[1] # intercetta a
b <- reglin$coefficients[2] # coefficiente angolare b
rho <- cor(x, y, method="spearman") # rho di Spearman
#
lines(c(min(x), max(x)), c(a+b*min(x), a+b*max(x)), col="black", lty=2, lwd=1) # traccia la retta di regressione
#
legend(min(x), max(y), legend=c(paste("Intercetta a =", round(a, digits=0)), paste("Coefficiente angolare b =", round(b, digits=3)), paste("Equazione: y =", round(a, digits=0), ifelse(sign(b) == 1, "+", "-"), round(b, digits=3), "· x"), paste("rho di Spearman =", round(rho, digits=3)))) # aggiunge al grafico la legenda
#
options(scipen=0) # ripristina espressione dei numeri in formato esponenziale
#
```

In questo caso uno dei test indica una significativa non normalità / non gaussianità nei dati ($p < 0.05$):

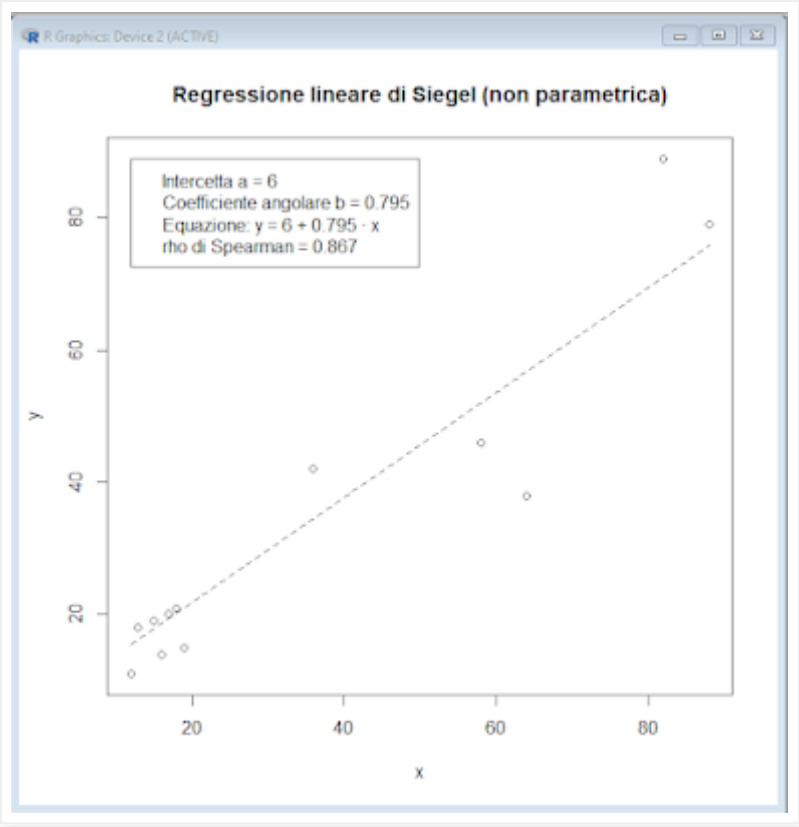
```
> shapiro.test(x) # test di Shapiro-Wilk

        Shapiro-Wilk normality test
```

```
data:  x
W = 0.79299, p-value = 0.007807

> shapiro.test(y)

        Shapiro-Wilk normality test

data:  y
W = 0.80417, p-value = 0.01045
```

A questo punto abbiamo:
→ due variabili che non sono adeguatamente distribuite in modo gaussiano;
→ un grafico che ci indica che tra le due, nell'ambito dei valori osservati, sussiste quella che possiamo ancora ragionevolmente ritenere una relazione lineare.

Le indicazioni che dobbiamo seguire sono pertanto:
→ esprimere la correlazione mediante un coefficiente di correlazione non parametrico – in questo caso impieghiamo il **coefficiente di correlazione per ranghi ρ (rho) di Spearman**;
→ riportare una regressione lineare non parametrica – in questo caso impieghiamo la **regressione lineare non parametrica di Siegel** [4].

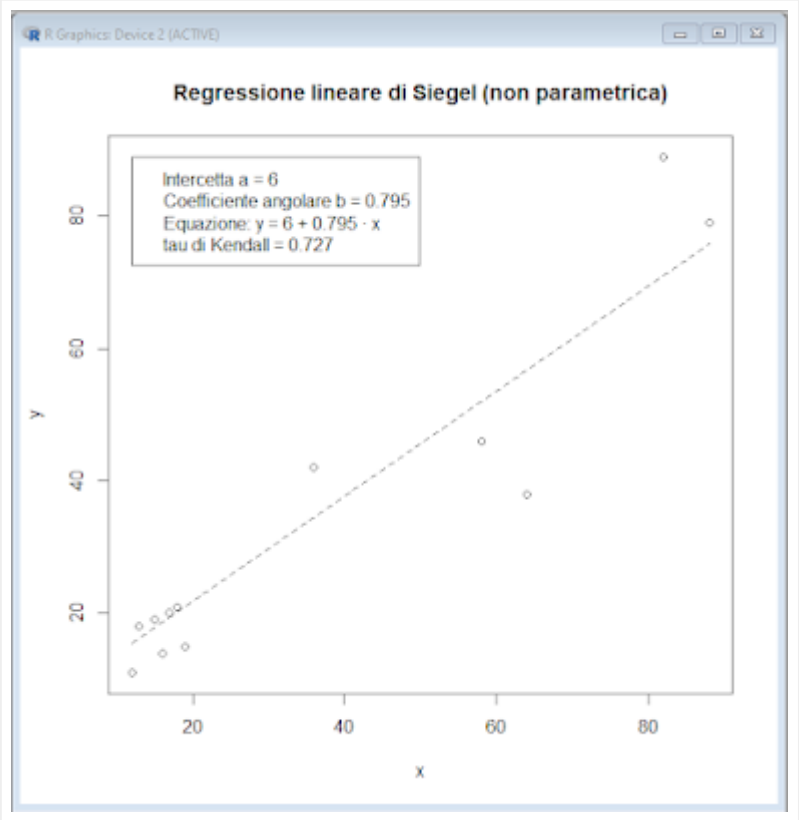
Questo è il grafico generato per presentare in forma sintetica i risultati:



Se preferite in alternativa impiegando gli istessi dati potete esprimere la correlazione mediante il **coefficiente di correlazione τ (tau) di Kendall** (test non parametrico) e rappresentando sempre la retta di regressione lineare non parametrica [5]. Per farlo copiate e incollate nella `Console di R` quest'ultimo script e premete ↵ `Invio`:

```
# Coefficiente di correlazione τ (tau) di Kendall e regressione lineare non parametrica di Siegel
#
x <- c(18, 16, 19, 12, 17, 13, 15, 58, 64, 36, 82, 88) # variabile in ascisse
y <- c(21, 14, 15, 11, 20, 18, 19, 46, 38, 42, 89, 79) # variabile in ordinate
#
# coefficienti di correlazione
#
options(scipen=999) # predispone espressione dei numeri in formato fisso
cor.test(x, y, method="pearson") # r di Pearson
cor.test(x, y, method="spearman") # ρ (rho) di Spearman
cor.test(x, y, method="kendall") # τ (tau) di Kendall
#
# test di gaussianità
#
library(moments) # carica il pacchetto
agostino.test(x) # test di D'Agostino per il coefficiente di asimmetria di x
anscombe.test(x) # test di Anscombe-Glynn per il coefficiente di curtosi
#
agostino.test(y) # test di D'Agostino per il coefficiente di asimmetria di y
anscombe.test(y) # test di Anscombe-Glynn per il coefficiente di curtosi
#
shapiro.test(x) # test di Shapiro-Wilk
shapiro.test(y)
#
# grafico xy
#
plot(x, y, xlim = c(min(x), max(x)), ylim = c(min(y), max(y)), main="Regressione lineare
di Siegel (non parametrica)")
#
# aggiunge al grafico la regressione lineare non parametrica e τ (tau) di Kendall
#
library(mblm) # carica il pacchetto per la regressione di Siegel
reglin <- mblm(y ~ x, repeated=TRUE) # calcola la regressione lineare non parametrica di Siegel
a <- reglin$coefficients[1] # intercetta a
b <- reglin$coefficients[2] # coefficiente angolare b
tau <- cor(x, y, method="kendall") # tau di Kendall
#
lines(c(min(x), max(x)), c(a+b*min(x), a+b*max(x)), col="black", lty=2, lwd=1) # traccia
la retta di regressione
#
legend(min(x), max(y), legend=c(paste("Intercetta a =", round(a, digits=0)),
paste("Coefficiente angolare b =", round(b, digits=3)), paste("Equazione: y =", round(a,
digits=0), ifelse(sign(b) == 1, "+", "-"), round(b, digits=3), "· x"), paste("tau di Kendall
=", round(tau, digits=3)))) # aggiunge al grafico la legenda
#
options(scipen=0) # ripristina espressione dei numeri in formato esponenziale
#
```

La differenza sostanziale rispetto al **ρ** (rho) di Spearman è un coefficiente di correlazione **τ** (tau) di Kendall che risulta meno significativo, come previsto per un test più conservativo. Questo è il grafico che illustra in forma sintetica i risultati.



La conclusione?

Non è vero che ricercare una correlazione equivale semplicemente a calcolare il coefficiente di correlazione lineare **r** *e la sua significatività*.

Infatti è necessario ricordare i princìpi generali soggiacenti alla correlazione:
→ uno studio di correlazione ha un senso solamente quando sia supportato da un modello di un evento – fisico, chimico-fisico, biologico, fisiologico, econometrico o quant'altro – che si assume spieghi e descriva adeguatamente la relazione di funzione che intercorre tra le variabili;
→ è indispensabile verificare che gli assunti alla base del modello di correlazione impiegato siano rispettati;
→ esistono alternative al coefficiente di correlazione lineare **r** che è necessario utilizzare quando detti assunti non sono rispettati;
→ correlazione statisticamente significativa non significa causazione [6], l'assunzione implicita di un rapporto di causa-effetto in seguito ad un coefficiente di correlazione significativo è una delle trappole subdola e ricorrente nella statistica [7].

Nonostante questi limiti, in **R** sono disponibili gli strumenti che, impiegati in modo appropriato, possono fornire un valido aiuto quando si vogliono analizzare i dati alla ricerca di una possibile correlazione, che in ogni caso va interpretata con attenzione, cautela e spirito critico.

---

[1] Cerco per quanto possibile di evitare l'aggettivo "*normale*" in quanto viene impiegato con troppi differenti significati: la distribuzione cui ci si riferisce è a rigore la "*distribuzione gaussiana*".

[2] Trovate i relativi manuali in: *Available CRAN Packages By Name*. URL consultato il 20/04/2022: https://goo.gl/hLC9BB

[3] Per i dettagli sull'impiego della funzione e sulla rappresentazione della retta vedere i post La regressione lineare: assunti e modelli e il post Regressione lineare semplice parametrica.

[4] Due altri metodi per il calcolo della regressione lineare non parametrica, alternativi a quello di Siegel, sono riportati nel post Regressione lineare semplice non parametrica.

[5] In realtà i modelli di regressione qui impiegati, parametrici e non parametrici, prevedono tutti che x sia la variabile indipendente mentre i dati reali che si intende analizzare potrebbero non rispettare questo assunto. Per questo tema, e per i modelli di regressione lineare alternativi che devono essere presi in considerazione, vedere il post La regressione lineare: assunti e modelli.

[6] "*Correlation is not causation*". In: Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, p. 103.

[7] Vedere quanto ho già sottolineato nel post Correlazione e causazione e quanto ci ricorda Tyler Vigen nel suo Spurious correlations.

lunedì 25 aprile 2022

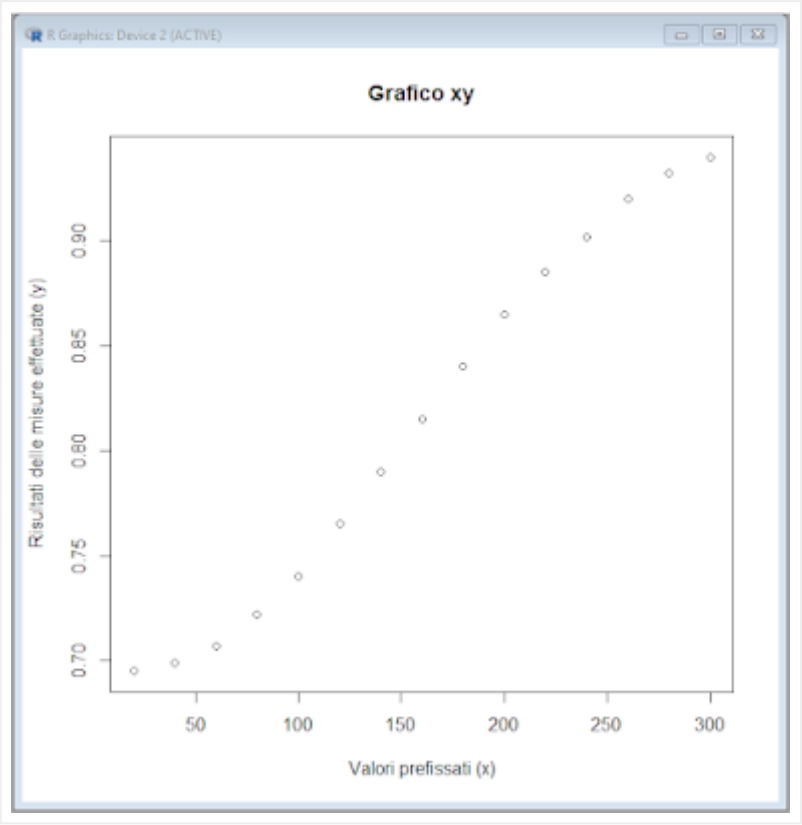
Regressione polinomiale di terzo grado

Supponiamo di avere, per ciascuno di una serie di valori prefissati `x` (riportati in ascisse), il risultato di altrettante misure `y` (riportate in ordinate) e con la funzione **plot()** visualizziamo i dati sotto forma di un grafico `xy`.

Copiate il codice che segue nella `Console di R` e premete `↵ Invio`:

```
#
x <- c(20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300) # valori in ascisse
y <- c(0.695, 0.699, 0.707, 0.722, 0.740, 0.765, 0.790, 0.815, 0.840, 0.865, 0.885, 0.902, 0.920, 0.932, 0.940) # valori in ordinate
plot(y~x, col="black", pch=1, main="Grafico xy", xlab="Valori prefissati (x)", ylab="Risultati delle misure effettuate (y)") # traccia il grafico xy
#
```

Dalla ispezione del grafico risulta evidente una relazione non lineare.



Questo ci fa escludere la possibilità di descrivere la relazione di funzione che intercorre tra le due variabili mediante una retta. E pone il tema di quale sia la curva che potrebbe meglio descriverla.

La prima cosa da valutare è se esiste un modello deterministico di un evento fisico, chimico-fisico, biologico, econometrico o quant'altro, che si assume descriva adeguatamente la relazione di funzione che intercorre tra le due variabili: in tal caso non resta che applicare l'equazione fornita dal modello. In caso contrario è necessario ricorrere ad un modello empirico: ed è questo che consideriamo essere il caso nostro.

Qualora si debba ricorrere ad un modello empirico, la regola fondamentale è impiegare l'equazione più semplice compatibile con una adeguata descrizione dei dati. Nel nostro caso visto che è evidente

dall'ispezione del grafico che l'equazione di una retta

$$y = a + b \cdot x$$

calcolata mediante la classica regressione lineare con il metodo dei minimi quadrati sarebbe inadeguata a descrivere la relazione di funzione che lega la y alla x, possiamo pensare di ricorrere ad una equazione appena più complessa, come quella fornita da una **regressione polinomiale di secondo grado** nella forma

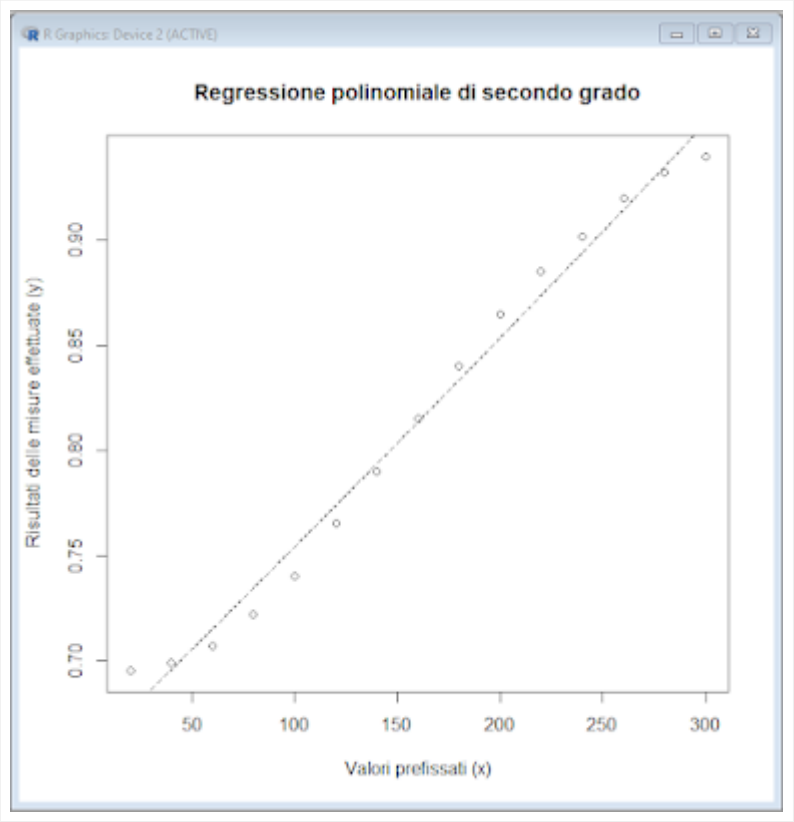
$$y = a + b \cdot x + c \cdot x^2$$

e calcolata anch'essa con il metodo dei minimi quadrati [1].

Copiate lo script nella `Console di R` e premete `↵ Invio`:

```
# REGRESSIONE POLINOMIALE di secondo grado
#
x <- c(20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300) # valori in ascisse
y <- c(0.695, 0.699, 0.707, 0.722, 0.740, 0.765, 0.790, 0.815, 0.840, 0.865, 0.885, 0.902, 0.920, 0.932, 0.940) # valori in ordinate
#
options(scipen=999) # esprime i numeri in formato fisso
#
# calcola la regressione
#
polisec <- lm(y~poly(x, degree=2, raw=TRUE)) # calcola la regressione
summary(polisec) # riporta le statistiche della regressione
#
# traccia il grafico
#
newx <- seq(min(x), max(x), (max(x)-min(x))/1000) # valori x in corrispondenza dei quali calcolare il valore del polinomio
newy <- predict(polisec, list(x=newx)) # calcola il valore corrispondente del polinomio
plot(y~x, col="black", pch=1, main="Regressione polinomiale di secondo grado", xlab="Valori prefissati (x)", ylab="Risultati delle misure effettuate (y)") # predispone il grafico e riporta i dati
lines(newx, newy, col="black", lty=2, lwd=1) # traccia il polinomio
#
options(scipen=0) # ripristina la notazione scientifica
#
```

La prima cosa che notiamo è che il polinomio di secondo grado descrive in pratica una retta.



Inoltre se andate alla quinta riga di codice tra le statistiche generate con la funzione **summary(polisec)** trovate:

```
Coefficients:
                                    Estimate   Std. Error t value             Pr(>|t|)
(Intercept)                     0.6573076923 0.0100102431  65.664 < 0.0000000000000002 ***
poly(x, degree = 2, raw = TRUE)1 0.0009554000 0.0001439488   6.637             0.000024 ***
poly(x, degree = 2, raw = TRUE)2 0.0000001299 0.0000004374   0.297                0.772
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Mentre l'intercetta `a` con un `p=0.0000000000000002` e il coefficiente di primo grado `b` del polinomio con un `p=0.000024` sono significativi, il coefficiente di secondo grado `c` con un `p=0.772` non lo è affatto. Pertanto l'equazione del polinomio di secondo grado

$$y = a + b \cdot x + c \cdot x^2$$

privata del coefficiente $c \cdot x^2$ si riduce all'equazione di una retta

$$y = a + b \cdot x$$

cosa che conferma il grafico ottenuto.

Dati che descrivono una curva con un raggio di curvatura omogeneo, senza massimi, senza minimi e senza punti di flesso - cioè senza punti di passaggio tra due curvature che vanno in senso opposto - sono compatibili con il ramo di una parabola descritta appunto da un polinomio di secondo grado. Ma in questo caso i dati presentano un punto di flesso tra una curvatura iniziale rivolta verso l'alto e una curvatura finale rivolta verso il basso. Il polinomio di secondo grado media tra le due, finendo con il fornire una curvatura talmente ridotta da potere essere assimilata ad una retta.

Dobbiamo pertanto vedere cosa accade applicando il modello immediatamente successivo, rappresentato da una **regressione polinomiale di terzo grado** nella forma

$$y = a + b \cdot x + c \cdot x^2 + d \cdot x^3$$

Copiate lo script nella `Console di R` e premete `↲ Invio`:

```
# REGRESSIONE POLINOMIALE di terzo grado
#
x <- c(20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300) # valori in ascisse
y <- c(0.695, 0.699, 0.707, 0.722, 0.740, 0.765, 0.790, 0.815, 0.840, 0.865, 0.885, 0.902, 0.920, 0.932, 0.940) # valori in ordinate
#
options(scipen=999) # esprime i numeri in formato fisso
#
# calcola la regressione
#
politer <- lm(y~poly(x, degree=3, raw=TRUE)) # calcola la regressione
summary(politer) # riporta le statistiche della regressione
#
# traccia il grafico
#
newx <- seq(min(x), max(x), (max(x)-min(x))/1000) # valori x in corrispondenza dei quali calcolare il valore del polinomio
newy <- predict(politer, list(x=newx)) # calcola il valore corrispondente del polinomio
plot(y~x, col="black", pch=1, main="Regressione polinomiale di terzo grado", xlab="Valori prefissati (x)", ylab="Risultati delle misure effettuate (y)") # predispone il grafico e riporta i dati
lines(newx, newy, col="black", lty=2,lwd=1) # traccia il polinomio
#
options(scipen=0) # ripristina la notazione scientifica
#
```

Le prime due righe di codice servono semplicemente ad assegnare (**<-**) i valori inclusi nella funzione **c()** al vettore **x** e al vettore **y** che in questo modo conterranno i nostri dati.

Poi mediante la funzione **options()** con l'argomento **scipen=999** facciamo si che i risultati siano espressi in formato fisso, cosa che li renderà meglio leggibili (questo almeno a mio modo di vedere: se volete lasciare i risultati nella notazione scientifica eliminate questa riga di codice o, forse meglio, anteponete a questa riga di codice il simbolo #).

La quarta riga di codice assegna (**<-**) all'oggetto **politer** i risultati del calcolo della regressione polinomiale **y~poly(x)** di terzo grado (**degree=3**), l'argomento **raw=TRUE** è impiegato perché non ci interessano i polinomi ortogonali (l'opzione di default è **raw=FALSE**).

Se siete interessati ad approfondimenti, ricordo che potete visualizzare ad esempio la struttura dell'oggetto **politer** digitando **str(politer)** e potete avere informazioni sulle funzioni impiegate nello script digitando **help(***nomedellafunzione***)**.

Alla quinta riga di codice la funzione **summary(politer)** consente infine di visualizzare le statistiche generate:

```
Call:
lm(formula = y ~ poly(x, degree = 3, raw = TRUE))

Residuals:
       Min         1Q     Median         3Q        Max
-0.0032624 -0.0012957  0.0006167  0.0014060  0.0019740

Coefficients:
                                  Estimate       Std. Error t value             Pr(>|t|)
(Intercept)                 0.6966205128205  0.0025292032241 275.431 < 0.0000000000000002 ***
```

```
poly(x, degree = 3, raw = TRUE)1 -0.0003180913177  0.0000662417487  -4.802             0.000552 ***
poly(x, degree = 3, raw = TRUE)2  0.0000097653837  0.0000004730826  20.642       0.000000000381 ***
poly(x, degree = 3, raw = TRUE)3 -0.0000000200739  0.0000000009739 -20.612       0.000000000387 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.001865 on 11 degrees of freedom
Multiple R-squared:  0.9997,    Adjusted R-squared:  0.9996
F-statistic: 1.081e+04 on 3 and 11 DF,  p-value: < 0.00000000000000022
```

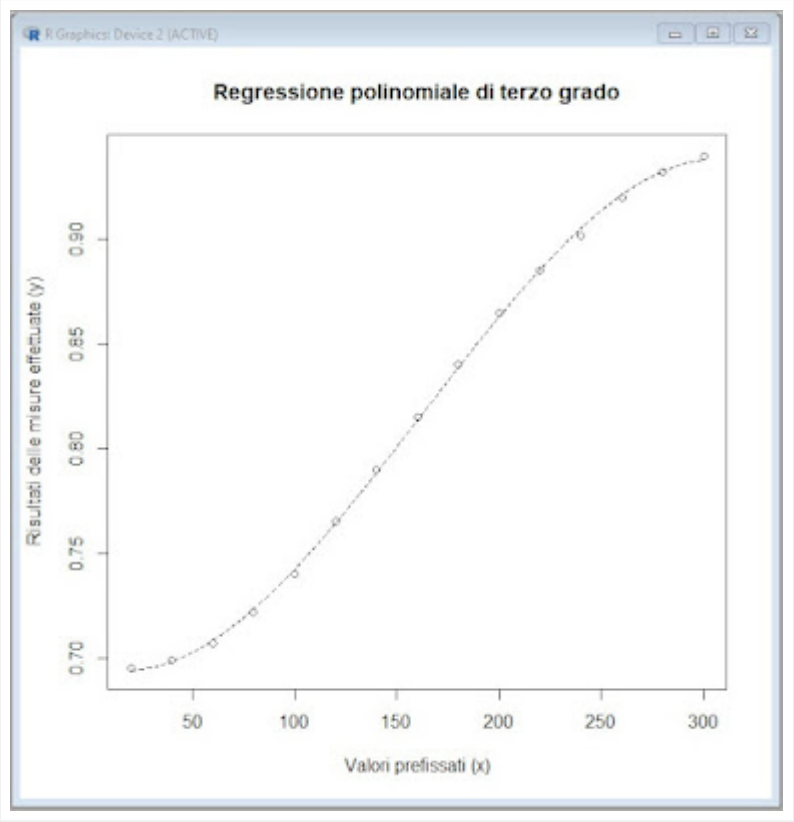
La voce `Residuals` riporta le statistiche delle differenze che, dopo l'approssimazione dei punti con il polinomio di terzo grado, residuano tra i punti stessi e la curva calcolata: differenze piccole come quelle qui osservate indicano un'ottima approssimazione della curva ai dati.

La voce `Coefficients` riporta dall'alto verso il basso il valore stimato (`Estimate`) dell'intercetta `a` (`0.6966205128205`), del coefficiente di primo grado `b` (`-0.0003180913177`), del coefficiente di secondo grado `c` (`0.0000097653837`) e del coefficiente di terzo grado `d` (`-0.0000000200739`) e ci consente di scrivere l'equazione del polinomio che sarà quindi

$$y = 0.6966205128205 - 0.0003180913177 \cdot x + 0.0000097653837 \cdot x^2 - 0.0000000200739 \cdot x^3$$

Per ciascun coefficiente viene calcolato il valore del test `t di Student` come rapporto tra il valore stimato (`Estimate`) e il suo errore standard (`Std. Error`) e viene infine riportato il valore di probabilità (`Pr(>|t|)`) di osservare per caso il valore di `t` riportato: se tale probabilità è sufficientemente bassa ci assumiamo il rischio di considerare significativo il coefficiente in questione. Nel nostro caso essendo i valori di `p` sempre molto piccoli (di gran lunga inferiori al valore `p = 0.05` generalmente considerato come valore soglia per la significatività) possiamo affermare che tutti e tre i coefficienti del polinomio di terzo grado sono significativi e lo rendono un'equazione in grado di descrivere adeguatamente i dati forniti.

Il coefficiente di correlazione multipla R^2 (`Multiple R-squared: 0.9997`) molto elevato con una statistica `F` significativa (`p-value: < 0.00000000000000022`) conferma ulteriormente la bontà dell'approssimazione. E niente meglio del grafico dei punti e del polinomio che li approssima può confermarla.



Il grafico è tracciato in modo da garantire una risoluzione grafica elevata:
→ l'intervallo compreso tra il valore minimo **min(x)** e il valore massimo **max(x)** della variabile in ascisse viene suddiviso per ottenere **1000** valori (ma potete cambiarne il numero) salvati nell'oggetto **newx** e in corrispondenza dei quali calcolare il valore del polinomio;
→ con la funzione **predict()** sono calcolati e riportati nell'oggetto **newy** i valori delle ordinate del polinomio corrispondenti ai valori delle ascisse contenuti in **newx** ;
→ la funzione **plot()** viene impiegata per riportare sul grafico di dati contenuti in **x** e in **y**;
→ la funzione **lines()** viene impiegata per sovrapporre ai dati il polinomio.

Potete riportare i dati modificando il simbolo impiegato (**pch=...**) e il suo colore (**col="..."**), come pure tracciare il polinomio modificando l'aspetto o stile della linea tracciata (**lty=...**), il suo colore (**col="..."**) e il suo spessore (**lwd=...**) [2].

Da notare che il grafico è stato tracciato esclusivamente all'interno dei dati, quindi in un'ottica di interpolazione: e questo ci ricorda anche che, trattandosi di un modello empirico, è da escludere l'impiego del polinomio per la estrapolazione, cioè per trarre conclusioni al di fuori dei dati impiegati per calcolarlo.

A chiudere lo script è la funzione **options()** che con l'argomento **scipen=0** ripristina l'espressione dei numeri nella notazione scientifica [3].

Lo script può essere riutilizzato semplicemente riportando manualmente in **x** e in **y** i nuovi dati, o meglio ancora importandoli da un file [4].

---

[1] Il **metodo dei minimi quadrati** prevede - ed è quello che abbiamo assunto essere il nostro caso - che la variabile in ascisse (x) abbia *valori prefissati*, che la variabile in ordinate (y) sia rappresentata da *misure* (affette da un errore casuale distribuito normalmente) effettuate corrispondenza delle x e *minimizza* la somma dei quadrati delle differenze y - y' ovvero delle differenze tra ciascun valore misurato y e il valore y' = f(x) corrispondente calcolato mediante la regressione (lineare o polinomiale). Esempi possono essere la risposta y di un sensore misurata in corrispondenza di una serie di concentrazioni date x di una sostanza chimica, la misurazione di un evento y in corrispondenza di una serie data di tempi x, e così via.

[2] Trovate le relative indicazioni:
→ nel post Cambiare i simboli dei punti di R,
→ nel post Visualizzare i colori disponibili in R
→ nel post Cambiare gli stili delle linee di R

[3] La *notazione scientifica* è una *notazione esponenziale*, cioè una notazione che consente di scrivere un numero N assegnato come prodotto di un opportuno numero a per la potenza di un altro numero b^k essendo b la base della notazione esponenziale. In particolare la notazione scientifica è una notazione esponenziale in base 10 ($b = 10$). Tuttavia "*notazione esponenziale*" non è sinonimo di "*notazione scientifica*" in quanto esistono notazioni esponenziali che impiegano una base diversa da quella impiegata dalla notazione scientifica: ne sono un esempio la notazione esponenziale in base 2 ($b = 2$) impiegata in informatica, e la notazione esponenziale in base e ($b = e$) - essendo il numero di Nepero e = 2.718 281 828 459... - impiegata in campo matematico e in campo scientifico.
[Precisazione aggiuntiva: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata non solo negli script ma anche nei file di dati e in tutto il testo di questo sito, quindi anche nell'espressione del numero e di Nepero.

Per le norme che regolano la punteggiatura all'interno dei numeri rimando al post Come separare i decimali e come raggruppare le cifre].

[4] Alla pagina Indice nel capitolo **R - gestione dei dati** trovate i collegamenti ai post che contengono gli script per importare i dati da file esterni.

martedì 3 maggio 2022

Come riportare i numeri in formato fisso

Se nel confronto tra due numeri sono interessato soprattutto al loro ordine di grandezza la **notazione scientifica** [1] va benissimo. Ma per un confronto puntuale preferisco il **formato fisso**.

Così, giusto per fare un esempio, quando devo confrontare due valori di probabilità `p` anziché `p=8.327701e-03` e `p=6.412771e-02` preferisco leggere `p=0.008327701` e `p=0.06412771`.

Copiate lo script che segue, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# RIPORTARE I NUMERI IN FORMATO FISSO o nella notazione scientifica
#
col_1 <- c(0.00003230609, 0.01850436, 0.008327701) # prima colonna
col_2 <- c(0.6991678, 0.00005211903, 0.02457609) # seconda colonna
mydataframe <- data.frame(col_1, col_2) # combina le due colonne in una tabella
#
mydataframe # mostra la tabella
#
options(scipen=999) # esprime i numeri in formato fisso
mydataframe # mostra la tabella
#
options(scipen=0) # ripristina la notazione scientifica
mydataframe # mostra la tabella
#
```

Come si vede a fronte dell'inserimento in una tabella di numeri espressi in formato fisso (0.00003230609, 0.01850436, 0.008327701, 0.6991678, 0.00005211903, 0.02457609) di default **R** riporta i numeri in notazione scientifica

```
> mydataframe # mostra la tabella
         col_1        col_2
1 3.230609e-05 6.991678e-01
2 1.850436e-02 5.211903e-05
3 8.327701e-03 2.457609e-02
```

ma consente di passare all'espressione dei numeri in formato fisso impiegando all'interno della funzione **options()** [2] l'argomento **scipen=999**

```
> options(scipen=999) # esprime i numeri in formato fisso
> mydataframe # mostra la tabella
          col_1         col_2
1 0.00003230609 0.69916780000
2 0.01850436000 0.00005211903
3 0.00832770100 0.02457609000
```

e di ripristinare l'espressione dei numeri in notazione scientifica impiegando all'interno della funzione **options()** l'argomento **scipen=0**

```
> options(scipen=0) # ripristina la notazione scientifica
> mydataframe # mostra la tabella
         col_1        col_2
1 3.230609e-05 6.991678e-01
```

```
2 1.850436e-02 5.211903e-05
3 8.327701e-03 2.457609e-02
```

Nota bene: in Windows nella cartella `C:\Programmi\R\R-n.n.n\etc` (dove `n.n.n` sono i numeri della versione di **R** installata) si trova il file `Rprofile.site` [3] che viene eseguito automaticamente quando si lancia **R**. Nel file `Rprofile.site` possono essere salvate le **options()** che si desidera siano impiegate da **R** di default. Il file può essere elaborato con un comune editor di testo.

Se volete che i numeri siano di default rappresentati in formato fisso, inserite nel file `Rprofile.site` la riga

options(scipen=999)

Ovviamente potete, in alternativa, utilizzare all'interno dei vostri script **options(scipen=999)** ogniqualvolta preferite il formato fisso (soluzione consigliata), ripristinando poi la notazione scientifica con **options(scipen=0)**.

Infine due indicazioni per completare il tema:
→ il razionale per decidere quale deve essere il numero di cifre significative da impiegare nella rappresentazione di un numero è riportato nel post Come stabilire il giusto numero di cifre significative;
→ per esprimere in pratica i risultati di una elaborazione statistica, che sono generalmente ottenuti con un numero di cifre in eccesso rispetto a quelle significative, vedere il post Come arrotondare i numeri.

---

[1] La *notazione scientifica* è una notazione esponenziale, cioè una notazione che consente di scrivere un numero `N` assegnato come prodotto di un opportuno numero `a` per la potenza di un altro numero b^k essendo `b` la base della notazione esponenziale. In particolare la notazione scientifica è una notazione esponenziale in base 10 (`b = 10`). Tuttavia "*notazione esponenziale*" non è sinonimo di "*notazione scientifica*" in quanto esistono notazioni esponenziali che impiegano una base diversa da quella impiegata dalla notazione scientifica: ne sono un esempio la notazione esponenziale in base 2 (`b = 2`) impiegata in informatica, e la notazione esponenziale in base *e* (`b = e`) - essendo il numero di Nepero `e` = 2.718 281 828 459... - impiegata in campo matematico e in campo scientifico.
[Precisazione aggiuntiva: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata non solo negli script ma anche nei file di dati e in tutto il testo di questo sito, quindi anche nell'espressione del numero `e` di Nepero. Per le norme che regolano la punteggiatura all'interno dei numeri rimando al post Come separare i decimali e come raggruppare le cifre].

[2] Digitate **help(options)** nella `Console di R` per scoprire gli altri argomenti della funzione **options()**.

[3] Se avete installato la versione a 32 bit di **R** in Windows trovate il file `Rprofile.site` nella cartella `C:\Programmi(x86)\R\R-n.n.n\etc` (dove `n.n.n` sono sempre i numeri della versione di **R** installata).

giovedì 11 agosto 2022

Test di Fisher

Quando si effettuano dei conteggi nei quali le osservazioni sono organizzate in una tabella di due righe per due colonne, si applicano queste regole:
→ si impiega il test chi-quadrato se le osservazioni sono indipendenti e sono numerose;
→ si impiega il **test di Fisher** se le osservazioni sono indipendenti e sono poche;
→ si impiega il test di McNemar nel caso di osservazioni non indipendenti (cioè nel caso di dati appaiati).

Armitage [1] e Snedecor [2] indicano la necessità di impiegare il **test di Fisher** (detto anche **test esatto di Fisher**) in alternativa al test chi-quadrato quando:
→ il *numero totale di osservazioni* è inferiore a 20;
→ il numero totale di osservazioni è compreso tra 20 e 40 e vi sono *frequenze attese* inferiori a 5.

Le indicazioni che potete trovare su altri test di statistica sono meno vincolanti sui limiti da applicare al numero totale di osservazioni, mentre concordano tutte nel sottolineare la necessità di applicare il test di Fisher quando in una o più celle vi è un numero ridotto di **frequenze attese**, le frequenze che dovremmo osservare in teoria e che sono calcolate per ciascuna cella come prodotto delle somme marginali (di riga e di colonna) corrispondenti, diviso per il numero totale dei casi - così ad esempio nella tabella riportata qui sotto per la cella in alto a sinistra contenente il valore osservato 4 la frequenza attesa sarà (4+16) · (4+1) / (4+16+1+21) = 20 · 5 / 42 = 2.38 e per la cella in basso a destra con il valore osservato 21 la frequenza attesa sarà (21+1) · (21+16) / (4+16+1+21) = 22 · 37 / 42 = 19.38.

Il test di Fisher lo applichiamo all'analisi della relazione che potrebbe intercorrere (e che vogliamo verificare) tra il tipo di allattamento, naturale o artificiale, e la presenza in età successiva nel bambino di malocclusione dentaria. La suzione da una tettarella in gomma non è del tutto fisiologica e potrebbe determinare un qualche effetto sulla morfogenesi delle arcate dentarie che è in atto nel lattante. I dati sono tratti da [3] e sono riportati in questa tabella che contiene quindi le **frequenze osservate**:

Allattamento	**Denti normali**	**Malocclusione**
Allattamento naturale	4	16
Allattamento artificiale	1	21

Questo script prevede di inserire i dati manualmente. Copiatelo e incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# TEST DI FISHER - 2 righe · 2 colonne
#
cells <- c(4,16,1,21) # genera l'array cells con i valori numerici contenuti nelle celle
rnames <- c("Allattamento_naturale", "Allattamento_artificiale") # genera l'array rnames con i nomi delle righe
cnames <- c("Denti_normali", "Malocclusione") # genera l'array cnames con i nomi delle colonne
mydata <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames)) # genera la matrice dei dati
mydata # mostra i dati
fisher.test(mydata) # esegue il test di Fisher
#
```

Nelle prime tre righe sono generati con la funzione **c()**:
→ il vettore che contiene i quattro dati (che come si vede devono essere inseriti in sequenza leggendoli da sinistra a destra e dall'alto in basso) che sono salvati nell'oggetto **cells**;
→ il vettore che contiene i nomi delle righe, salvato nell'oggetto **rnames**;
→ il vettore che contiene i nomi delle colonne, salvato nell'oggetto **cnames**.

I tre vettori sono combinati a formare la matrice dei dati mediante la funzione **matrix()** che impiega gli argomenti che indicano:
→ l'oggetto/vettore contenente i dati (**cells**);
→ il numero di righe (**nrow=2**) e il numero di colonne (**ncol=2**) della matrice che sarà pertanto di due righe per due colonne;
→ la modalità di riempimento della matrice, che deve essere riempita per righe (**byrow=TRUE**) quindi da sinistra a destra e dall'alto in basso;
→ i nomi da assegnare alle righe e alle colonne (**dimnames=list(rnames, cnames)**).

La matrice dei dati così costruita viene salvata nell'oggetto **mydata**, che viene infine mostrato per un controllo finale.

Come si vede la parte principale del lavoro consiste nella corretta immissione e strutturazione dei dati, mentre il calcolo vero e proprio del test viene effettuato in modo molto semplice, con la funzione **fisher.test()** [4].

Questi sono i risultati:

```
> mydata # mostra i dati

                         Denti_normali Malocclusione
Allattamento_naturale                4            16
Allattamento_artificiale             1            21
> fisher.test(mydata) # esegue il test di Fisher

        Fisher's Exact Test for Count Data

data:  mydata
p-value = 0.1745
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
   0.4440411 270.1636947
sample estimates:
odds ratio
  5.059936
```

Considerando significativi i valori inferiori al tradizionale valore soglia `p = 0.05` il test di Fisher fornisce nel nostro caso come risultato un valore di `p = 0.1745` che non è significativo: pertanto la conclusione è che non abbiamo evidenza statistica del fatto che i due tipi di allattamento comportino differenze nella prevalenza di malocclusione.

La funzione **fisher.test()** fornisce una analisi aggiuntiva della tabella calcolando anche l'**odds ratio**. In questo caso un risultato diverso da `1` deporrebbe per una associazione tra tipo di allattamento e tipo di occlusione dentaria. Un risultato diverso da 1 si avrebbe se i limiti di confidenza dell'odds ratio trovato, che è uguale a `5.059936`, non includessero il valore `1`. Dato che lo includono (vanno infatti da `0.4440411` a `270.1636947`) il risultato non è significativo, conferma la mancanza di associazione tra i fattori presenti nella tabella e conferma anche il valore di `p` ottenuto con il test di Fisher. Per il tema odds ratio si rimanda ai test di statistica: si trova tra gli altri in Marubini [5], Campbell [6] e Ingelfinger [7].

Abbiamo qui una tipica applicazione del buonsenso in statistica: con un numero totale di osservazioni superiore a 40 (sono in totale 42 osservazioni) avremmo dovuto a rigore applicare il test chi-quadrato, ma è stato applicato il test di Fisher in quanto è stato dato maggior peso al fatto che vi sono celle con un numero ridotto di osservazioni. A conferma di ciò potete eseguire il test chi-quadrato digitando

chisq.test(mydata)

con questo risultato:

```
> chisq.test(mydata)

        Pearson's Chi-squared test with Yates' continuity correction

data:  mydata
X-squared = 1.1398, df = 1, p-value = 0.2857

Messaggio di avvertimento:
In chisq.test(mydata) : Chi-squared approximation may be incorrect
```

Il test chi-quadrato indica una non significatività nelle differenze (`p = 0.2857`) ma un messaggio avverte che l'impiego del test chi-quadrato in questo caso non è appropriato. La ragione? Se digitate

chisq.test(mydata)$expected

ottenete la *tabella delle frequenze attese*

```
> chisq.test(mydata)$expected
                         Denti_normali Malocclusione
Allattamento_naturale         2.380952      17.61905
Allattamento_artificiale      2.619048      19.38095
Messaggio di avvertimento:
In chisq.test(mydata) : Chi-squared approximation may be incorrect
```

nella quale notate sulla sinistra due valori inferiori a 5. Il messaggio "`Chi-squared approximation may be incorrect`" viene fornito dalla funzione **chisq.test()** in quanto anche in **R** la regola "*frequenze attese inferiori a 5*" rappresenta il razionale per l'impiego del test di Fisher e prevale sul numero delle osservazioni.

In alternativa potete procedere come segue.

Copiate le tre righe riportate qui sotto, incollatele in un editor di file di testo aggiungendo un ↲ `Invio` al termine dell'ultima riga, e salvate il tutto in `C:\Rdati\` sotto forma di un file di testo denominato `chi_Fisher.csv` (di default i file di testo sono salvati con l'estensione `.txt` ma noi qui salviamo il file con l'estensione `.csv`) [8].

```
Allattamento;Denti_normali;Malocclusione
Allattamento_naturale;4;16
Allattamento_artificiale;1;21
```

Impiegate quest'altro script che prevede di eseguire il test di Fisher sui dati letti dal file `chi_Fisher.csv` precedentemente salvato. Copiatelo e incollatelo nella `Console di R` e premete ↲ `Invio`:

```
# TEST DI FISHER - 2 righe · 2 colonne
#
mydata <- read.table("C:/Rdati/chi_Fisher.csv", header=TRUE, sep=";",
row.names="Allattamento") # importa i dati
mydata # mostra i dati
fisher.test(mydata) # esegue il test di Fisher
#
```

Come vedete lo script è più compatto del precedente, le uniche cose da notare sono gli argomenti della funzione **read.table()**:
→ **"C:/Rdati/chi_Fisher.csv"** che specifica nome e posizione del file dal quale importare i dati;
→ **header=TRUE** che indica che nella prima riga del file sono contenuti i nomi delle variabili;
→ **sep=";"** che specifica il separatore di campo impiegato nel file;
→ **row.names="Allattamento"** che indica che i nomi delle righe sono contenuti nel campo "Allattamento".

Salvate entrambi gli script. Se preferite intervenire sullo script per adattarlo di volta in volta ai nuovi dati, potete impiegare il primo dei due. Se volete evitare di mettere mano ogni volta allo script e preferite intervenire sul file di dati, potete impiegare il secondo.

---

[1] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, MIlano, 1979, p.140.

[2] Snedecor GW, Cochran WG. *Statistical Methods*. The Iowa State University Press, 1980, ISBN 0-8138-1560-6, p. 127.

[3] Armitage P. *Statistica medica*. Giangiacomo Feltrinelli Editore, MIlano, 1979, pp. 138-140.

[4] Digitate **help(fisher.test)** nella `Console di R` per la documentazione della funzione **fisher.test()**.

[5] Bossi A, Cortinovis I, Duca PG, Marubini E. *Introduzione alla statistica medica*. La Nuova Italia Scientifica, Roma, 1994, ISBN 88-430-0284-8, pp. 282-285.

[6] Campbell MJ, Machin D. *Medical Statistics. A Commonsense Approach*. John Wiley & Sons, New York, 1993, ISBN 0-471-93764-9, pp. 121-123 e 154-156.

[7] Ingelfinger JA, Mosteller F, Thibodeau LA, Ware JH. *Biostatistica in medicina*. Macmillan Publishing Co., Inc.; New York, 1983, ISBN 88-7078-065.1, pp. 30-33.

[8] Per eventuali approfondimenti sul formato `.csv` (*comma separated value*) che è il formato dati raccomandato per **R** vedere i post nella sezione *R - gestione dati* della pagina Indice.

lunedì 3 ottobre 2022

Analisi dei gruppi (clustering non esclusivo)

L'obiettivo dell'**analisi dei gruppi** (*cluster analysis* o *clustering*) è concettualmente semplice: verificare la possibile esistenza, in un insieme di oggetti, di sottoinsiemi di oggetti particolarmente simili tra loro (gruppi/cluster).

L'analisi dei gruppi si applica a *dati multivariati* ed è un metodo statistico di tassonomia numerica che riveste un ruolo importante nella *analisi esplorativa dei dati*.

Nonostante alla base dell'analisi dei gruppi vi sia un'idea semplice e logica, vi sono numerosi modi per realizzarla [1], qui ci occupiamo dell'implementazione del *clustering non escusivo* con metodi *fuzzy* nelle due versioni:
→ clustering con il **metodo di Bezdek** (*fuzzy c-means*);
→ clustering con il **pacchetto 'fclust'** (*fuzzy clustering con algoritmi alternativi*).

Come dati impieghiamo i valori di BMI (indice di massa corporea) rilevati a livello europeo alcuni anni fa e pubblicati dall'Istat [2].

Per proseguire è necessario:
→ effettuare il download del file di dati `bmi.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7

Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2

Inoltre è necessario scaricare dal **CRAN** il pacchetto aggiuntivo **ppclust** [3] e il pacchetto aggiuntivo **fclust** [4].

Per capire meglio il *fuzzy clustering* dotiamoci di una tabella di riferimento costruita manualmente, nella quale è stata le Nazioni sono state raggruppate, secondo un minimo di criteri, in quattro gruppi:
→ in un primo gruppo (colore rosso) sono state riportate le Nazioni in cui prevalgono i soggetti con peso normale (oltre il 47%);
→ in un secondo gruppo (colore blu) sono state riportate le Nazioni caratterizzate dalla presenza di una percentuale elevata di soggetti obesi (oltre il 20%);
→ è stata evidenziata (colore verde) una Nazione che si differenzia nettamente dalla altre per la percentuale particolarmente elevata di soggetti con sovrappeso (46.4%) e particolarmente ridotta di obesi (9.4%);
→ infine sono state lasciate nello stesso gruppo (colore giallo) le rimanenti Nazioni, che hanno una percentuale di soggetti con peso normale inferiore al primo gruppo, una percentuale di obesi inferiore al secondo gruppo e una percentuale di sovrappeso mediamente elevata.

	A	B	C	D	E
1	Nazione	sottopeso	normale	sovrappeso	obeso
2	Danimarca	2,2	50,0	32,9	14,9
3	Austria	2,4	49,6	33,3	14,7
4	Francia	3,2	49,6	31,9	15,3
5	Lussemburgo	2,8	49,3	32,4	15,6
6	Olanda	1,6	49,0	36,0	13,3
7	Svezia	1,8	48,3	35,9	14,0
8	Belgio	2,7	48,0	35,3	14,0
9	Cipro	3,9	47,8	33,8	14,5
10	Malta	2,0	37,0	35,0	26,0
11	Lettonia	1,7	41,8	35,2	21,3
12	Ungheria	2,9	41,9	34,0	21,2
13	Estonia	2,2	43,9	33,5	20,4
14	Regno Unito	2,1	42,2	35,6	20,1
15	Romania	1,3	42,9	46,4	9,4
16	Grecia	1,9	41,3	39,4	17,3
17	Bulgaria	2,2	43,8	39,2	14,8
18	Croazia	1,9	40,7	38,7	18,7
19	Lituania	1,9	42,5	38,3	17,3
20	Slovacchia	2,1	43,6	38,0	16,3
21	Repubblica Ceca	1,1	42,1	37,6	19,3
22	Polonia	2,4	42,9	37,5	17,2
23	Slovenia	1,6	41,8	37,4	19,2
24	Irlanda	1,9	42,3	37,0	18,7
25	Portogallo	1,8	44,6	36,9	16,6
26	Finlandia	1,2	44,1	36,4	18,3
27	Spagna	2,2	45,4	35,7	16,7
28	Germania	1,8	46,1	35,2	16,9

Vediamo ora il fuzzy clustering con il metodo di Bezdek (*fuzzy c-means*).

Copiate questo script, incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# CLUSTERING NON ESCLUSIVO (FUZZY) con il metodo di Bezdek (fuzzy c-means)
#
library(ppclust) # carica il pacchetto
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
#
fuzzy <- fcm(mydata, centers=4) # calcola le percentuali di appartenenza dei dati ai 4 cluster
clunes <- cbind(fuzzy$u, fuzzy$cluster) # combina dati originali e cluster più rappresentativo
clunes[order(clunes[,5]),] # mostra la tabella ordinata per cluster
#
```

Con prima cosa viene caricato il pacchetto **ppclust**, quindi i dati sono importati in **mydata**.

Il fuzzy clustering viene effettuato impiegando la funzione **fcm()** (da fuzzy c-means), che ha come unici argomenti i dati (**mydata**) e il numero (**4**) dei cluster in cui suddividere i dati.

Dall'oggetto **fuzzy** che contiene i dati del fuzzy clustering sono estratti i dati originali (**fuzzy$u**) e, per ciascun dato, il cluster cui corrisponde la percentuale di appartenenza più elevata (**fuzzy$cluster**), che con la funzione **cbind()** sono combinati in un'unica tabella denominata **clunes** da clustering non esclusivo.

Con l'ultima riga di codice la tabella **clunes** viene ordinata per numero del cluster di appartenenza (colonna numero **[,5]**), con questo risultato:

```
> clunes[order(clunes[,5]),] # mostra la tabella ordinata per cluster
                 Cluster 1  Cluster 2   Cluster 3   Cluster 4
Estonia         0.667634277 0.11252359 0.149410982 0.070431153 1
Lettonia        0.958266772 0.01095656 0.026474715 0.004301955 1
Malta           0.491021069 0.15624358 0.264287496 0.088447851 1
Regno Unito     0.790353366 0.05102544 0.143114367 0.015506822 1
Ungheria        0.914138562 0.02579834 0.047401053 0.012662047 1
Bulgaria        0.047573779 0.73911039 0.159914511 0.053401319 2
Finlandia       0.174433173 0.38232277 0.382317053 0.060927001 2
Germania        0.131783981 0.39538759 0.171854699 0.300973735 2
Polonia         0.055772389 0.49224613 0.429183994 0.022797491 2
Portogallo      0.035927962 0.82921903 0.099909121 0.034943891 2
Romania         0.161443930 0.35907319 0.283259904 0.196222972 2
Slovacchia      0.006818789 0.95511365 0.033080423 0.004987139 2
Spagna          0.111429023 0.53373933 0.177822943 0.177008704 2
Croazia         0.085407339 0.10214244 0.795057675 0.017392545 3
Grecia          0.074338046 0.23144021 0.666957863 0.027263879 3
Irlanda         0.084041629 0.08650640 0.817102118 0.012349852 3
Lituania        0.042125460 0.28646014 0.655145014 0.016269381 3
Repubblica Ceca 0.090628501 0.07445451 0.822199171 0.012717815 3
Slovenia        0.078295256 0.05547535 0.856745593 0.009483801 3
Austria         0.008129413 0.01539587 0.008807597 0.967667119 4
Belgio          0.031551737 0.10068345 0.043130871 0.824633938 4
Cipro           0.030977486 0.06876469 0.035518430 0.864739392 4
Danimarca       0.020413065 0.03551855 0.021319185 0.922749201 4
Francia         0.042794189 0.06388769 0.040922110 0.852396008 4
Lussemburgo     0.031571890 0.04926516 0.030614865 0.888548086 4
Olanda          0.048915824 0.15009665 0.068931421 0.732056106 4
Svezia          0.045051649 0.15475022 0.064798888 0.735399244 4
```

Con un grado di appartenenza che va dal **73.2%** al **96.7%** le Nazioni che il fuzzy clustering codifica come prevalentemente appartenenti al cluster **4** trovano corripondenza nella tabella di riferimento riportata sopra:

```
                 Cluster 1  Cluster 2   Cluster 3   Cluster 4
Austria         0.008129413 0.01539587 0.008807597 0.967667119 4
Belgio          0.031551737 0.10068345 0.043130871 0.824633938 4
Cipro           0.030977486 0.06876469 0.035518430 0.864739392 4
Danimarca       0.020413065 0.03551855 0.021319185 0.922749201 4
Francia         0.042794189 0.06388769 0.040922110 0.852396008 4
Lussemburgo     0.031571890 0.04926516 0.030614865 0.888548086 4
Olanda          0.048915824 0.15009665 0.068931421 0.732056106 4
```

```
Svezia          0.045051649 0.15475022 0.064798888 0.735399244 4
```

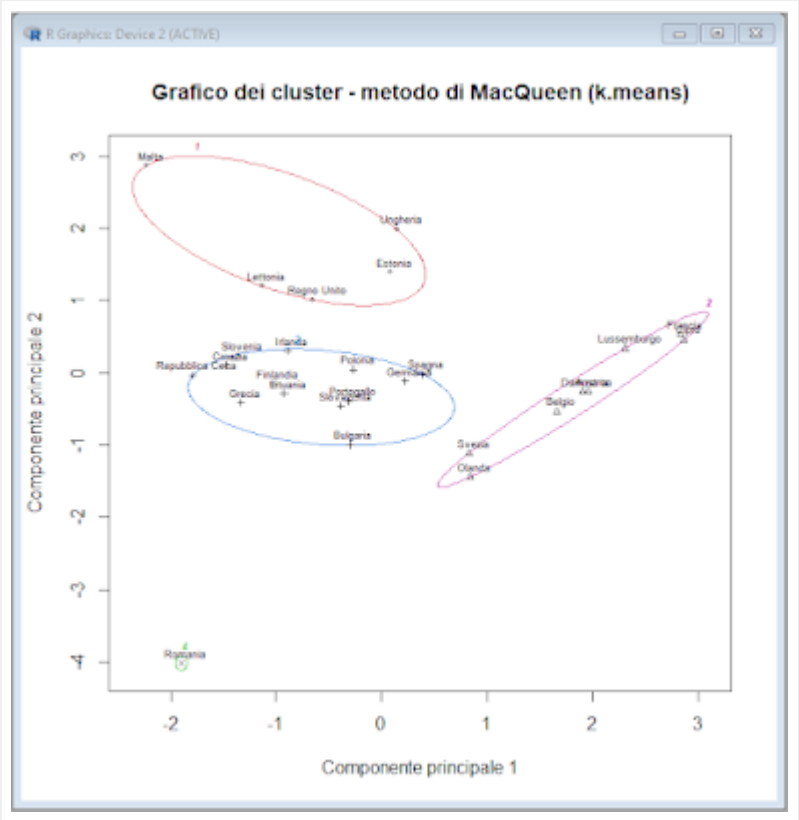
Una corrispondenza con la tabella di riferimento costruita manualmente la abbiamo anche per le Nazioni che il fuzzy clustering codifica come prevalentemente appartenenti al cluster `1` con percentuali che vanno dal `49.1%` al `95.8%`:

```
                Cluster 1  Cluster 2   Cluster 3   Cluster 4
Estonia       0.667634277 0.11252359 0.149410982 0.070431153 1
Lettonia      0.958266772 0.01095656 0.026474715 0.004301955 1
Malta         0.491021069 0.15624358 0.264287496 0.088447851 1
Regno Unito   0.790353366 0.05102544 0.143114367 0.015506822 1
Ungheria      0.914138562 0.02579834 0.047401053 0.012662047 1
```

I due rimananenti cluster raggruppano tutte le altra Nazioni: ma a questo punto potrebbe essere opinabile se sia meglio riportare in un unico cluster separato la Romania, oppure se sia preferibile decomporne l'appartenenza tra tutti i quattro cluster, come prevede il fuzzy clustering.

Da notare che quando si riesegue lo script la numerazione dei cluster può cambiare, senza tuttavia che cambino i valori percentuali di appartenenza ai vari cluster e il raggruppamento in base alla percentuale di appartenenza più elevata.

Un aiuto ulteriore nella valutazione dei risultati del *fuzzy clustering* ci può venire dal grafico del *clustering non gerarchico* con il metodo di MacQueen delle *k-means*, che mostra la Romania isolata nell'angolo inferiore sinistro del grafico: ma bisogna ricordare che si tratta di strategie di clustering differenti, e quindi l'interpretazione dei risultati deve essere condotta tenendo ben presente questo fatto.



Vediamo ora il clustering con il pacchetto 'fclust' (*fuzzy clustering con algoritmi alternativi*).

```
# CLUSTERING NON ESCLUSIVO (FUZZY) con il pacchetto "fclust"
#
library(fclust) # carica il pacchetto
```

mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";", row.names="Nazione") # importa i dati
#
fuzzy <- Fclust(mydata, k=4) # calcola le percentuali di appartenenza dei dati a 4 cluster
clunes <- cbind(fuzzy$U, fuzzy$clus[,1]) # unisce dati originali e cluster più rappresentativo
clunes[order(clunes[,5]),] # mostra la tabella ordinata per cluster
#

Rispetto allo script precedente qui di diverso ci sono solamente il pacchetto **fclust**, e la funzione impiegata per calcolare le percentuali di appartenenza dei dati a 4 cluster, che ora è la funzione **Fclust()** mentre il resto è identico.

```
> clunes[order(clunes[,5]),] # mostra la tabella ordinata per cluster
                   Clus 1      Clus 2      Clus 3      Clus 4
Bulgaria        0.73911039 0.159914512 0.053401319 0.047573779 1
Finlandia       0.38232277 0.382317052 0.060927001 0.174433172 1
Germania        0.39538759 0.171854699 0.300973734 0.131783981 1
Polonia         0.49224613 0.429183994 0.022797491 0.055772389 1
Portogallo      0.82921903 0.099909121 0.034943891 0.035927961 1
Romania         0.35907319 0.283259904 0.196222972 0.161443930 1
Slovacchia      0.95511365 0.033080424 0.004987140 0.006818789 1
Spagna          0.53373933 0.177822942 0.177008704 0.111429022 1
Croazia         0.10214244 0.795057676 0.017392545 0.085407339 2
Grecia          0.23144021 0.666957864 0.027263879 0.074338046 2
Irlanda         0.08650640 0.817102117 0.012349852 0.084041629 2
Lituania        0.28646014 0.655145015 0.016269381 0.042125460 2
Repubblica Ceca 0.07445451 0.822199170 0.012717815 0.090628501 2
Slovenia        0.05547535 0.856745592 0.009483801 0.078295256 2
Austria         0.01539587 0.008807597 0.967667119 0.008129413 3
Belgio          0.10068345 0.043130871 0.824633938 0.031551737 3
Cipro           0.06876469 0.035518430 0.864739392 0.030977486 3
Danimarca       0.03551855 0.021319185 0.922749201 0.020413065 3
Francia         0.06388769 0.040922110 0.852396008 0.042794189 3
Lussemburgo     0.04926516 0.030614865 0.888548086 0.031571890 3
Olanda          0.15009665 0.068931421 0.732056105 0.048915824 3
Svezia          0.15475022 0.064798888 0.735399244 0.045051649 3
Estonia         0.11252359 0.149410982 0.070431153 0.667634277 4
Lettonia        0.01095656 0.026474715 0.004301955 0.958266772 4
Malta           0.15624358 0.264287496 0.088447851 0.491021069 4
Regno Unito     0.05102544 0.143114367 0.015506822 0.790353366 4
Ungheria        0.02579834 0.047401053 0.012662047 0.914138562 4
```

Da notare che di default la funzione **Fclust()** impiega lo stesso algoritmo della funzione **fcm()** dello script precedente, quindi vi sarà facile constatare che i risultati sono identici ai precedenti. Per gli algoritmi alternativi che è possibile impiegare si rimanda al manuale del pacchetto [4].

Conclusione: l'analisi dei gruppi con il clustering non esclusivo (fuzzy) non è commisurabile con quella fornita dai metodi esclusivi, in quanto per definizione decompone l'appartenenza di un oggetto in frazioni percentuali, tante quanti sono i cluster in cui si intende classificare i dati. Tuttavia è anche riconducibile ai metodi di clustering esclusivo quando si considera il cluster al quale l'oggetto appartiene maggiormente in percentuale. E questo duplice punto di vista sui dati, anziché fonte di perplessità, deve essere considerato una informazione utile nella fase di interpretazione dei risultati degli altri tipi di clustering.

Trovate il seguito e le strategie alternative di clustering e di analisi dei dati multivariati nei post:
→ Analisi delle componenti principali

→ Analisi dei gruppi (clustering gerarchico)
→ Analisi dei gruppi (clustering non gerarchico)

---

[1] Nonostante alla base dell'analisi dei gruppi vi sia un'idea semplice e logica, vi sono numerosi modi per realizzarla, infatti esistono:
→ *metodi gerarchici*, che danno luogo a una suddivisione ad albero (dendrogramma) in base alla distanza tra i singoli oggetti dell'insieme;
→ *metodi non gerarchici*, nei quali l'appartenenza di un oggetto (dell'insieme) ad uno specifico sottoinsieme/gruppo/cluster viene stabilita sulla base della sua distanza dal centro o dalla media dei dati o da un punto rappresentativo del cluster;
→ metodi *bottom-up* noti anche come *metodi agglomerativi* nei quali all'inizio del processo di classificazione ad ogni oggetto viene fatto corrispondere un cluster. In questo stadio gli oggetti sono considerati tutti dissimili tra di loro. Al passaggio successivo i due oggetti più simili sono raggruppati nello stesso cluster. Il numero dei cluster risulta quindi pari al numero di oggetti diminuito di uno. Il procedimento viene ripetuto ciclicamente, fino ad ottenere (all'ultimo passaggio) un unico cluster;
→ metodi *top-down* noti anche come *metodi divisivi* nei quali inizialmente tutti gli oggetti sono considerati come appartenenti ad un unico cluster, che viene via via suddiviso in cluster fino ad avere un numero di cluster uguale al numero degli oggetti;
→ *metodi esclusivi*, che prevedono che un oggetto possa appartenere esclusivamente a un cluster;
→ metodi *non esclusivi* (*fuzzy*) che prevedono che un oggetto possa appartenere, in modo quantitativamente diverso, a più di un cluster.

[2] Vedere il post Indice di massa corporea (BMI).

[3] Trovate la documentazione nel manuale di riferimento del pacchetto: Package ‘ppclust’. URL consultato il 06/01/2023.

[4] Trovate la documentazione nel manuale di riferimento del pacchetto: Package ‘fclust’. URL consultato il 06/01/2023.

sabato 15 ottobre 2022

Istogrammi

Per la rappresentazione grafica della distribuzione di una variabile, in alternativa ai kernel density plot possiamo impiegare i più tradizionali istogrammi.

È arcinoto che gli **istogrammi** sono realizzati raggruppando i valori espressi in una *scala numerica continua*, riportati in ascisse, in un certo numero di sottoinsiemi (**classi**) e riportando in ordinate la *frequenza* dei valori (espressa come *numero* dei casi, come *percentuale* dei casi o come *probabilità)* che cadono in ciascuna classe.

Forse meno noto è che trattandosi della rappresentazione di dati espressi in una scala numerica continua, negli istogrammi le barre che rappresentano le classi devono essere unite tra loro. Questo aspetto li differenzia dai **grafici a barre**, che sono impiegati per rappresentare dati espressi in forma di *scale nominali*, *scale ordinali* e *scale numeriche discrete*, che non sono scale continue e per questa ragione devono impiegare barre staccate [1].

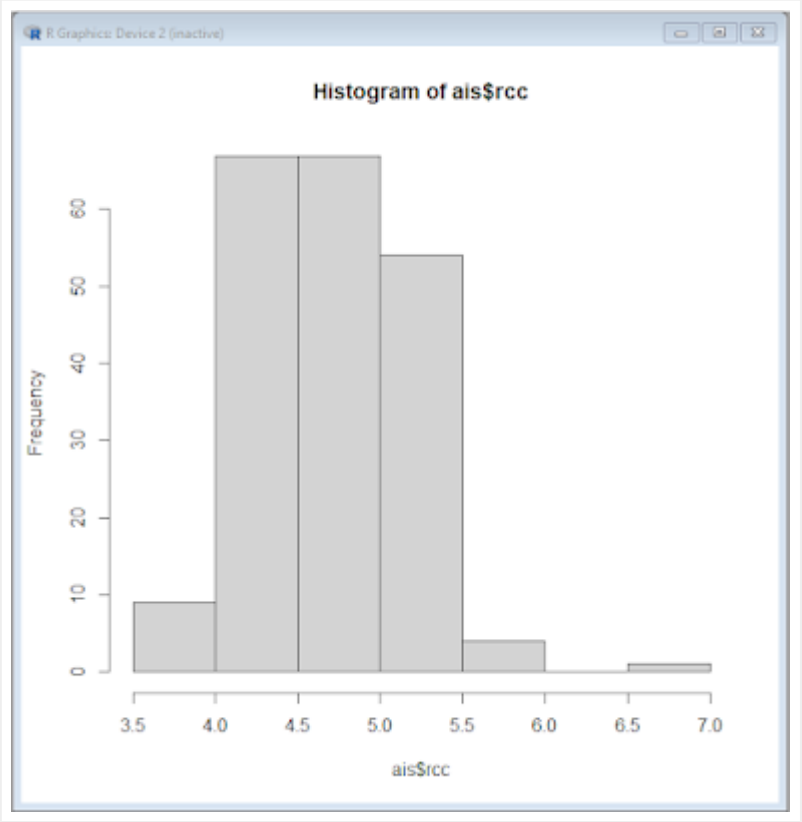
Vediamo ora una serie di esempi di istogrammi, interessanti in quanto realizzati senza la necessità di installare pacchetti aggiuntivi ma impiegando solamente le *funzioni statistiche e grafiche di base* di **R**.

Tutti gli istogrammi mostrano la distribuzione della concentrazione degli eritrociti (globuli rossi) nel sangue rilevata in 202 atleti australiani riportata nella colonna/variabile **rcc** della tabella **ais** inclusa nel pacchetto **DAAG**. Potete scaricare il pacchetto dal **CRAN** oppure acquisire la tabella seguendo le indicazioni alternative riportate in [2].

Gli esempi sono stati riportati in script separati, in modo che quelli che più interessano possano essere salvati indipendentemente l'uno dall'altro ed essere successivamente ripresi per analizzare i propri dati semplicemente eliminando la prima riga di codice, sostituendo **ais$rcc** con il nome della propria variabile e adattando opportunamente titoli ed etichette.

Copiate e incollate il primo script nella `Console di R` e premete `↵ Invio`:

```
# ISTOGRAMMA SEMPLICE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
hist(ais$rcc) # traccia l'istogramma
#
```

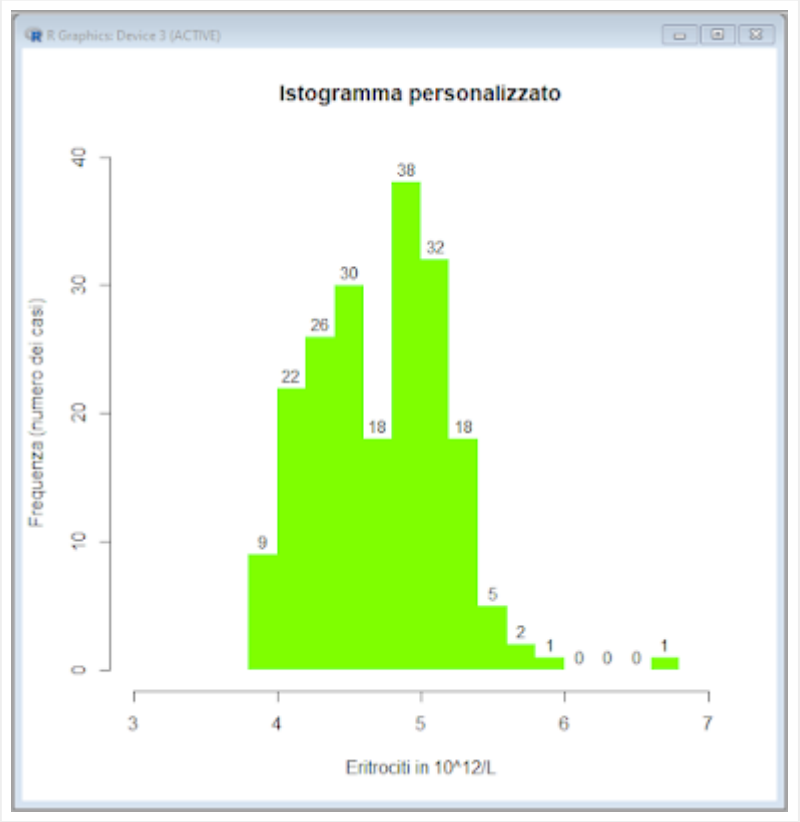


Questo è il codice più conciso con cui è possibile realizzare un istogramma con **R**: prevede di impiegare nella funzione **hist()**, come unico argomento, l'indicazione dei dati da rappresentare, nel nostro caso **ais$rcc** ovvero i valori della colonna/variabile **rcc** della tabella **ais**. Si tratta di una soluzione minima, che impiega i valori di default previsti per la funzione, e che può essere utile quando si voglia effettuare senza fatica una rapida analisi preliminare dei dati.

Nota bene: nella funzione **hist()** il valore di default per l'argomento che calcola il numero di classi è **breaks="Sturges"** che qui porta alla suddivisione dei dati in sole 7 classi. Nei prossimi script vediamo come sia possibile in alternativa:
→ specificare **breaks="Scott"** oppure **breaks="FD"** (scelte consigliate);
→ calcolare il numero di classi con **sqrt(length(***nomedellavariabile***))** cioè come radice quadrata **sqrt()** del numero di dati **length(***nomedellavariabile***)**, soluzione meno sofisticata delle precedenti ma tendenzialmente adeguata;
→ imporre come numero di classi un valore numerico arbitrario (scelta sconsigliata a meno di avere preventivamente stabilito per altra via il numero opportuno delle classi).

Ora copiate e incollate questo secondo script nella `Console di R` e premete ↵ `Invio`:

```
# ISTOGRAMMA PERSONALIZZATO
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
istog <- hist(ais$rcc, breaks=15, xlim=c(3,7), ylim=c(0,40), col="chartreuse",
border=FALSE, main="Istogramma personalizzato", xlab="Eritrociti in 10^12/L",
ylab="Frequenza (numero dei casi)") # traccia un istogramma personalizzato
text(cex=0.7, istog$mids, istog$counts, labels=istog$counts, adj=c(0.5, -0.5)) # aggiunge
il numero dei casi per ciascuna classe
#
```



Se nella `Console di R` digitate **help(hist)** trovate l'elenco completo degli argomenti che potete impiegare per personalizzare un istogramma e dei quali qui vediamo quelli essenziali:
→ **breaks=15** per fissare a **15** il numero delle classi (vedere nota allo script precedente);
→ **xlim=c(3,7)** per fissare limite inferiore e limite superiore dell'asse delle ascisse;
→ **ylim=c(0,40)** per fissare limite inferiore e limite superiore dell'asse delle ordinate;
→ **col="chartreuse"** per colorare l'istogramma con uno dei moltissimi colori della tavolozza di **R** [3];
→ **border=FALSE** per eliminare il bordo che delimita ciascuna classe;
→ **main="..."** per riportare il titolo del grafico;
→ **xlab="..."** e **ylab="..."** per riportare le legende degli assi.

L'istogramma viene sia tracciato sia salvato nell'oggetto **istog**. Se nella `Console di R` digitate

str(istog)

potete visualizzare la struttura dell'oggetto **istog**

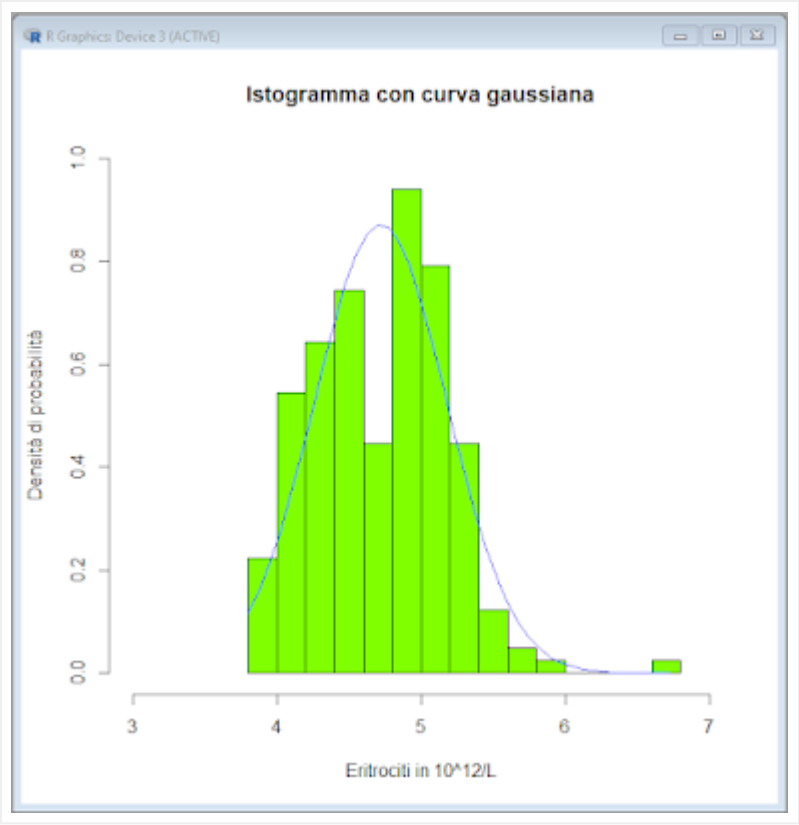
```
> str(istog)
List of 6
 $ breaks  : num [1:16] 3.8 4 4.2 4.4 4.6 4.8 5 5.2 5.4 5.6 ...
 $ counts  : int [1:15] 9 22 26 30 18 38 32 18 5 2 ...
 $ density : num [1:15] 0.223 0.545 0.644 0.743 0.446 ...
 $ mids    : num [1:15] 3.9 4.1 4.3 4.5 4.7 4.9 5.1 5.3 5.5 5.7 ...
 $ xname   : chr "ais$rcc"
 $ equidist: logi TRUE
 - attr(*, "class")= chr "histogram"
```

che contiene le informazioni che ci servono per poter riportare in testa a ciascuna classe il numero di dati in essa contenuti impiegando la funzione **text()** dell'ultima riga di codice. Nella `Console di R` digitate **help(text)** per una spiegazione completa degli argomenti impiegati per ottenere questo risultato.

Nota bene: rispetto all'istogramma precedente, che lasciando il valore di default per l'argomento **breaks** era suddiviso in 7 classi e mostrava una distribuzione *unimodale*, la suddivisione dell'istogramma in 15 classi mostra ora una *distribuzione bimodale*, che tipicamente si osserva quando una variabile/serie di misure è in realtà composta da due sottoinsiemi aventi mode/medie differenti.

Se volete sovrapporre a un istogramma la curva gaussiana corrispondente copiate e incollate nella `Console di R` questo terzo script e premete `↲ Invio`:

```
# ISTOGRAMMA CON LA CURVA GAUSSIANA CORRISPONDENTE e ordinate in comune
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
hist(ais$rcc, xlim=c(3,7), ylim=c(0,1), freq=FALSE, breaks="FD", col="chartreuse",
main="Istogramma con curva gaussiana", xlab="Eritrociti in 10^12/L", ylab="Densità di
probabilità") # traccia l'istogramma
x <- seq(min(ais$rcc), max(ais$rcc), length.out=100) # calcola ascisse della curva gaussiana
y <- dnorm(x, mean=mean(ais$rcc), sd=sd(ais$rcc)) # calcola ordinate della curva gaussiana
lines(x, y, col="blue", lwd=1) # sovrappone all'istogramma la curva gaussiana
#
```



In questo script nella funzione **hist()** che traccia l'istogramma compaiono due soli argomenti nuovi:
→ **freq=FALSE** che consente di riportare sull'asse delle ordinate, invece del numero di dati come è accaduto nei due script precedenti, la densità di probabilità, per poter successivamente sovrapporre all'istogramma la curva gaussiana le cui ordinate sono espresse nella stessa scala;
→ **breaks="FD"** con il quale il numero delle classi viene calcolato impiegando la regola di Freedman-Diaconis [4], che consente di minimizzare la differenza tra l'area dell'istogramma e l'area della gaussiana corrispondente e porta alla suddivisione dell'istogramma in 15 classi.

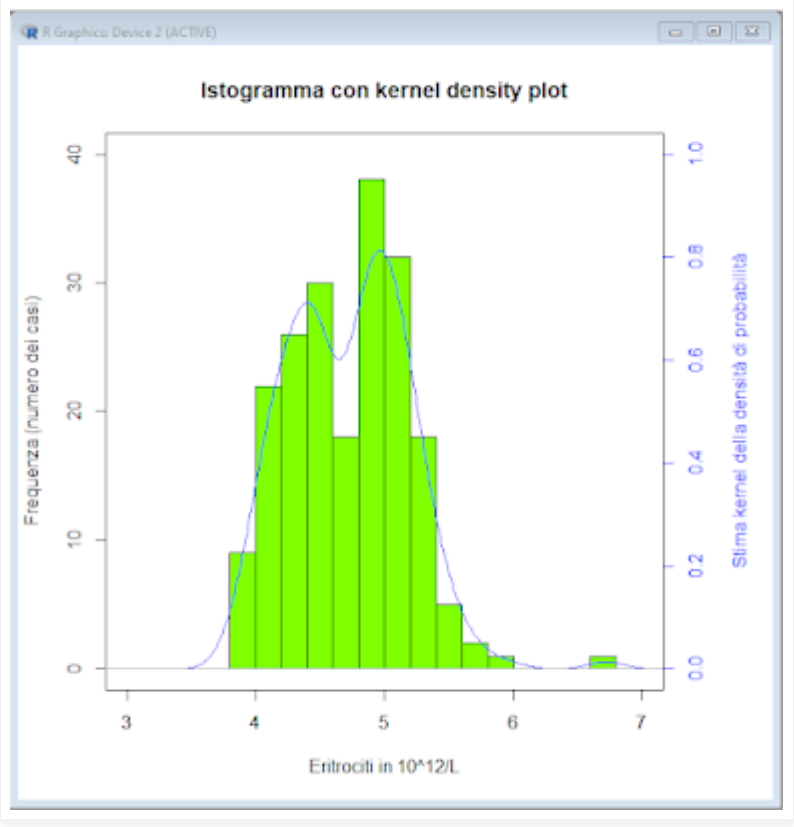
La curva gaussiana teorica corrispondente ai dati osservati viene calcolata in tre passaggi:
→ con la funzione **seq()** sono calcolati 100 (**length.out=100**) valori **x** delle ascisse compresi tra il valore minimo **min(ais$rcc)** e il valore massimo **max(ais$rcc)** della variabile rappresentata - il

numero di valori calcolati è stato scelto in base alla risoluzione grafica ma può ovviamente essere aumentato al bisogno;
→ con la funzione **dnorm()** sono calcolati i valori in ordinate **y** corrispondenti ai 40 valori **x** precedenti e aventi la media **mean=mean(ais$rcc)** e la deviazione standard **sd=sd(ais$rcc)** della variabile analizzata;
→ con la funzione **lines()** viene sovrapposta all'istogramma la curva gaussiana passante per i 100 punti di coordinate **x,y** così calcolati.

Nota bene: risulta evidente come la curva gaussiana, per definizione *unimodale*, mal si adatta alla distribuzione bimodale dei dati osservabile nell'istogramma.

Ora copiate e incollate nella `Console di R` questo quarto script e premete `↵ Invio`:

```
# ISTOGRAMMA CON IL KERNEL DENSITY PLOT CORRISPONDENTE e ordinate separate
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
par(mar=c(5.1,4.1,4.1,5.1)) # aumenta il margine destro da 2.1 a 5.1
hist(ais$rcc, xlim=c(3,7), ylim=c(0,40), freq=TRUE, col="chartreuse", breaks="Scott",
main="Istogramma con kernel density plot", xlab="Eritrociti in 10^12/L",
ylab="Frequenza (numero dei casi)") # traccia l'istogramma
#
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
plot(density(ais$rcc), yaxt="n", xlim=c(3,7), ylim=c(0,1), col="blue", lwd=1) #
sovrappone all'istogramma il kernel density plot
#
axis(4, col.ticks="blue", col.axis="blue") # riporta sulla destra l'asse delle y del kernel density
plot
mtext("Stima kernel della densità di probabilità", side=4, line=3, cex=1, col="blue") #
aggiunge la legenda all'asse delle y sulla destra
par(mar=c(5.1,4.1,4.1,2.1)) # ripristina i margini di default
#
```

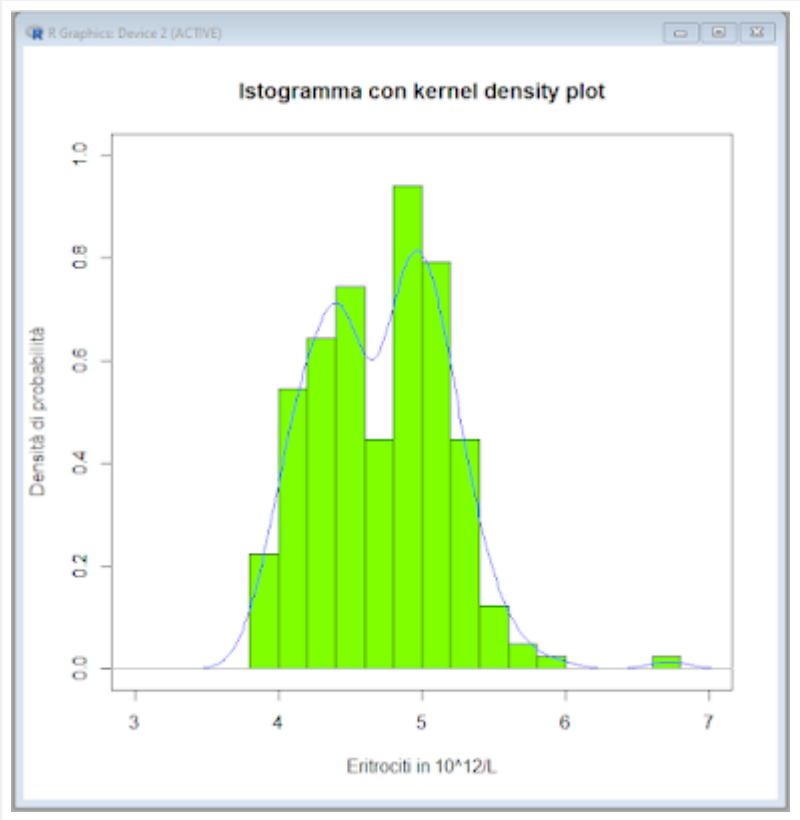


In questo caso l'esercizio consiste nel sovrapporre istogramma e kernel density plot riportando due assi delle ordinate separati e con scale diverse [5]:
→ l'asse delle `y` sulla sinistra riferito all'istogramma e che riporta la frequenza espressa come numero dei casi;
→ l'asse delle `y` sulla destra riferito al kernel density plot e che riporta la stima kernel della densità di probabilità.

Ecco le principali osservazioni sugli argomenti e sulle funzioni impiegate in questo script:
→ come prima cosa il margine sulla destra del grafico viene aumentato dal valore di default **2.1** a **5.1** per creare lo spazio necessario per la legenda dell'asse delle `y` del kernel density plot, posizionato sulla destra;
→ nella funzione **hist()** viene riportato in ordinate il numero dei casi ponendo **freq=TRUE**;
→ il numero delle classi viene calcolato con l'argomento **breaks="Scott"** che fornisce un risultato (15 classi) uguale a quello ottenuto con la regola di Freedman-Diaconis;
→ la funzione **par(new=TRUE, ann=FALSE)** specifica che il grafico successivo verrà sovrapposto (**new=TRUE**) ma che non devono essere annotati automaticamente titolo e legende (**ann=FALSE**), perchè questo lo faremo con le righe successive;
→ con la funzione **plot()** il kernel density plot viene sovrapposto all'istogramma specificando con **yaxt="n"** che non deve essere riportato l'asse delle `y` che si sovrapporrebbe a quello dell'istogramma e che invece noi aggiungeremo ora sulla destra;
→ con la funzione **axis()** riportiamo l'asse delle `y` del kernel density plot nella posizione **4** cioè sulla destra (nella grafica di base di **R** i margini sono 1=margine inferiore, 2=margine sinistro, 3=margine superiore, 4=margine destro) e riportiamo in colore **"blue"** sia le tacche (**col.ticks**) sia i corrispondenti valori numerici della scala (**col.axis**);
→ con la funzione **mtext()** aggiungiamo sulla destra (**side=4**) la legenda dell'asse delle `y` del kernel density plot, posizioniamo l'etichetta **"Stima kernel della densità di probabilità"** sulla **line=3** (la linea 0 contiene le tacche, la 1 contiene i valori numerici, la 2 è volutamente lasciata vuota), riportiamo i caratteri nella dimensione standard (**cex=1**) e in colore blu (**col="blue"**);
→ con la funzione **par()** ripristiniamo i margini di default, di fatto riportiamo quello di destra (il quarto valore dell'argomento **mar**) da **5.1** a **2.1**.

Se preferite impiegare un'unica scala sia per l'istogramma sia per il kernel density plot, esprimendo per entrambi le ordinate in termini di *densità di probabilità*, copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

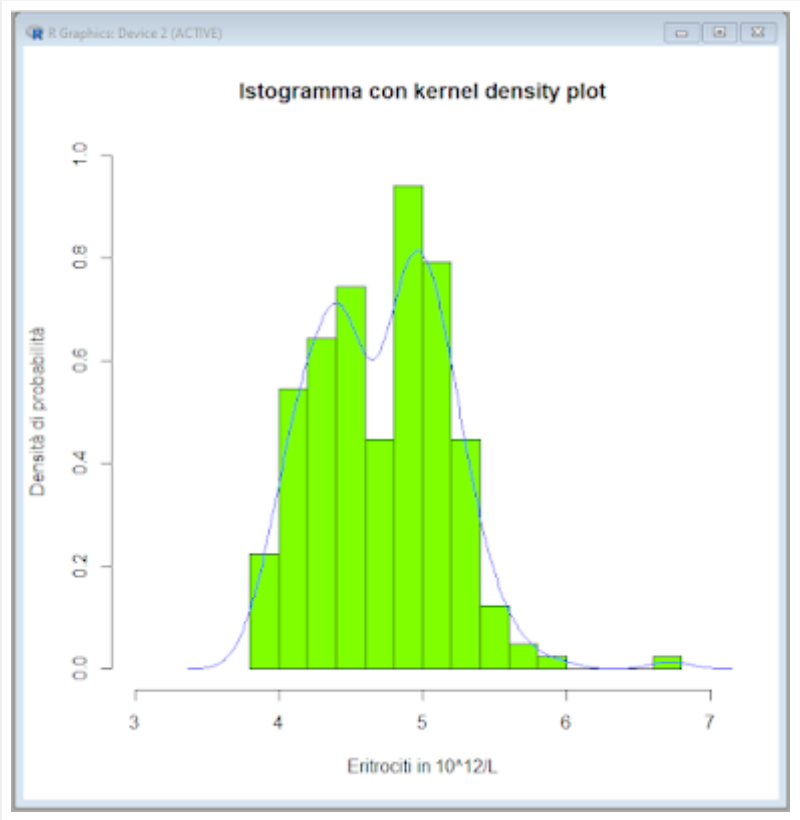
```
# ISTOGRAMMA CON IL KERNEL DENSITY PLOT CORRISPONDENTE e ordinate in comune
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
hist(ais$rcc, xlim=c(3,7), ylim=c(0,1), freq=FALSE, breaks="FD", col="chartreuse",
main="Istogramma con kernel density plot", xlab="Eritrociti in 10^12/L", ylab="Densità
di probabilità") # traccia l'istogramma
#
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
plot(density(ais$rcc), xlim=c(3,7), ylim=c(0,1), col="blue", lwd=1) # sovrappone
all'istogramma il kernel density plot
#
```



il codice risulterà semplificato in quanto sono sufficienti tre sole righe con:
→ la funzione **hist()** per tracciare l'istogramma;
→ la funzione **par()** per consentire la sovrapposizione del grafico successivo;
→ la funzione **plot()** per tracciare il kernel density plot.

Se vi accontentate di un'estetica un poco più spartana potete anche ridurre il codice ai minimi termini, copiate e incollate nella `Console di R` questo script e premete ↵ `Invio`:

```
# ISTOGRAMMA CON IL KERNEL DENSITY PLOT CORRISPONDENTE e ordinate in comune
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
hist(ais$rcc, xlim=c(3,7), ylim=c(0,1), freq=FALSE, breaks="FD", col="chartreuse",
main="Istogramma con kernel density plot", xlab="Eritrociti in 10^12/L", ylab="Densità
di probabilità") # traccia l'istogramma
#
lines(density(ais$rcc), xlim=c(3,7), ylim=c(0,1), col="blue", lwd=1) # sovrappone
all'istogramma il kernel density plot
#
```

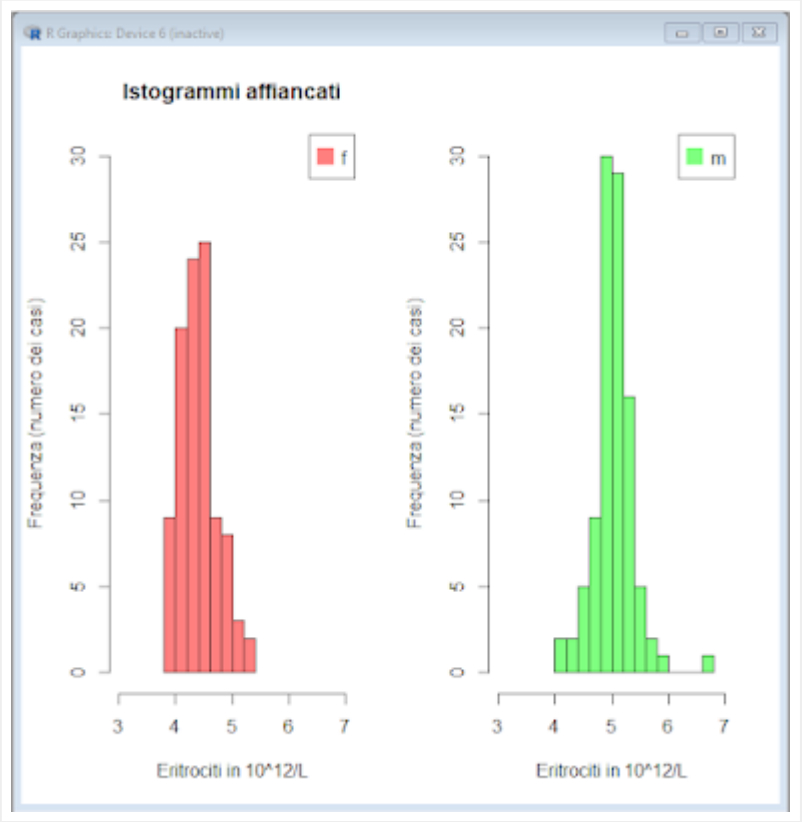


Ora sono sufficienti due sole righe di codice con:
→ la funzione **hist()** per tracciare l'istogramma;
→ la funzione **lines()** per sovrapporre il kernel density plot.

Nota bene: diversamente dalla distribuzione gaussiana, il kernel density plot, proprio per il modo in cui viene generato [6], descrive bene la distribuzione bimodale dei dati osservabile nell'istogramma.

Per affiancare due istogrammi è sufficiente disporli in due colonne affiancate, copiate e incollate questo script nella `Console di R` e premete ↵ `Invio`:

```
# DUE ISTOGRAMMI AFFIANCATI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
par(mfrow=c(1,2)) # grafici organizzati in una riga e due colonne
#
hist(f, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), col=rgb(1,0,0,0.5), main="Istogrammi affiancati", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di f
legend("topright", legend=c("f"), col=c(rgb(1,0,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
hist(m, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(m)), col=rgb(0,1,0,0.5), main="", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di m
legend("topright", legend=c("m"), col=c(rgb(0,1,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
#
```



Nel set di dati ais la concentrazione dei globuli rossi nel sangue è stata determinata in 202 atleti, 100 donne e 102 uomini ed è noto che nel sesso femminile essa è mediamente inferiore a quella del sesso maschile: questa differenza dovrebbe giustificare la distribuzione bimodale osservata nei tre istogrammi precedenti. Per effettuare una riprova in questo e nei due script che seguono viene utilizzato il fattore sesso (**sex**) della tabella **ais** per separare i valori riscontrati nelle donne (**f**) da quelli riscontrati negli uomini (**m**) e rappresentare separatamente gli istogrammi di questi due sottoinsiemi. Impieghiamo la funzione **subset()** per estrarre con **select=c(rcc)** la concentrazione dei globuli rossi prima per le donne con **sex=="f"** poi per gli uomini con **sex=="m"** e salvarla separatamente in **f** e in **m**.

Gli istogrammi sono generati con la funzione **hist()** i cui argomenti di base sono già noti, mentre qui si può notare in aggiunta che:
→ gli istogrammi sono disposti con **par(mfrow=c(1,2)** in una riga e due colonne;
→ il numero delle classi **breaks=sqrt(length(f))** è calcolato come radice quadrata del numero dei dati, il metodo di calcolo tradizionale e più semplice;
→ gli istogrammi prevedono l'impiego di un colore specificato con la funzione **rgb(1,0,0,0.5)** nella quale il primo numero indica il rosso (red), il secondo il verde (green), il terzo il blu (blu) e il quarto il grado di trasparenza del colore impiegato in una scala che va da **0** a **1**;
→ la legenda è posizionata in alto a destra (**"topright"**) e il colore del relativo istogramma viene riportato in un quadrato (**pch=15**) le cui dimensioni relative sono specificate come **pt.cex=2**.

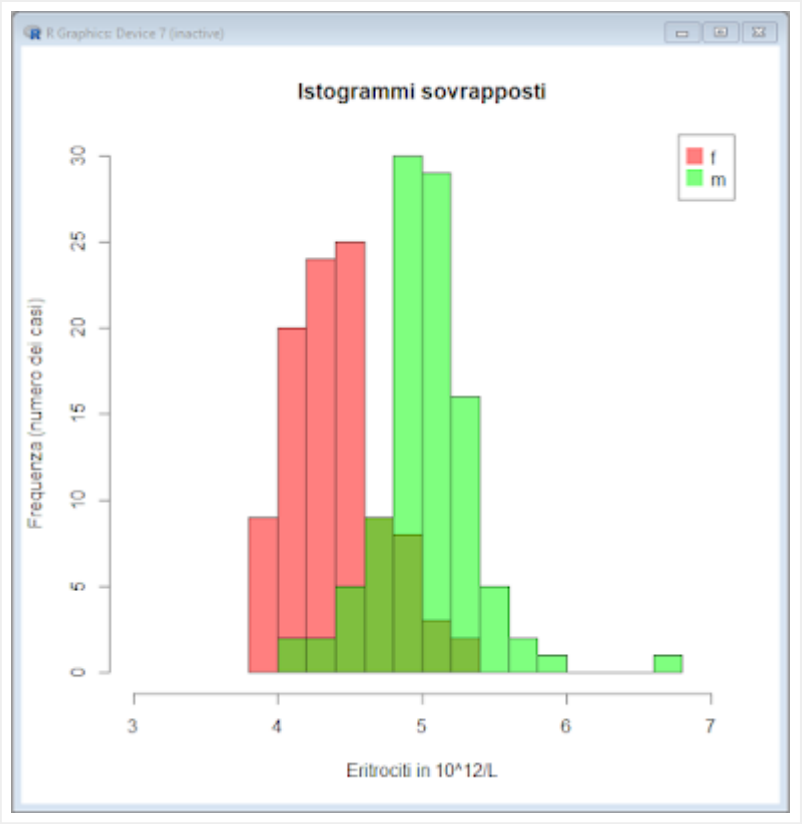
Nota bene: se osserviamo attentamente il lato sinistro dell'istogramma della distribuzione dei globuli rossi nel sesso femminile, istogramma che è chiaramente unimodale, e il lato destro dell'istogramma nel sesso maschile, anch'esso unimodale, possiamo riconoscere il lato sinistro e il lato destro dell'istogramma che mostra la distribuzione bimodale dei dati globali negli script precedenti.

Ora il salto di qualità nella rappresentazione consiste nel sovrapporre i due precedenti istogrammi unimodali dei dati separati per sesso tracciando due istogrammi sovrapposti e visibili nella loro interezza, senza che uno nasconda l'altro.

Copiate e incollate questo script nella `Console di R` e premete ↵ `Invio`:

```
# DUE ISTOGRAMMI SOVRAPPOSTI
```

#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
#
hist(f, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), col=rgb(1,0,0,0.5), main="Istogrammi sovrapposti", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di f
hist(m, add=TRUE, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(m)), col=rgb(0,1,0,0.5),) # istogramma di m
legend("topright", legend=c("f","m"), col=c(rgb(1,0,0,0.5), rgb(0,1,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
#

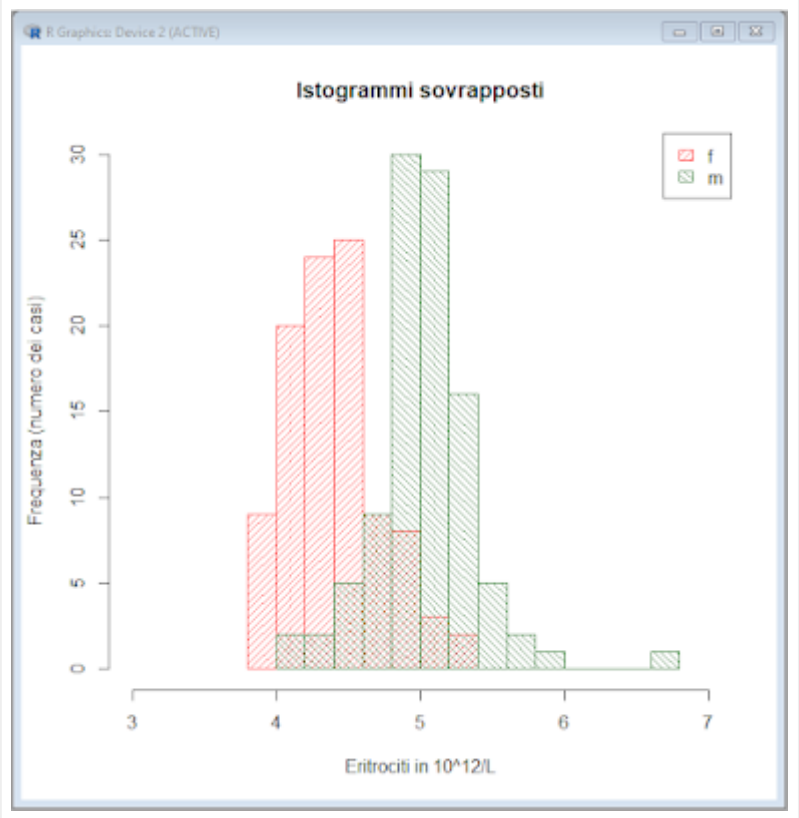


Qui l'unica novità è l'argomento **add=TRUE** che consente di sovrapporre il secondo istogramma al primo, facendo si che entrambi gli istogrammi siano visibili nella loro interezza grazie alla trasparenza del colore assicurata con il quarto valore della funzione **rgb()** che viene posto uguale a **0.5**.

Potete anche considerare per la sovrapposizione di due istogrammi questa estetica alternativa, copiate e incollate questo script nella `Console di R` e premete ↵ `Invio`:

DUE ISTOGRAMMI SOVRAPPOSTI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f

m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
#
hist(f, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), border="red", col="red", density=20, angle=45, main="Istogrammi sovrapposti", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di f
hist(m, add=TRUE, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(m)), border="green4", col="green4", density=20, angle=-45) # istogramma di m
legend("topright", legend=c("f","m"), fill=c("red","green4"), border=c("red","green4"), col=c("red","green4"), density=c(20,20), angle=c(45,-45)) # riporta la legenda
#



Gli argomenti aggiuntivi che è necessario impiegare nella funzione **hist()** per realizzare questo tipo di trasparenza sono:
→ **border=...** per specificare il colore del bordo che delimita le barre del primo **c("red",...)** e del secondo **c(...,"green4")** istogramma;
→ **col=...** per il colore da applicare al tratteggio definito con i due argomenti che seguono;
→ **density=...** per la densità del tratteggio interno alle barre dei due istogrammi;
→ **angle=...** per l'angolo di pendenza del tratteggio che con un valore positivo dell'angolo (**45**) nel primo istogramma sarà inclinato da sinistra a destra dal basso in alto, e con un valore negativo dell'angolo (**-45**) nel secondo istogramma sarà inclinato da sinistra a destra dall'alto in basso.

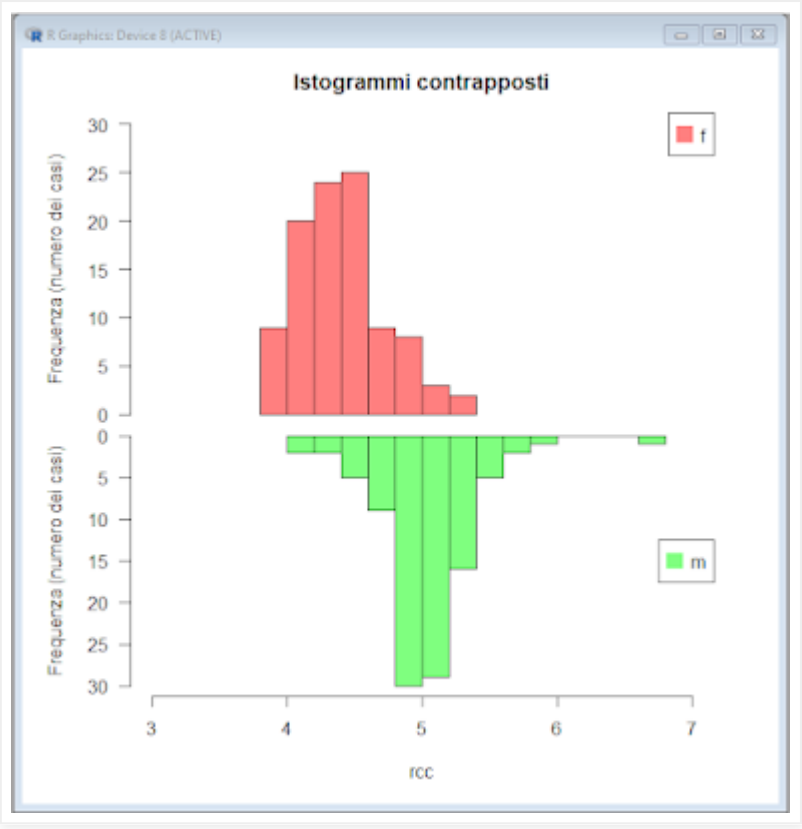
Nella funzione **legend()** l'argomento **fill=c("red","green4")** genera i due riquadri che, impiegando questi stessi quattro argomenti, richiamano colori e aspetto delle barre dei due istogrammi.

Nota bene: dalla sovrapposizione dei due istogrammi unimodali dei dati separati per sesso riotteniamo l'istogramma bimodale della concentrazione degli eritrociti nei dati originari, come volevasi dimostrare.

Potete anche tracciare due istogrammi contrapposti, copiate e incollate quest'ultimo script nella `Console di R` e premete `↵ Invio`:

DUE ISTOGRAMMI CONTRAPPOSTI

#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme di ais con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme di ais con sesso=m
par(mfrow=c(2,1)) # grafici organizzati in due righe e una colonna
#
par(mar=c(0,5,3,3)) # adatta i margini al grafico superiore
hist(f, xaxt="n", xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), col=rgb(1,0,0,0.5), las=1, main="Istogrammi contrapposti", xlab="", ylab="Frequenza (numero dei casi)") # istogramma di f
legend("topright", legend=c("f"), col=c(rgb(1,0,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
par(mar=c(5,5,0,3)) # adatta i margini al grafico inferiore
hist(m, xlim=c(3,7), ylim=c(30,0), breaks=sqrt(length(m)), col=rgb(0,1,0,0.5), las=1, main="", xlab="rcc", ylab="Frequenza (numero dei casi)") # istogramma di m
legend("right", legend=c("m"), col=c(rgb(0,1,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
par(mar=c(5.1,4.1,4.1,2.1)) # ripristina i margini di default
#



Qui non c'è molto da aggiungere, se non che si tratta di una estetica alternativa a quella dello script precedente, che può piacere o non piacere e che viene realizzata con alcuni piccoli accorgimenti, come:
→ **par(mfrow=c(2,1))** per disporre i grafici in due righe per una colonna quindi in definitiva uno sopra l'altro;
→ **par(mar=c(0,5,3,3))** per eliminare il margine inferiore al grafico superiore;
→ **xaxt="n"** per togliere l'asse delle ascisse al grafico superiore;
→ **par(mar=c(5,5,0,3))** per eliminare il margine superiore al grafico superiore;
→ **ylim=c(30,0)** necessario per invertire nel grafico inferiore l'asse delle **y** rispetto al grafico superiore, nel quale abbiamo il più usuale **ylim=c(0,30)**.

Infine ricordo che nel post Salvare i grafici di R in un file è riportato uno script che consente di salvare i grafici sotto forma di file `.bmp`, `.jpeg`, `.png`, `.pdf`, `.ps` per poterli stampare, archiviare, inserire in una pubblicazione, in un post o in un sito web.

Conclusione: gli istogrammi - un rappresentazione grafica semplice e tradizionale della distribuzione di una variabile espressa in una scala numerica continua - anche se potrebbero sembrare uno strumento vetusto, quando impiegati con una opportuna suddivisione in classi, adeguatamente rappresentati e attentamente esaminati, offrono un importante ausilio nell'analisi dei dati.

---

[1] Vedere il post Scale nominali, scale ordinali e scale numeriche.

[2] Vedere nel post Il set di dati ais come acquisire i dati della tabella **ais** anche senza istallare il pacchetto **DAAG**. La concentrazione degli eritrociti è espressa in 10^{12}/L o in 10^{6}/μL, le due unità di misura impiegate nella pratica medica, che sono numericamente identiche.

[3] Vedere il post Visualizzare i colori disponibili in R.

[4] Vedere *Wikipedia*. Freedman–Diaconis rule. URL consultato il 05/01/2023.

[5] Vedere anche il post Adattare i margini a un grafico.

[6] Vedere il post Kernel density plot.

giovedì 1 dicembre 2022

Inferenza causale in statistica

In passato avevo riportato una breve nota nella quale ricordavo che inferire acriticamente da una correlazione un rapporto causa-effetto è sbagliato [1].

Ora ho aggiunto alla bibliografia [2] il link a un articolo di Judea Pearl nel quale, come indica l'Autore stesso, sono illustrati "*... i cambiamenti paradigmatici che devono essere intrapresi nel passaggio dall'analisi statistica tradizionale all'analisi causale di dati multivariati*":

Judea Pearl. "*Causal inference in statistics: An overview.*" Statist. Surv. 3 96 - 146, 2009. https://doi.org/10.1214/09-SS057

Anche se questo è un sito per definizione limitato a fornire un ausilio pratico per il primo utilizzo di **R** penso che ogni tanto tornare ai fondamenti dell'analisi statistica possa essere interessante e stimolante.

---

[1] Vedere il post Correlazione e causazione.

[2] Vedere la pagina Bibliografia.

venerdì 2 dicembre 2022

Kernel density plot

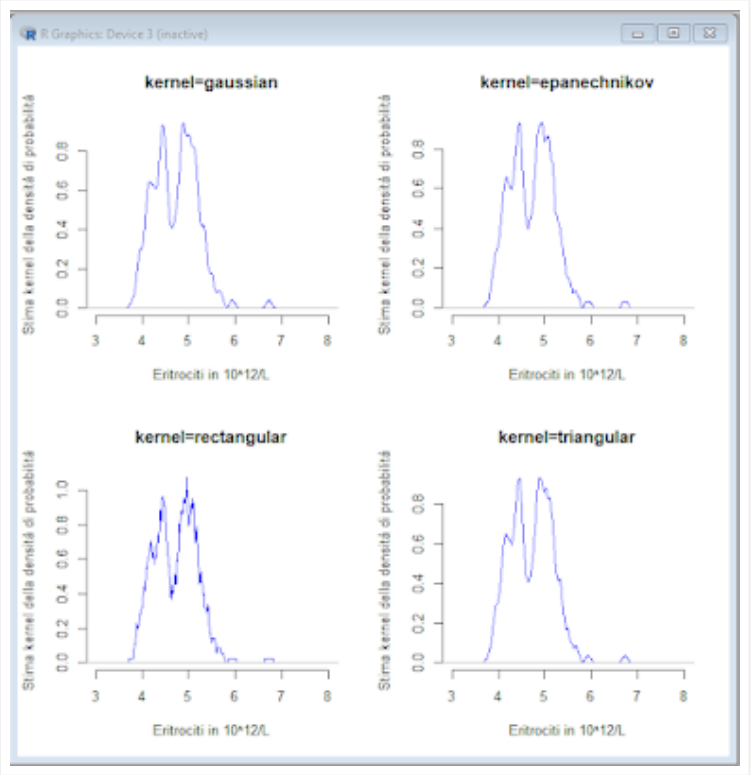
I **kernel density plot** forniscono la rappresentazione grafica della densità di probabilità di una distribuzione o più precisamente della stima kernel di detta densità di probabilità e possono essere utilizzati in alternativa o accanto ai tradizionali istogrammi dei quali rappresentano una interessante evoluzione.

Il metodo non parametrico [1] impiegato per realizzare i kernel density plot - invece di suddividere i dati in classi e rappresentarli sotto forma di barre come negli istogrammi - prevede di collocare in corrispondenza di ciascun dato una piccola *gobba* (**bump**), determinata da un *fattore di forma* (**kernel**), e di sommare tutte le gobbe impiegando un fattore di smussamento detto *ampiezza di banda* (**bandwidth**). La curva risultante, il kernek densiy plot, riporta quindi sulle ascisse `x` i valori della variabile oggetto di studio espressi in una *scala numerica continua* [2] e sulle ordinate `y` la frequenza di tali valori espressa come *[stima kernel della] densità di probabilità della distribuzione*.

Iniziamo esaminando il primo fattore che condiziona l'aspetto di un kernel density plot, il **fattore di forma** (*kernel*) impiegando la concentrazione degli eritrociti (globuli rossi) nel sangue rilevata in 202 atleti australiani riportata nella colonna/variabile **rcc** della tabella **ais** inclusa nel pacchetto **DAAG**. Accertatevi di avere installato il pacchetto o in alternativa procedete come indicato in [3].

Copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT effetto del fattore di forma (kernel)
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # suddivide la finestra in quattro quadranti,  uno per grafico
#
kdpg <-density(ais$rcc, kernel=c("gaussian"), bw=0.05, adjust=1) # calcola il kernel density plot
plot(kdpg, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="kernel=gaussian", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
kdpe <-density(ais$rcc, kernel=c("epanechnikov"), bw=0.05, adjust=1) # calcola il kernel density plot
plot(kdpe, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="kernel=epanechnikov", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
kdpr <-density(ais$rcc, kernel=c("rectangular"), bw=0.05, adjust=1) # calcola il kernel density plot
plot(kdpr, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="kernel=rectangular", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
kdpt <-density(ais$rcc, kernel=c("triangular"), bw=0.05, adjust=1) # calcola il kernel density plot
plot(kdpt, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="kernel=triangular", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
```



L'utilizzo del fattore di forma (*kernel*) è illustrato con quattro grafici, ciascuno generato impiegando:
1) la funzione **density()** con questi argomenti:
→ **ais$rcc** per indicare la variabile da analizzare;
→ **kernel="..."** per indicare il fattore di forma da applicare;
→ **bw=...** per indicare l'ampiezza di banda da applicare;
→ **adjust=...** per indicare il fattore di moltiplicazione da applicare a **bw**, cosa che ne facilita la personalizzazione;
2) la funzione **plot()** che ha come argomenti:
→ l'oggetto generato alla riga precedente dalla funzione **density()** e contenente i dati necessari per tracciare il grafico ;
→ **frame.plot=FALSE** per togliere la cornice rettangolare che delimita il grafico;
→ **xlim=c(... , ...)** per indicare valore minimo e massimo da applicare all'asse delle ascisse;
→ **col="..."** per applicare un colore specifico alla traccia del grafico (valore di default = **"black**");
→ **lwd=...** per indicare lo spessore della linea impiegata per rappresentare il grafico (valore di default = **1**);
→ **main="..."**, **xlab="..."** e **ylab="..."** per specificare titolo, etichetta dell'asse delle `x` ed etichetta dell'asse delle `y`.

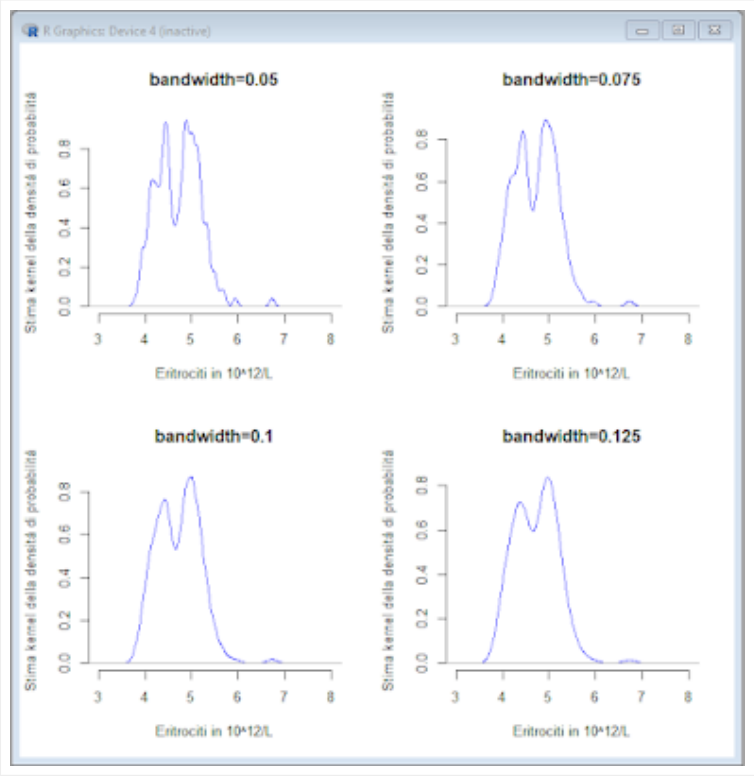
Nella funzione **density()** il fattore di forma viene specificato con l'argomento **kernel="..."** che può assumere i seguenti valori: **"gaussian"**, **"epanechnikov"**, **"rectangular"**, **"triangular"**, **"biweight"**, **"cosine"**, **"optcosine"** [4].

Questo significa che l'algoritmo che genera il kernel density plot sostituisce ciascun valore della variabile in esame con una *gobba* che nei nostri quattro casi ha la forma di una curva gaussiana (**"gaussian"**), di un vertice di parabola con convessità diretta verso l'alto (**"epanechnikov"**), di un rettangolo (**"rectangular"**), di un triangolo (**"triangular"**), forme che risultano evidenti nei due valori isolati e singoli visibili sulla destra in ciascun grafico.

Nota bene: lasciando fissa l'ampiezza di banda **bw=0.05** e variando il fattore di forma si vede che la somma di 202 rettangoli porta a un risultato più irregolare e frastagliato della somma di altrettante curve gaussiane, vertici di parabola o triangoli, ma in ogni caso il kernel density plot risultante mostra una distribuzione *bimodale* dei dati [5].

Vediamo ora il secondo fattore che condiziona l'aspetto di un kernel density plot, e cioè l'**ampiezza di banda** (*bandwidth*). Copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT effetto dell'ampiezza di banda (bandwidth)
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,2)) # suddivide la finestra in quattro quadranti,  uno per grafico
#
kdp50 <-density(ais$rcc, kernel=c("gaussian"), bw=0.05, adjust=1) # calcola il kernel density plot
plot(kdp50, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="bandwidth=0.05", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
kdp75 <-density(ais$rcc, kernel=c("gaussian"), bw=0.075, adjust=1) # calcola il kernel density plot
plot(kdp75, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="bandwidth=0.075", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
kdp100 <-density(ais$rcc, kernel=c("gaussian"), bw=0.1, adjust=1) # calcola il kernel density plot
plot(kdp100, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="bandwidth=0.1", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
kdp125 <-density(ais$rcc, kernel=c("gaussian"), bw=0.125, adjust=1) # calcola il kernel density plot
plot(kdp125, frame.plot=FALSE, xlim=c(3,8), col="blue", lwd=1, main="bandwidth=0.125", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
#
```



Nello script precedente abbiamo variato il fattore di forma lasciando fissa l'ampiezza di banda **bw=0.05**. Qui invece mantenendo fisso il fattore di forma **kernel="gaussian"** abbiamo fatto variare l'ampiezza di banda, portandola dal valore iniziale **bw=0.05** prima a **bw=0.075** poi a **bw=0.1** e infine a **bw=0.125**.

Risulta così documentato l'effetto dell'ampiezza di banda sul grafico: aumentandola aumenta lo smussamento del kernel density plot risultante, mentre diminuendola il grafico si adatta viepiù ai singoli punti e diventa perciò irregolare.

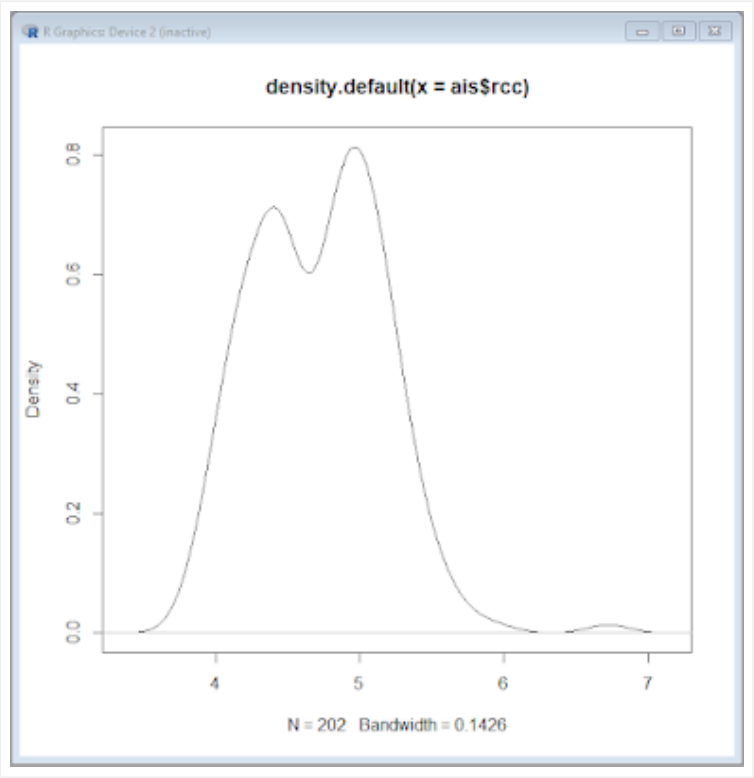
Nota bene: nel nostro caso il valore **bw=0.05** porta ad un eccessivo adattamento della curva ai singoli punti (*overfitting*) rendendola irregolare mentre il valore **bw=0.125** inizia a mostrare una minor separazione tra le due sottoclassi e quindi ci sta spostando verso un adattamento della curva ai dati troppo ridotto (*underfitting*) a causa dell'eccessivo fattore di smussamento. A questo punto la scelta tra un **bw=0.075** che conserva un maggior dettaglio e un **bw=0.1** che comunque mantiene l'informazione rilevante diventa una scelta personale, che dipende anche da quello che si intende comunicare e discutere in merito ai dati.

Dopo averne brevemente illustrato razionale e tecniche soggiacenti, possiamo passare a qualche esempio pratico di applicazione e personalizzazione dei kernel density plot realizzati impiegando solamente le *funzioni statistiche e grafiche di base* di **R**, aggiungendo al termine un esempio che impiega le funzioni contenute nel pacchetto aggiuntivo **sm** che consentono di realizzare facilmente kernel density plot multipli sovrapposti.

Gli esempi sono stati riportati in script separati, in modo che quelli che più interessano possano essere salvati indipendentemente l'uno dall'altro ed essere successivamente ripresi per analizzare i propri dati semplicemente eliminando la prima riga di codice e sostituendo **ais$rcc** con il nome della propria variabile.

Copiate e incollate questo primo script nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT SEMPLICE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
plot(density(ais$rcc)) # traccia il kernel density plot
#
```



Il kernel density plot più semplice che è possibile realizzare con **R** prevede, dopo avere caricato i dati e aperto e inizializzato una nuova finestra grafica, una sola riga di codice nella quale:

→ la funzione **density()**, che ha come unico argomento il nome della variabile da analizzare **ais$rcc**, genera la stima kernel della densità di probabilità necessaria per tracciare il grafico;
→ la funzione **plot()** impiega i dati generati dalla funzione **density()** per tracciare il grafico.

Se digitate

str(density(ais$rcc))

potete visualizzare la struttura dei dati generati dalla funzione **density()**

```
> str(density(ais$rcc))
List of 7
 $ x         : num [1:512] 3.37 3.38 3.39 3.39 3.4 ...
 $ y         : num [1:512] 0.000214 0.000251 0.000297 0.000349 0.000409 ...
 $ bw        : num 0.143
 $ n         : int 202
 $ call      : language density.default(x = ais$rcc)
 $ data.name: chr "ais$rcc"
 $ has.na    : logi FALSE
 - attr(*, "class")= chr "density"
```

mentre se digitate

density(ais$rcc)$x

e

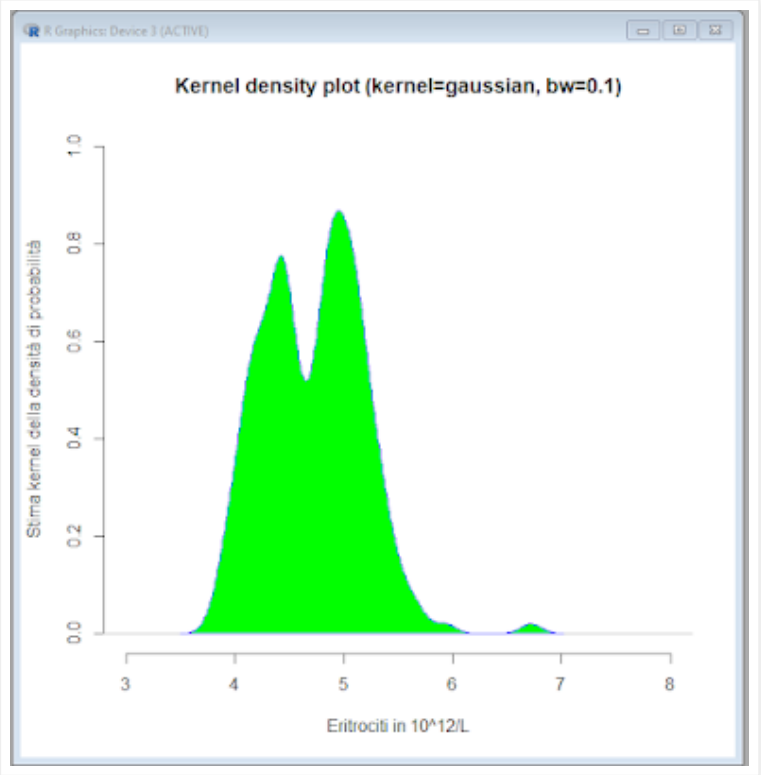
density(ais$rcc)$y

potete visualizzare rispettivamente in dettaglio tutti i 512 valori delle `x` e gli altrettanti valori delle `y` generati dalla funzione **density()** che la funzione **plot()** impiega per tracciare il grafico.

Nota bene: lo script contiene una soluzione minima che può essere utile quando si voglia effettuare una rapida analisi preliminare dei dati, ma non solo, perché la rappresentazione ottenuta con gli argomenti di default previsti da **R** fornisce, oltre al numero `N` dei dati, un'informazione importante: l'ampiezza di banda (*bandwidth*) impiegata per tracciare il grafico, che la funzione **density()** calcola automaticamente di volta in volta.

Per realizzare un kernel density plot personalizzato ora copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT PERSONALIZZATO
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
kdp <-density(ais$rcc, kernel=c("gaussian"), bw=0.1, adjust=1) # calcola il kernel density plot
plot(kdp, frame.plot=FALSE, xlim=c(3,8), ylim=c(0,1), main="Kernel density plot (kernel=gaussian, bw=0.1)", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot
polygon(kdp, col="green", border="blue", lwd=1) # colora il kernel density plot
#
```



Qui la funzione **density()** è stata arricchita con gli argomenti:
→ **ais$rcc** per indicare la variabile da analizzare;
→ **kernel="gaussian"** per indicare che il fattore di forma da applicare è la curva gaussiana;
→ **bw=0.1** per indicare l'ampiezza di banda da applicare;
→ **adjust=1** per indicare il fattore di moltiplicazione da applicare a **bw**, cosa che in questo caso lascia **bw** immodificato.

La funzione **plot()** è stata personalizzata con gli argomenti già commentati nel primo script.

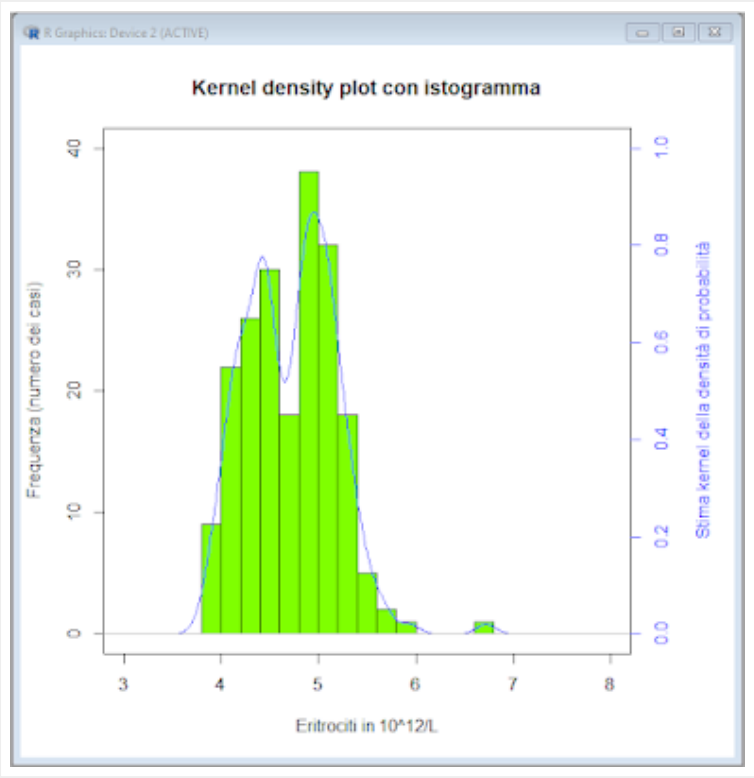
La novità è la funzione **polygon()** che consente di colorare il grafico specificando:
→ **kdp** per la variabile da analizzare;
→ **col="green"** per il colore di riempimento del kernel density plot;
→ **border="blue"** per il colore della linea che delimita il kernel density plot;
→ **lwd=1** per lo spessore della linea che delimita il kernel density plot.

Digitate **help(polygon)** nella `Console di R` se volete approfondire la conoscenza di questa funzione.

A questo punto vediamo come fare una cosa che prima o poi viene in mente a tutti: sovrapporre un kernel density plot e l'istogramma corrispondente. Copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT CON L'ISTOGRAMMA CORRISPONDENTE e ordinate separate
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
par(mar=c(5.1,4.1,4.1,5.1)) # aumenta il margine destro da 2.1 a 5.1
#
hist(ais$rcc, xlim=c(3,8), ylim=c(0,40), freq=TRUE, col="chartreuse", breaks="FD",
main="Kernel density plot con istogramma", xlab="Eritrociti in 10^12/L",
ylab="Frequenza (numero dei casi)") # traccia l'istogramma
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
#
kdp <-density(ais$rcc, kernel=c("gaussian"), bw=0.1, adjust=1) # calcola il kernel density
plot
```

plot(kdp, yaxt="n", xlim=c(3,8), ylim=c(0,1), col="blue", lwd=1) # traccia il kernel density plot
axis(4, col.ticks="blue", col.axis="blue") # riporta l'asse delle y del kernel density plot sulla destra
mtext("Stima kernel della densità di probabilità", side=4, line=3, cex=1, col="blue") # aggiunge la legenda all'asse delle y sulla destra
#
par(mar=c(5.1,4.1,4.1,2.1)) # ripristina i margini di default
#



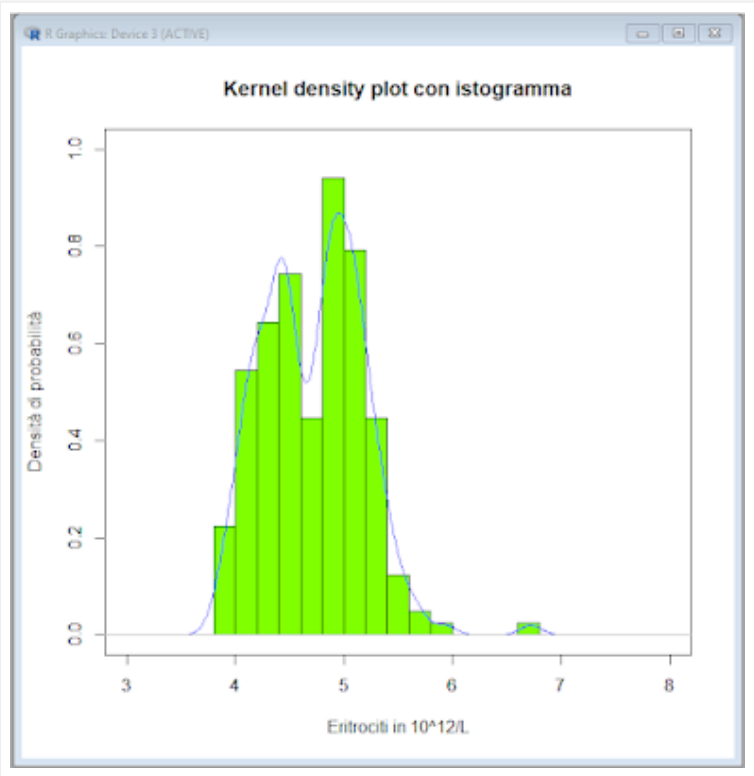
Vogliamo sovrapporre istogramma e kernel density plot riportando due assi delle ordinate separati e con scale diverse:
→ l'asse delle `y` sulla sinistra riferito all'istogramma e che riporta la *frequenza* espressa come numero dei casi;
→ l'asse delle `y` sulla destra riferito al kernel density plot e che riporta la *stima kernel della densità di probabilità*.

Ecco le principali osservazioni sugli argomenti e sulle funzioni impiegate:
→ come prima cosa con la funzione **par()** il margine sulla destra del grafico (il quarto valore dell'argomento **mar**) viene aumentato dal valore di default **2.1** a **5.1** per creare lo spazio necessario per la legenda dell'asse delle `y` del kernel density plot che verrà posizionato appunto sulla destra;
→ nella funzione **hist()** viene riportato in ordinate il numero dei casi ponendo **freq=TRUE**;
→ il numero delle classi **breaks** viene calcolato con la regola di Freedman-Diaconis (**FD**);
→ la funzione **par(new=TRUE, ann=FALSE)** specifica che il grafico successivo verrà sovrapposto (**new=TRUE**) ma che non devono essere annotati automaticamente titolo e legende (**ann=FALSE**), perchè questo lo faremo con le righe successive;
→ con la funzione **plot()** il kernel density plot viene sovrapposto all'istogramma specificando con **yaxt="n"** che non deve essere riportato l'asse delle `y` che si sovrapporrebbe a quello dell'istogramma e che invece noi aggiungeremo ora sulla destra;
→ con la funzione **axis()** riportiamo l'asse delle `y` del kernel density plot nella posizione **4** cioè sulla destra (nella grafica di base di **R** i margini sono 1=margine inferiore, 2=margine sinistro, 3=margine superiore, 4=margine destro) e riportiamo in colore **"blue"** sia le tacche sia i valori della scala;
→ con la funzione **mtext()** aggiungiamo sulla destra (**side=4**) la legenda dell'asse delle `y` del kernel density plot, posizioniamo l'etichetta **"Stima kernel della densità di probabilità"** sulla **line=3** (la linea 0 contiene le tacche, la 1 contiene i valori numerici, la 2 è volutamente lasciata vuota), riportiamo i caratteri nella dimensione standard (**cex=1**) e in colore blu (**col="blue"**);

→ con la funzione **par()** ripristiniamo i margini di default, di fatto riportiamo quello di destra (il quarto valore dell'argomento **mar**) da **5.1** a **2.1**.

Se preferite impiegare un'unica scala sia per l'istogramma sia per il kernel density plot, esprimendo per entrambi le ordinate in termini di *densità di probabilità*, copiate e incollate nella `Console di R` questo script e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT CON L'ISTOGRAMMA CORRISPONDENTE e ordinate comuni
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
hist(ais$rcc, xlim=c(3,8), ylim=c(0,1), freq=FALSE, breaks="FD", col="chartreuse", main="Kernel density plot con istogramma", xlab="Eritrociti in 10^12/L", ylab="Densità di probabilità") # traccia l'istogramma
par(new=TRUE, ann=FALSE) # consente la sovrapposizione del grafico successivo
#
kdp <-density(ais$rcc, kernel=c("gaussian"), bw=0.1, adjust=1) # calcola il kernel density plot
plot(kdp, xlim=c(3,8), ylim=c(0,1), col="blue", lwd=1) # sovrappone all'istogramma il kernel density plot
#
```

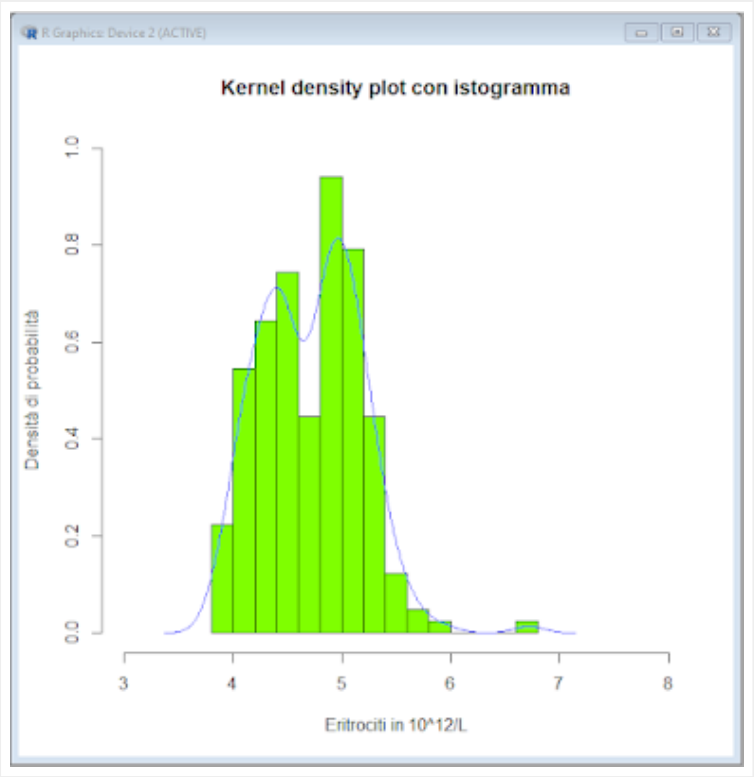


Come si vede sono ora sufficienti (tolte le prime due) quattro righe di codice:
→ la prima per tracciare con la funzione **hist()** l'istogramma, per gli argomenti del quale si rimanda al bisogno al post Istogrammi;
→ la seconda per consentire con la funzione **par()** la sovrapposizione all'istogramma di un nuovo grafico;
→ la terza per calcolare con la funzione **density()** il kernel density plot;
→ la quarta per tracciare con la funzione **plot()** il kernel density plot calcolato con la funzione **density()**.

Ma potete anche semplificare ulteriormente il codice impiegando questa sintetica variante:

```
# KERNEL DENSITY PLOT CON L'ISTOGRAMMA CORRISPONDENTE e ordinate comuni
#
```

library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
hist(ais$rcc, xlim=c(3,8), ylim=c(0,1), freq=FALSE, breaks="FD", col="chartreuse", main="Kernel density plot con istogramma", xlab="Eritrociti in 10^12/L", ylab="Densità di probabilità") # traccia l'istogramma
#
lines(density(ais$rcc), xlim=c(3,8), ylim=c(0,1), col="blue", lwd=1) # sovrappone all'istogramma il kernel density plot
#



Come si vede sono ora sufficienti:
→ la funzione **hist()** per tracciare l'istogramma;
→ la funzione **lines()** per sovrapporre all'istogramma il kernel density plot calcolato con **density(ais$rcc)**.

In questo caso il grafico è un po' più spartano e il kernel density plot risulta lievemente diverso dai due precedenti in quanto per semplificare al massimo il codice non viene imposto l'argomento **bw=0.1** ma viene lasciato che sia **R** a calcolarne il valore. Se digitate

density(ais$rcc)$bw

vi viene mostrato il valore di banda passante calcolato automaticamente da **R**

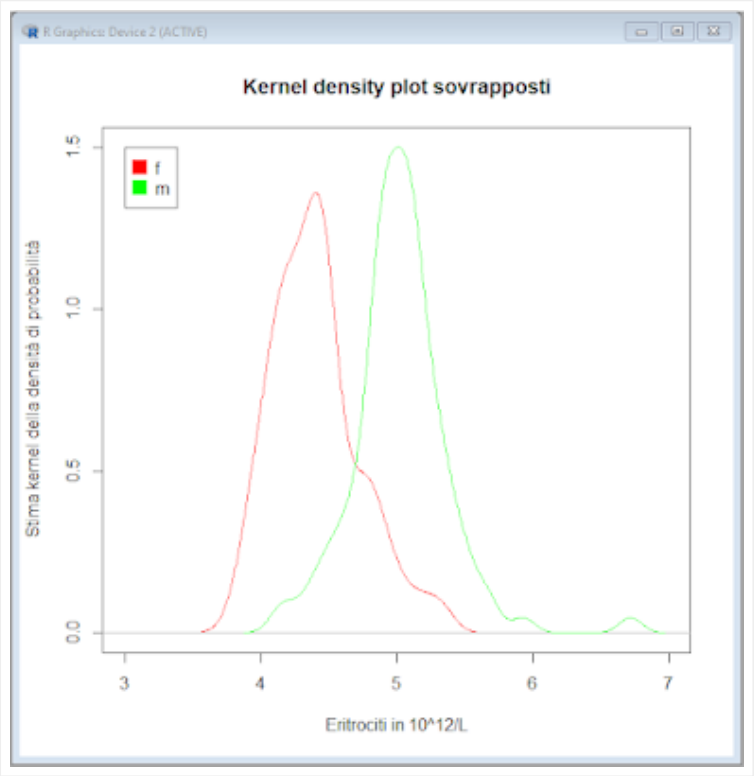
```
> density(ais$rcc)$bw
[1] 0.1425658
```

e potete notare che in effetti è superiore a **0.1** cosa che comporta un maggior smussamento del kernel density plot, che vedete più arrotondato e con una minore separazione dei picchi rispetto ai due precedenti script.

Nota bene: il kernel density plot proprio per il modo nel quale è calcolato presenta una distribuzione *bimodale* che ben si adatta a quella che si osserva nell'istogramma.

Facciamo un ulteriore passo avanti con il codice che sovrappone due kernel density plot in modo semplice e conciso. Copiate e incollate questo script nella `Console di R` e premete ↵ `Invio`:

DUE KERNEL DENSITY PLOT SOVRAPPOSTI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # sottoinsieme dei dati con sesso=m
#
plot(density(f), xlim=c(3,7), ylim=c(0,1.5), col="red", lwd=1, main="Kernel density plot sovrapposti", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") # traccia il kernel density plot del sesso=f
#
lines(density(m), xlim=c(3,7), ylim=c(0,1.5), col="green", lwd=1) # traccia il kernel density plot del sesso=m
#
legend(3, 1.5, legend=c("f","m"), col=c("red", "green"), pt.cex=2, pch=15) # aggiunge la legenda
#



Per la funzione **subset()** dobbiamo partire dalla tabella **ais**, se nella `Console di R` digitate

str(ais)

vi comparirà la struttura della tabella

```
> str(ais)
'data.frame':   202 obs. of  13 variables:
 $ rcc   : num  3.96 4.41 4.14 4.11 4.45 4.1 4.31 4.42 4.3 4.51 ...
 $ wcc   : num  7.5 8.3 5 5.3 6.8 4.4 5.3 5.7 8.9 4.4 ...
 $ hc    : num  37.5 38.2 36.4 37.3 41.5 37.4 39.6 39.9 41.1 41.6 ...
 $ hg    : num  12.3 12.7 11.6 12.6 14 12.5 12.8 13.2 13.5 12.7 ...
 $ ferr  : num  60 68 21 69 29 42 73 44 41 44 ...
 $ bmi   : num  20.6 20.7 21.9 21.9 19 ...
 $ ssf   : num  109.1 102.8 104.6 126.4 80.3 ...
 $ pcBfat: num  19.8 21.3 19.9 23.7 17.6 ...
```

```
$ lbm    : num  63.3 58.5 55.4 57.2 53.2 ...
$ ht     : num  196 190 178 185 185 ...
$ wt     : num  78.9 74.4 69.1 74.9 64.6 63.7 75.2 62.3 66.5 62.9 ...
$ sex    : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
$ sport  : Factor w/ 10 levels "B_Ball","Field",..: 1 1 1 1 1 1 1 1 1 1 ...
```

che come si vede contiene 13 variabili. Le prime 11 sono variabili numeriche (`num`), le ultime due sono fattori (`Factor`), sono cioè variabili qualitative opportunamente codificate, la variabile `sex` per indicare il sesso impiegando 2 livelli di codifica (`"f"` e `"m"`), la variabile `sport` per indicare lo sport praticato, impiegando 10 livelli di codifica.

Con la funzione **subset()** sono estratti da **ais** i sottoinsiemi di dati **f** e **m** impiegando i seguenti argomenti:
→ **ais** per indicare la fonte dei dati da estrarre;
→ **sex="f"** ovvero **sex="m"** per estrarre da **ais** rispettivamente i dati corrispondenti agli atleti di sesso femminile e agli atleti di sesso machile;
→ **select=(rcc)** per estrarre da **ais** solamente i valori della variabile **rcc** che sono quelli che ci interessano per tracciare i kernel density plot.

I due kernel density plot sono tracciati con due righe di codice che impiegano:
→ la funzione **plot()** per tracciare il primo kernel density plot;
→ la funzione **lines()** per sovrapporre il secondo.

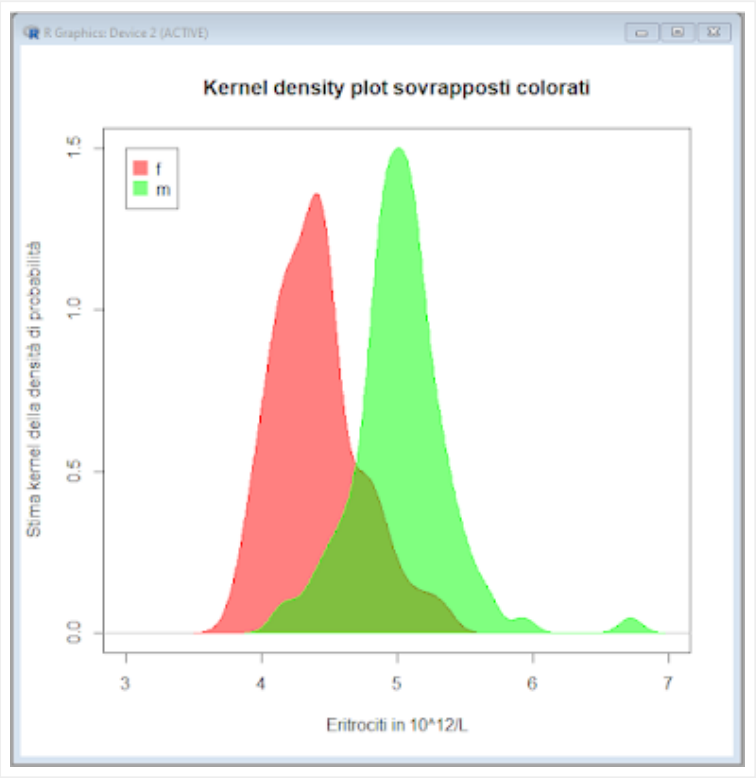
Da notare a proposito della funzione **lines()** che:
→ riporta sullo stesso grafico e unisce tra loro con segmenti di retta i punti con le coordinate `x,y` del kernel density plot calcolate dalla funzione **density()**;
→ riporta come scala degli assi gli stessi valori **xlim=c(3,7)** e **ylim=c(0,1.5)** impiegati per il primo kernel density plot;
→ traccia i segmenti di retta che descrivono il kernel density plot con il colore **col="green"** e con una linea di spessore **lwd=1**.

La funzione **legend()** [6] identifica i dati rappresentati impiegando i seguenti argomenti:
→ **3** per l'ascissa cui allineare il lato sinistro della legenda;
→ **1.5** per l'ordinata cui allineare l'angolo superiore della legenda;
→ **legend=c(...)** per l'elenco delle etichette che descrivono i dati:
→ **col=c(...)** per l'elenco dei colori corrispondenti ai dati
→ **pt.cex=2** per indicare il fattore di ingrandimento dei simboli;
→ **pch=15** per indicare il tipo di simbolo da impiegare (15=un quadrato).

Vediamo ora come sovrapporre due kernel density plot colorati e renderli visibili nella loro interezza, senza che uno nasconda parzialmente l'altro, utilizzando la trasparenza del colore. Copiate e incollate lo script nella `Console di R` e premete `↵ Invio`:

```
# DUE KERNEL DENSITY PLOT SOVRAPPOSTI COLORATI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # sottoinsieme dei dati con sesso=m
#
plot(density(f), xlim=c(3,7), ylim=c(0,1.5), main="Kernel density plot sovrapposti
colorati", xlab="Eritrociti in 10^12/L", ylab="Stima kernel della densità di probabilità") #
traccia il kernel density plot del sesso=f
polygon(density(f), col=rgb(255,0,0,127,maxColorValue=255), border="red", lwd=1) #
colora il kernel density plot del sesso=f
#
lines(density(m), xlim=c(3,7), ylim=c(0,1.5)) # traccia il kernel density plot del sesso=m
```

polygon(density(m), col=rgb(0,255,0,127,maxColorValue=255), border="green", lwd=1) # colora il kernel density plot del sesso=m
#
legend(3, 1.5, legend=c("f","m"), col=c(rgb(255,0,0,127,maxColorValue=255), rgb(0,255,0,127,maxColorValue=255)), pt.cex=2, pch=15) # aggiunge la legenda
#



La funzione **subset()** impiegata per estrarre da **ais** i sottoinsiemi di dati **f** e **m** è stata illustrata nello script precedente.

Per sovrapporre i due kernel density plot colorati sono state impiegate le funzioni **plot()**, **density()**, **polygon()** e **lines()** anch'esse già illustrate in precedenza.

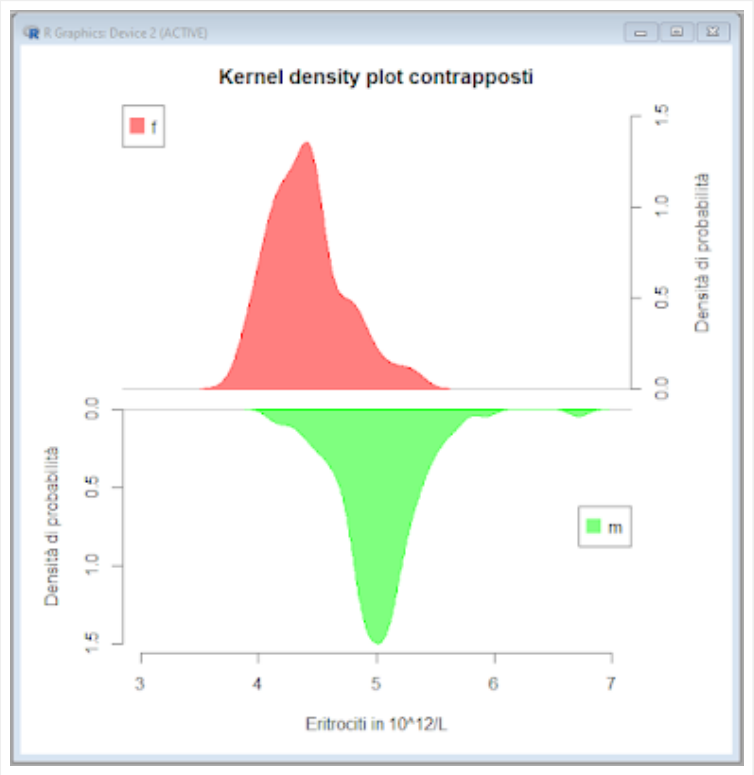
Nelle funzioni **poligon()** e **legend()** il colore è stato specificato con la funzione **rgb(n,n,n,n)** nella quale:
→ il primo **n**umero indica il rosso (**r**ed);
→ il secondo il verde (**g**reen);
→ il terzo il blu (**b**lu);
→ il quarto il grado di trasparenza del colore impiegato.

Ponendo **maxColorValue=255** questi quattro valori possono essere espressi in una scala compresa tra **0** e **255**. Così ad esempio i valori di **rgb** uguali a **255,0,0,127** del primo dei due kernel density plot stanno per rosso al 100%, no verde, no blu, trasparenza del colore uguale al 50%. Digitale **help(rgb)** per la documentazione completa della funzione.

Si possono anche realizzare due kernel density plot contrapposti. Copiate e incollate lo script nella `Console di R` e premete `↵ Invio`:

DUE KERNEL DENSITY PLOT CONTRAPPOSTI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
windows() # apre e inizializza una nuova finestra grafica
#
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme di ais con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme di ais con sesso=m

par(mfrow=c(2,1)) # grafici organizzati in due righe e una colonna
#
par(mar=c(0,5,3,5)) # adatta i margini al grafico superiore
plot(density(f), frame.plot=FALSE, xaxt="n", yaxt="n", xlim=c(3,7), ylim=c(0,1.5), main="Kernel density plot contrapposti", xlab="", ylab="") # traccia il kernel density plot del sesso=f
axis(4) # riporta l'asse delle y del kernel density plot sulla destra
mtext("Densità di probabilità", side=4, line=3) # aggiunge la legenda all'asse delle y sulla destra
polygon(density(f), col=rgb(1,0,0,0.5), border="red", lwd=1) # colora il kernel density plot
legend("topleft", legend=c("f"), col=c(rgb(1,0,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
#
par(mar=c(5,5,0,5)) # adatta i margini al grafico inferiore
plot(density(m), frame.plot=FALSE, xlim=c(3,7), ylim=c(1.5,0), main="", xlab="Eritrociti in 10^12/L", ylab="Densità di probabilità") # traccia il kernel density plot del sesso=m
polygon(density(m), col=rgb(0,1,0,0.5), border="green", lwd=1) # colora il kernel density plot
legend("right", legend=c("m"), col=c(rgb(0,1,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
#



Si tratta di una estetica alternativa a quella dello script precedente, che può piacere o non piacere, e che viene realizzata in modo molto semplice con le funzioni:
→ **plot()** per tracciare il grafico;
→ **polygon()** per colorare il grafico;
→ **legend()** per aggiungere la legenda.

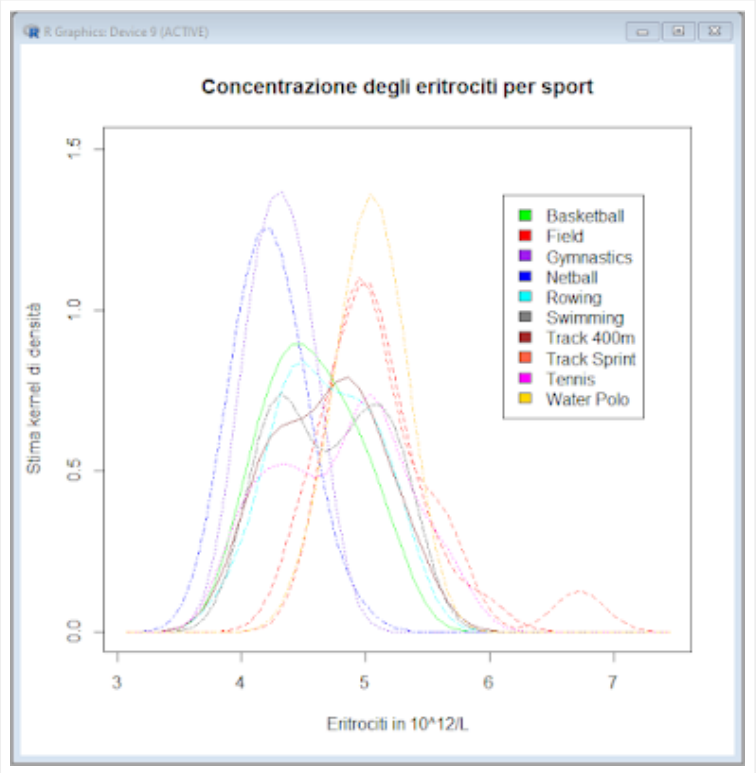
Per realizzare il grafico queste tre funzioni sono integrate con i seguenti accorgimenti:
→ **par(mfrow=c(2,1))** per disporre i grafici in due righe per una colonna quindi in definitiva uno sopra l'altro;
→ **par(mar=c(0,5,3,5))** per eliminare il margine inferiore e aumentare il margine destro al grafico superiore;
→ **xaxt="n"** per togliere l'asse delle x al grafico superiore (è sufficiente l'asse delle ascisse riportato al grafico inferiore);

→ **yaxt="n"** per togliere al grafico superiore l'asse delle `y` che per default verrebbe riportato sulla sinistra;
→ **axis(4)** per riportare l'asse delle `y` del grafico superiore sulla destra;
→ **mtext()** per aggiungere la legenda **"Densità di probabilità"** all'asse delle `y` di destra (**side=4**) sulla riga 3 (**line=3**);
→ **par(mar=c(5,5,0,4))** per eliminare il margine superiore e aumentare il margine destro al grafico inferiore;
→ **ylim=c(1.5,0)** necessario per invertire nel grafico inferiore l'asse delle y rispetto al grafico superiore, nel quale abbiamo **ylim=c(0,1,5)**.

Da notare che nelle funzioni **polygon()** e **legend()** per definire il colore viene impiegata la funzione **rgb(n,n,n,n)** ma, contrariamente a quanto avevamo fatto nello script precedente, questa volta non viene specificato l'argomento `maxColorValue` che pertanto qui assume il valore di default, che è uguale a `1`. Quindi i valori **n** assegnati nell'ordine al rosso (**r**ed), al verde (**g**reen), al blu (**b**lu) e al grado di trasparenza del colore, per definizione possono variare solamente tra `0` e `1`. In questa scala il valore **0.5** del grado di trasparenza equivale al valore **127** che nello script precedente era espresso in una scala da **0** a **255**.

Qualora si debbano sovrapporre molti kernel density plot, può risultare comodo impiegare le funzioni contenute nel pacchetto **sm**. Accertatevi di avere installato il pacchetto altrimenti dovete scaricarlo e installarlo dal **CRAN**. Ora copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# KERNEL DENSITY PLOT MULTIPLI SOVRAPPOSTI con il pacchetto sm
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(sm) # carica il pacchetto
windows() # apre e inizializza una nuova finestra grafica
#
sport <- c("B_Ball", "Field", "Gym", "Netball", "Row", "Swim", "T_400m", "T_Sprnt",
"Tennis", "W_Polo") # vettore che riporta la codifica impiegata per la variabile/fattore sport
etichette <- c("Basketball", "Field", "Gymnastics", "Netball", "Rowing", "Swimming",
"Track 400m", "Track Sprint", "Tennis", "Water Polo") # vettore con le etichette da assegnare
agli sport
colori <- c("green", "red", "purple", "blue", "cyan", "grey50", "brown", "tomato",
"magenta", "gold") # vettore con i colori da assegnare agli sport
#
sm.density.compare(ais$rcc, ais$sport, col=colori, xlab="Eritrociti in 10^12/L",
ylab="Stima kernel di densità") # traccia i kernel density plot separati per sport
#
title(main="Concentrazione degli eritrociti per sport praticato") # aggiunge il titolo
legend(locator(1), levels(factor(ais$sport, levels=sport, labels=etichette)), fill=colori) #
predispone la legenda con etichette e colori
#
# resta in attesa del click con il tasto sinistro del mouse nel punto in cui si desidera posizionare la
legenda
#
```



Il punto di forza è che se i dati sono associati a una variabile qualitativa per la quale sono previsti `N` livelli di codifica, vengono generati automaticamente sia gli `N` kernel density plot corrispondenti sia la legenda necessaria per identificarli. Inoltre la legenda può essere posizionata nel grafico in un punto a piacere con un semplice `click` del mouse.

Nel nostro caso per rappresentare la concentrazione degli eritrociti rilevata nei 10 sport praticati dagli atleti australiani in altrettanti kernel density plot procediamo come segue:
→ nel vettore **sport=c(...)** riportiamo la codifica dei livelli impiegata nella variabile (fattore) `sport`;
→ nel vettore **etichette=c(...)** riportiamo nello stesso ordine le scritte che vogliamo impiegare per decrivere ciascuno sport;
→ nel vettore **colori=c(...)** riportiamo nello stesso ordine i colori che vogliamo applicare a ciacuno sport nel kernel density plot e nella legenda corrispondente;
→ con la funzione **sm.density.compare()** tracciamo i kernel density plot per la variabile **ais$rcc** suddividendoli per sport (**ais$sport**) impiegando i colori definiti nel vettore **colori**;
→ con la funzione **title()** riportiamo il titolo del grafico;
→ con la funzione **legend()** riportiamo riportiamo nella legenda, per la variabile (fattore) **ais$sport**, i livelli definiti nel vettore **sport**, con le etichette definite nel vettore **etichette** e i colori definiti nel vettore **colori**.

Una volta che sono comparsi i grafici, posizionate il puntatore del mouse dove volete che compaia la legenda e fate `click` con il tasto sinistro del mouse per farla comparire, terminando così lo script. Da notare che il codice riportato non consente di riposizionare dinamicamente la legenda: se non si è soddisfatti della sua posizione, è necessario rieseguire l'intero script e fare nuovamente `click` con il tasto sinistro del mouse nel nuovo punto in cui la si vuole posizionare.

Come al solito gli script riportati possono essere immediatamente riutilizzati sostituendo **ais$rcc** con il nome della propria variabile e con il semplice adattamento di titoli ed etichette degli assi [7].

---

[1] I metodi parametrici sono basati sull'assunto che i dati abbiano una distribuzione gaussiana e utilizzano i "parametri" *media* (misura di posizione) e *deviazione standard* (misura di dispersione) nella sintesi dei dati e nell'esecuzione dei test statistici. I metodi non parametrici invece non prevedono assunti riguardo la distribuzione dei dati e per definizione non impiegano i citati "parametri" nella sintesi dei dati e nell'esecuzione dei test statistici.

[2] Vedere il post Scale nominali, scale ordinali e scale numeriche.

[3] Vedere il post Il set di dati ais. La concentrazione degli eritrociti viene espressa in 10^{12}/L o in 10^{6}/μL, le due unità di misura impiegate nella pratica, che sono numericamente identiche. Nel post trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[4] Qui vediamo i primi quattro fattori di forma, per i fattori **"biweight"**, **"cosine"** e **"optcosine"**, che forniscono risultati molto simili tra loro e praticamente identici a **"gaussian"**, si rimanda alla documentazione di **R**.

[5] La distribuzione *unimodale* è per definizione quella della distribuzione gaussiana, la distribuzione *bimodale* tipicamente si osserva quando una variabile/serie di misure è composta da due sottoinsiemi aventi *mode* (ma anche medie) differenti, mentre una distribuzione *plurimodale* (o *polimodale*) è quella con mode multiple.

[6] Trovate altri esempi di legende nel post Aggiungere la legenda a un grafico.

[7] Un poco di lavoro in più è necessario per adattare lo script per la generazione di kernel density plot multipli sovrapposti con il pacchetto **sm**, nel quale **ais$rcc** deve essere sostituito con il nome della `tabella$variabile` da analizzare, **ais$sport** deve essere sostituito con il nome della `tabella$variabile` che contiene i fattori dai quali generare i kernel density plot, e infine devono essere adattati opportunamente il vettore **sport** che contiene i nomi dei fattori e i vettori **etichette** e **colori**.

domenica 8 gennaio 2023

Analisi dei gruppi (clustering gerarchico)

L'analisi dei gruppi si applica a *dati multivariati* ed è un metodo statistico di tassonomia numerica che riveste un ruolo importante nella *analisi esplorativa dei dati*.

L'obiettivo dell'**analisi dei gruppi** (*cluster analysis* o *clustering*) è concettualmente semplice: verificare la possibile esistenza, in un insieme di oggetti, di sottoinsiemi di oggetti particolarmente simili tra loro (gruppi/cluster).

Nonostante alla base dell'analisi dei gruppi vi sia un'idea semplice e logica, vi sono numerosi modi per realizzarla [1], qui ci occupiamo dell'implementazione del *clustering gerarchico* con *metodi esclusivi* nelle due versioni:
→ **clustering agglomerativo** o *bottom-up* (con il metodo di Ward e con AGglomerative NESting, AGNES);
→ **clustering divisivo** o *top-down* (con DIvisive ANAlysis, DIANA).

Come dati impieghiamo i valori di BMI (indice di massa corporea) rilevati a livello europeo alcuni anni fa e pubblicati dall'Istat [2].

Per proseguire è necessario:
→ effettuare il download del file di dati `bmi.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un ↲ `Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

```
Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
```

Spagna;2.2;45.4;35.7;16.7
Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2

Inoltre è necessario scaricare dal **CRAN** il pacchetto aggiuntivo **cluster** [3] e il pacchetto aggiuntivo **dendextend** [4].

Iniziamo con un esempio di custering gerarchico che impiega esclusivamente i valori di default. Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# CLUSTERING GERARCHICO con i valori di default
#
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # calcola la deviata normale standardizzata dei dati
windows() # apre e inizializza una nuova finestra grafica
#
mat <- dist(z) # calcola la matrice delle distanze
clus <- hclust(mat) # effettua il clustering gerarchico
plot(clus) # traccia il dendrogramma
#
```

Con la prima riga di codice i dati sono importati nell'oggetto **mydata**.

Con la funzione **scale()** della seconda riga per ciascun dato viene calcolata la *deviata normale standardizzata z*. In pratica questa funzione prima calcola per i dati di ciascuna colonna/variabile la `media` e la `deviazione standard`, poi calcola per ciascuno dato `x` la corrispondente deviata normale standardizzata `z` come

```
z = (x - media) / deviazione standard
```

I valori di `z` sono salvati nel nuovo oggetto qui denominato per comodità mnemonica **z**. Se nella `Console di R` digitate **z**

```
> z
```

sono mostrati i dati standardizzati, in coda ai quali sono riportate la `media`

```
attr(,"scaled:center")
 sottopeso    normale sovrappeso      obeso
  2.103704  44.537037  36.240741  17.111111
```

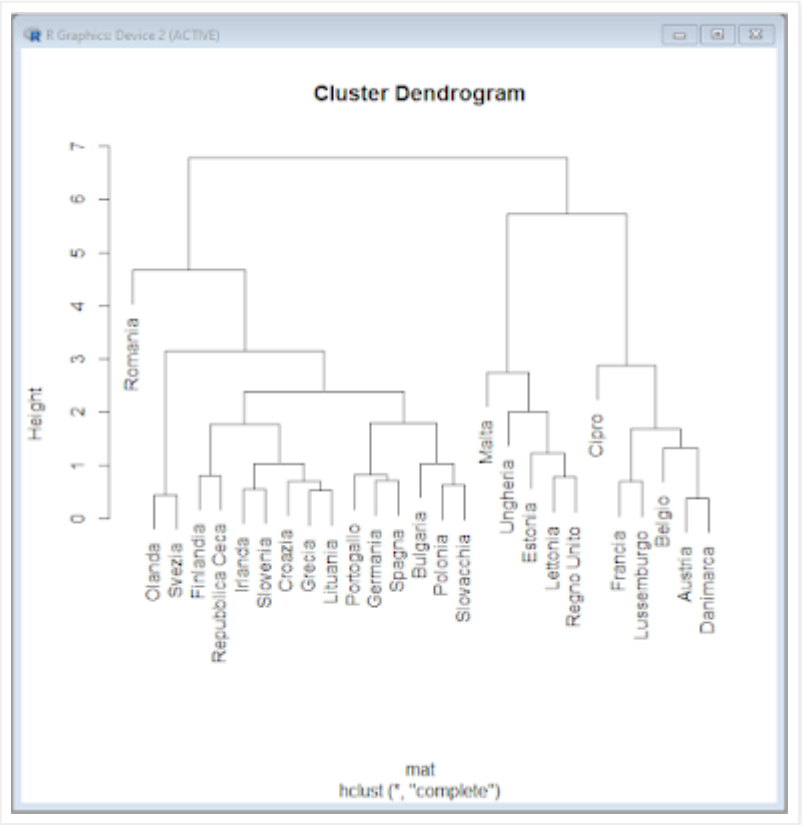
e la `deviazione standard`

```
attr(,"scaled:scale")
 sottopeso    normale sovrappeso      obeso
 0.6111033  3.3717318  2.8989732  3.2322097
```

impiegate per effettuare la standardizzazione dei dati.

Dopo aver aperto e inizializzato una nuova finestra grafica con **windows()**, si passa al clustering gerarchico, realizzato in modo molto semplice, con tre sole funzioni, che impiegano per i loro argomenti i valori di default:
→ la funzione **dist()** che, sui valori standardizzati **z**, calcola la matrice delle distanze;
→ la funzione **hclust()** che dalla matrice delle distanze genera il clustering gerarchico;

→ la funzione **plot()** che dal clustering gerarchico traccia il dendrogramma.



I principali argomenti delle tre funzioni sono evidenziati negli script successivi e al bisogno sono illustrati nella documentazione che si può richiamare con **help(***nomedellafunzione***)**.

In questo secondo esempio viene illustrato il metodo agglomerativo forse più estesamente impiegato nella pratica, e cioè il clustering gerarchico con il **metodo di Ward**.

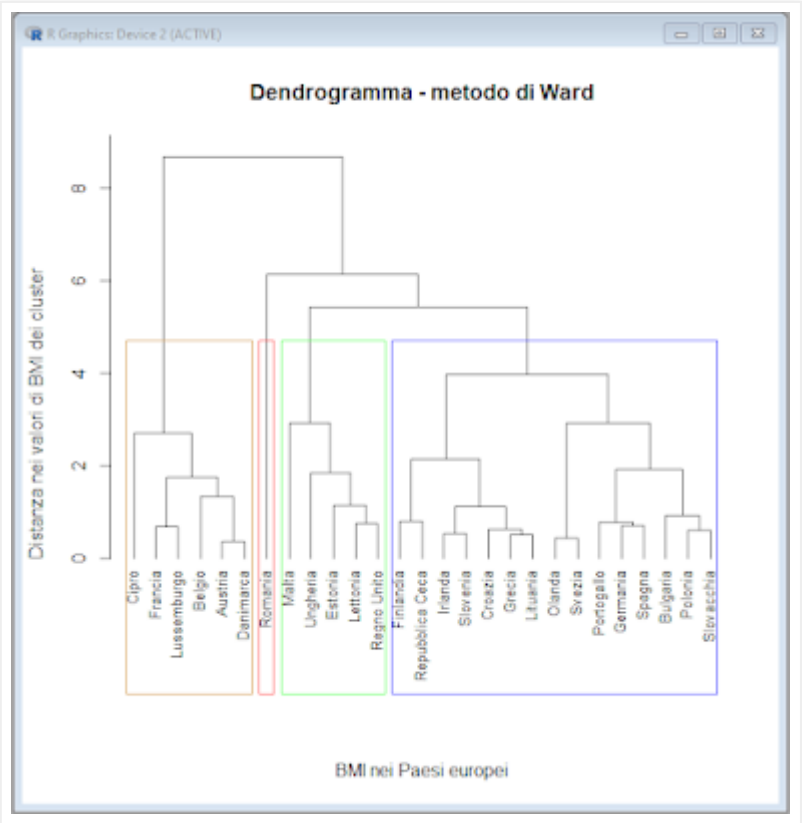
Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# CLUSTERING GERARCHICO con distanza euclidea e metodo di Ward
#
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # calcola la deviata normale standardizzata dei dati
windows() # apre e inizializza una nuova finestra grafica
#
mat <- dist(z, method="euclidean") # calcola la matrice delle distanze
clus <- hclust(mat, method="ward.D2") # effettua il clustering gerarchico
plot(clus, cex=0.8, hang=-1, main="Dendrogramma - metodo di Ward", xlab="BMI nei
Paesi europei", sub="", ylab="Distanza nei valori di BMI dei cluster") # traccia il
dendrogramma
#
rect.hclust(clus, k=4, border=c("goldenrod", "red", "green", "blue")) # evidenzia 4
gruppi/cluster
#
```

Qui le funzioni necessarie diventano quattro:
→ la matrice delle distanze **mat** è calcolata con la funzione **dist()** sui dati standardizzati **z** impiegando l'argomento **method="euclidean"** che prevede di misurare la distanza tra due punti con il teorema di Pitagora e può assumere in alternativa uno dei seguenti valori: "*euclidean*", "*maximum*", "*manhattan*", "*canberra*", "*binary*", "*minkowski*";

→ il clustering gerarchico **clus** è effettuato con la funzione **hclust()** sulla matrice delle distanze **mat** impiegando l'argomento **method=ward.D2** che specifica come costruire i cluster, e può assumere in alternativa uno dei seguenti valori: "*ward.D*", "*ward.D2*", "*single*", "*complete*", "*average*" (= *UPGMA*), "*mcquitty*" (= *WPGMA*), "*median*" (= *WPGMC*), "*centroid*" (= *UPGMC*), il valore di default per l'argomento **method** è "*complete*";
→ a partire dal clustering gerarchico contenuto nell'oggetto **clus** con la funzione **plot()** viene tracciato il dendrogramma. L'argomento **hang=-1** fa si che rami del dendrogramma ed etichette risultino allineati in basso. Potete togliere questo argomento se preferite il posizionamento dei rami del dendrogramma e delle etichette previsto di default (vedere il dendrogramma precedente);
→ con la funzione **rect.hclust()** e l'argomento **k=4** viene stabilito il numero dei gruppi da evidenziare nel dendrogramma, mentre i riquadri che li delimitano sono tracciati con i colori definiti nell'argomento **border=c(...)**.

Questo è il dendrogramma ottenuto con il metodo di Ward:



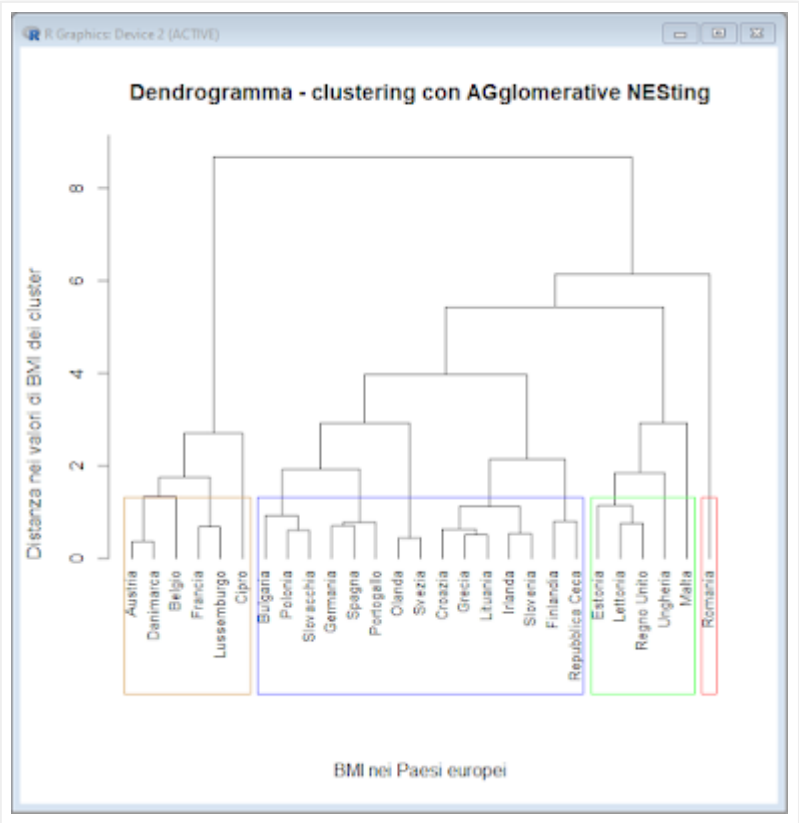
Ora vediamo come il clustering agglomerativo può essere realizzato anche con il metodo di **AGglomerative NESting** sostituendo la funzione **hclust()** dello script precedente con la funzione **agnes()**.

Copiate questo nuovo script, incollatelo nella `Console di R` e premete `↲ Invio`:

```
# CLUSTERING GERARCHICO agglomerativo (AGglomerative NESting)
#
library(cluster) # carica il pacchetto
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # calcola la deviata normale standardizzata dei dati
windows() # apre e inizializza una nuova finestra grafica
#
mat <- dist(z, method="euclidean") # calcola la matrice delle distanze
clus <- agnes(mat, method="ward") # effettua il clustering gerarchico agglomerativo (AGNES)
plot(clus, which.plots=2, cex=0.8, hang=-1, main="Dendrogramma - clustering con
AGglomerative NESting", xlab="BMI nei Paesi europei", sub="", ylab="Distanza nei valori
```

di BMI dei cluster") # traccia il dendrogramma
#
rect.hclust(clus, k=4, border=c("goldenrod","blue","green","red")) # evidenzia 4 gruppi/cluster
#

Le differenze di codice necessarie per realizzare il clustering agglomerativo in questo script rispetto al precedente sono poche, ma ovviamente determinanti:
→ è richiesto il pacchetto **cluste**r;
→ per il clustering gerarchico agglomerativo, viene impiegata la funzione **agnes()** che prevede per l'argomento **method** i valori "*average*", "*single*", "*complete*", "*ward*", "*weighted*", "*gaverage*";
→ nella funzione **plot()** l'argomento **which.plot=2** serve per rappresentare solamente il dendrogramma, mentre l'argomento **hang=-1** di nuovo allinea rami del dendrogramma ed etichette in basso.



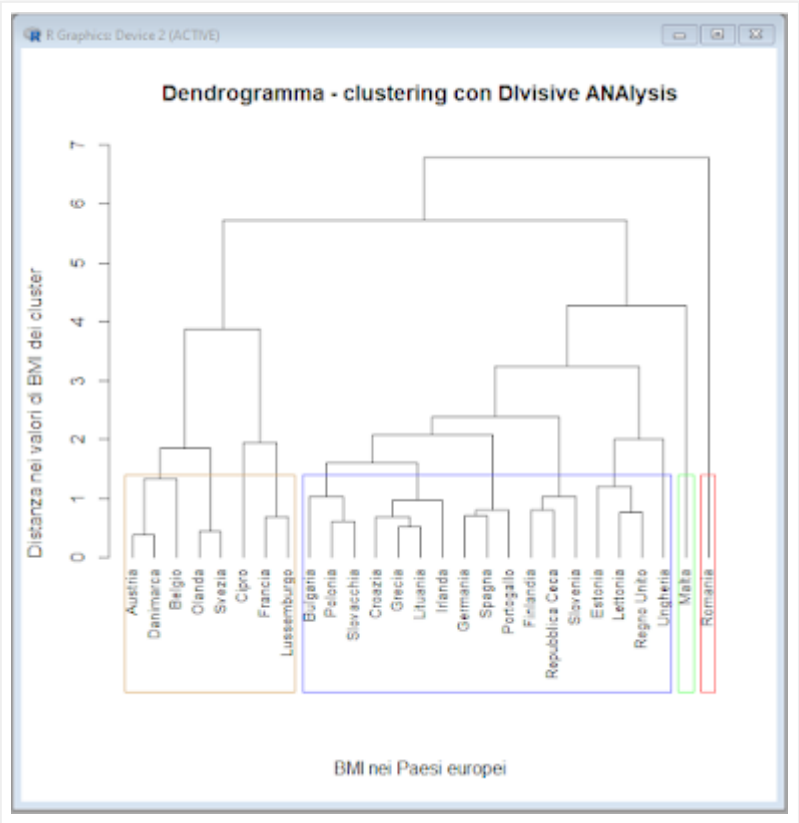
In effetti come vedete impiegando l'argomento **method="ward"** i cluster sono uguali a quelli ottenuti con il metodo di Ward dello script precedente.

Per realizzare il clustering divisivo (top-down) con la **DIvisive ANAlysis** impieghiamo quest'altro script, incollatelo nella `Console di R` e premete `↵ Invio`:

CLUSTERING GERARCHICO divisivo (DIvisive ANAlysis)
#
library(cluster) # carica il pacchetto
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # calcola la deviata normale standardizzata dei dati
windows() # apre e inizializza una nuova finestra grafica
#
mat <- dist(z, method="euclidean") # calcola la matrice delle distanze
clus <- diana(mat) # effettua il clustering gerarchico divisivo (DIANA)
plot(clus, which.plots=2, cex=0.8, hang=-1, main="Dendrogramma - clustering con DIvisive ANAlysis", xlab="BMI nei Paesi europei", sub="", ylab="Distanza nei valori di

BMI dei cluster") # traccia il dendrogramma
#
rect.hclust(clus, k=4, border=c("goldenrod","blue","green","red")) # evidenzia 4 gruppi/cluster
#

Nella documentazione della funzione **diana()** per il clustering con la DIvisive ANAlysis si legge che questo metodo "*... is probably unique in computing a divisive hierarchy, whereas most other software for hierarchical clustering is agglomerative*".



La funzione prevede un unico algoritmo di clusterizzazione, per cui si riduce all'espressione **diana(mat)** mentre il resto del codice, a parte naturamente titolo ed etichette, resta identico a quello dello script precedente.

Per evidenziare le differenze tra i risultati ottenuti con il clustering divisivo e quelli ottenuti con il precedente clustering agglomerativo, impieghiamo la possibilità piuttosto interessante di effettuare il **confronto tra due dendrogrammi**, sia graficamente sia calcolando un indice numerico di corrispondenza.

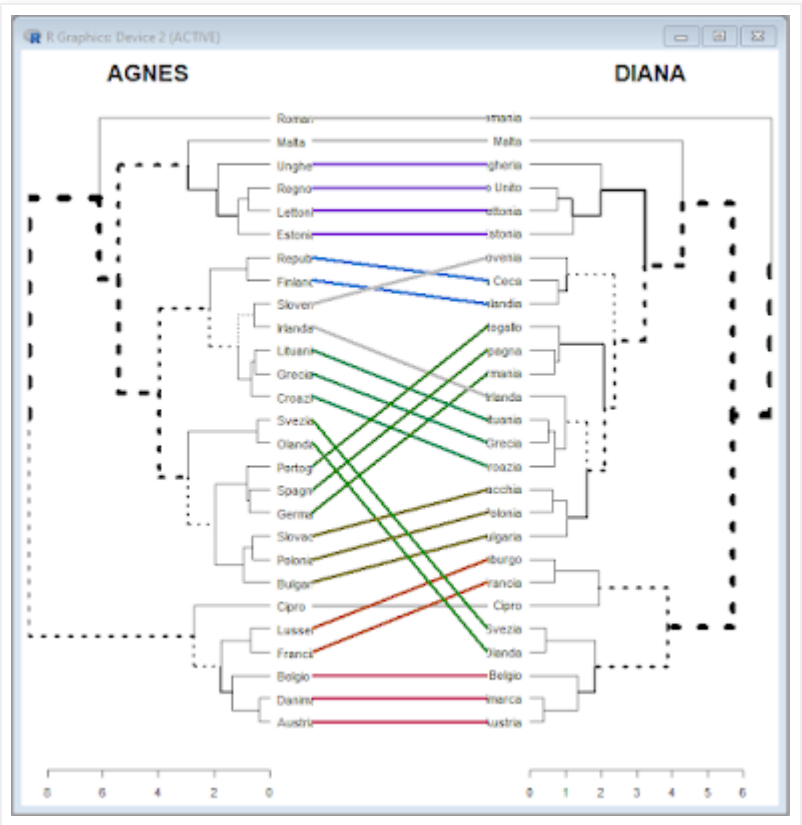
Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

CLUSTERING GERARCHICO confronto tra i dendrogrammi AGNES e DIANA
#
library(dendextend) # carica il pacchetto
library(cluster) # carica il pacchetto
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # calcola la deviata normale standardizzata dei dati
windows() # apre e inizializza una nuova finestra grafica
#
mat <- dist(z, method="euclidean") # calcola la matrice delle distanze
clusAgnes <- agnes(mat, method="ward") # effettua il clustering gerarchico agglomerativo (AGNES)

clusDiana <- diana(mat) # effettua il clustering gerarchico divisivo (DIANA)
dendAgnes <- as.dendrogram(clusAgnes) # predispone il dendrogramma
dendDiana <- as.dendrogram(clusDiana) # predispone il dendrogramma
tanglegram(dendAgnes, dendDiana, main_left="AGNES", main_right="DIANA") # traccia il grafico di confronto tra i due dendrogrammi
entanglement(dendAgnes, dendDiana) # calcola la diffferenza tra i due dendrogrammi
#

Dopo avere caricato anche il pacchetto **dendextend**, i passaggi prevedono di:
→ calcolare come sempre la matrice delle distanze con la funzione **dist()**;
→ effettuare il clustering gerarchico con i due algoritmi che vogliamo confrontare, che sono rispettivamente **agnes()** per il clustering agglomerativo e **diana()** per il clustering divisivo;
→ con la funzione **as.dendrogram()** estrarre e salvare i due dendrogrammi che vogliamo confrontare;
→ impiegare la funzione **tanglegram()** per tracciare il grafico che confronta visivamente i due dendrogrammi;
→ impiegare la funzione **entanglement()** per calcolare l'indice numerico che misura la differenza tra due dendrogrammi.

Il grafico risultante



mette a confronto i due dendrogrammi.

L'indice numerico che misura la differenza tra i due dendrogrammi, che nella documentazione del pacchetto è denominato "*entanglement*", ha un valore compreso tra `0` e `1`, dove:
→ `0` indica che non vi è alcuna differenza tra i due dendrogrammi;
→ `1` indica che i due dendrogrammi non hanno nulla in comune e sono totalmente differenti.

Nel nostro caso abbiamo

```
> entanglement(dendAgnes, dendDiana) # calcola la differenza tra i due dendrogrammi
[1] 0.09419058
```

quindi il valore ottenuto `0.09419058` conferma che i due dendrogrammi sono molto simili, anche se determinano un raggruppamento in cluster lievemente diverso.

Trovate il seguito e le strategie alternative di clustering e di analisi dei dati multivariati nei post:
→ Analisi delle componenti principali
→ Analisi dei gruppi (clustering non gerarchico)
→ Analisi dei gruppi (clustering non esclusivo)

---

[1] Nonostante alla base dell'analisi dei gruppi vi sia un'idea semplice e logica, vi sono numerosi modi per realizzarla, infatti esistono:
→ *metodi gerarchici*, che danno luogo a una suddivisione ad albero (dendrogramma) in base alla distanza tra i singoli oggetti dell'insieme;
→ *metodi non gerarchici*, nei quali l'appartenenza di un oggetto (dell'insieme) ad uno specifico sottoinsieme/gruppo/cluster viene stabilita sulla base della sua distanza dal centro o dalla media dei dati o da un punto rappresentativo del cluster;
→ metodi *bottom-up* noti anche come *metodi agglomerativi* nei quali all'inizio del processo di classificazione ad ogni oggetto viene fatto corrispondere un cluster. In questo stadio gli oggetti sono considerati tutti dissimili tra di loro. Al passaggio successivo i due oggetti più simili sono raggruppati nello stesso cluster. Il numero dei cluster risulta quindi pari al numero di oggetti diminuito di uno. Il procedimento viene ripetuto ciclicamente, fino ad ottenere (all'ultimo passaggio) un unico cluster;
→ metodi *top-down* noti anche come *metodi divisivi* nei quali inizialmente tutti gli oggetti sono considerati come appartenenti ad un unico cluster, che viene via via suddiviso in cluster fino ad avere un numero di cluster uguale al numero degli oggetti;
→ *metodi esclusivi*, che prevedono che un oggetto possa appartenere esclusivamente a un cluster;
→ *metodi non esclusivi* (*fuzzy*) che prevedono che un oggetto possa appartenere, in modo quantitativamente diverso, a più di un cluster.

[2] Vedere il post Indice di massa corporea (BMI).

[3] Trovate la documentazione nel manuale di riferimento del pacchetto Package 'cluster'. URL consultato il 05/01/2023.

[4] Trovate la documentazione nel manuale di riferimento del pacchetto Package ‘dendextend’. URL consultato il 05/01/2023.

sabato 18 febbraio 2023

Ordinamento dei dati

Ordinare i dati è una attività semplice e sovente necessaria per la quale **R** offre molte soluzioni.

Qui vediamo le funzioni di ordinamento contenute nel *pacchetto base* di **R** - il pacchetto di funzioni statistiche e grafiche che viene installato con il programma **R** - e le funzioni contenute nei due *pacchetti aggiuntivi* **dplyr** e **data.table**, che sono quelle più frequentemente utilizzate. I due pacchetti aggiuntivi devono ovviamente essere scaricati e installati dal **CRAN**.

Per proseguire è necessario:
→ effettuare il download del file di dati `peso_altezza.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `peso_altezza.csv` (assicuratevi che il file sia effettivamente salvato in formato testo e con l'estensione `.csv`).

id;sesso;anni;peso_kg;altezza_m
MT;M;69;76;1,78
GF;F;56;63;
MC;F;53;71;1,60
SB;M;28;73;1,78
FE;F;61;54;1,54
AB;M;46;92;1,84
RF;F;31;81;1,56

Iniziamo con la funzione **order()** contenuta nel pacchetto base precisando subito che con questa funzione la tabella originaria viene sì mostrata nell'ordine desiderato ma non viene modificata, e il nuovo ordine si applica salvando i dati ordinati in una tabella che:
→ può avere lo stesso nome della tabella originaria, nel qual caso questa viene sovrascritta e quindi risulterà effettivamente modificata;
→ può avere un nuovo nome, nel qual caso la tabella originaria resta immodificata, come facciamo qui (noi che vogliamo essere prudenti).

Vediamo la cosa con questo semplice script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ORDINAMENTO DATI
#
mydata <- read.table("c:/Rdati/peso_altezza.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata # mostra la tabella originaria
#
mydata[order(mydata$id),] # ordina e mostra la tabella ordinata
#
mydata # la tabella originaria resta immodificata
#
newdata <- mydata[order(mydata$id),] # ordina e salva in una nuova tabella
#
newdata # mostra la nuova tabella
#
```

Ecco quindi cosa accade, questa è la tabella originaria **mydata** importata:

```
> mydata # mostra la tabella originaria
  id sesso anni peso_kg altezza_m
1 MT      M   69      76      1.78
2 GF      F   56      63        NA
3 MC      F   53      71      1.60
4 SB      M   28      73      1.78
5 FE      F   61      54      1.54
6 AB      M   46      92      1.84
7 RF      F   31      81      1.56
```

Con la funzione **order()** la tabella **mydata** viene ordinata in ordine crescente in base alla variabile **id** e viene mostrata nella `Console di R`:

```
> mydata[order(mydata$id),] # ordina e mostra la tabella ordinata
  id sesso anni peso_kg altezza_m
6 AB      M   46      92      1.84
5 FE      F   61      54      1.54
2 GF      F   56      63        NA
3 MC      F   53      71      1.60
1 MT      M   69      76      1.78
7 RF      F   31      81      1.56
4 SB      M   28      73      1.78
```

Tuttavia se verifichiamo la tabella originaria **mydata** richiamandone il nome vediamo che non è stata modificata:

```
> mydata # la tabella originaria resta immodificata
  id sesso anni peso_kg altezza_m
1 MT      M   69      76      1.78
2 GF      F   56      63        NA
3 MC      F   53      71      1.60
4 SB      M   28      73      1.78
5 FE      F   61      54      1.54
6 AB      M   46      92      1.84
7 RF      F   31      81      1.56
```

Per ottenere una tabella con i dati ordinati si ordina nuovamente la tabella **mydata** salvando il risultato (**<-**) in una nuova tabella che denominiamo **newdata**:

```
> newdata <- mydata[order(mydata$id),] # ordina e salva in una nuova tabella
```

Se ora verifichiamo il contenuto della nuova tabella **newdata** richiamandone il nome, vediamo che contiene i dati ordinati:

```
> newdata # mostra la nuova tabella
  id sesso anni peso_kg altezza_m
6 AB      M   46      92      1.84
5 FE      F   61      54      1.54
2 GF      F   56      63        NA
3 MC      F   53      71      1.60
1 MT      M   69      76      1.78
7 RF      F   31      81      1.56
4 SB      M   28      73      1.78
```

Una volta chiarito questo punto, molto importante dal punto di vista pratico, possiamo precedere con gli script. Copiate il primo, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ORDINAMENTO ordina per nome della tabella$variabile
#
mydata <- read.table("c:/Rdati/peso_altezza.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata # mostra la tabella originaria
#
mydata[order(mydata$peso_kg),] # ordina il peso in ordine crescente
#
mydata[order(-mydata$peso_kg),] # ordina il peso in ordine decrescente
#
mydata[order(mydata$sesso, -mydata$peso_kg),] # ordina il sesso in ordine crescente, il peso in ordine decrescente
#
mydata[order(mydata$id),] # ordina id in ordine crescente
#
mydata[order(-mydata$id),] # ordina id in ordine decrescente
#
mydata[order(-xtfrm(mydata$id)),] # ordina id in ordine decrescente
#
mydata[order(-xtfrm(mydata$sesso), mydata$peso_kg),] # ordina il sesso in ordine decrescente, il peso in ordine crescente
#
mydata[order(mydata$altezza_m, na.last=TRUE),] # ordina l'altezza in ordine crescente, NA in coda
#
mydata[order(mydata$altezza_m, na.last=FALSE),] # ordina l'altezza in ordine crescente, NA in testa
#
mydata[order(-mydata$altezza_m, na.last=NA),] # ordina l'altezza in ordine decrescente, esclude NA
#
mydata # la tabella originaria resta immodificata
#
```

I commenti riportati in ciascuna riga di codice sono di per sè esplicativi.

La cosa interessante accade quando si tenta di ordinare la variabile **mydata$id** in ordine decrescente con **mydata[order(-mydata$id),]**, questo infatti è il risultato:

```
> mydata[order(-mydata$id),] # ordina id in ordine decrescente
Errore in -mydata$id : argomento non valido per l'operatore unario
```

L'errore è legato al fatto che **R** non ammette davanti a una variabile di tipo testo il segno meno (**-**) e viene corretto nella riga successiva con la funzione **xtfrm()** che nell'help è definita come "*A generic auxiliary function that produces a numeric vector which will sort in the same order as x*".

Questo è il risultato della nuova sintassi che corregge l'errore che impediva l'ordinamento in senso decrescente della variabile testo:

```
> mydata[order(-xtfrm(mydata$id)),] # ordina id in ordine decrescente
  id sesso anni peso_kg altezza_m
4 SB     M   28      73      1.78
7 RF     F   31      81      1.56
```

```
1 MT      M    69       76       1.78
3 MC      F    53       71       1.60
2 GF      F    56       63         NA
5 FE      F    61       54       1.54
6 AB      M    46       92       1.84
```

Come potete constatare dai risultati che vedete nella `Console di R` le ultime tre righe di codice illustrano come impiegare l'argomento **na.las**t per controllare il trattamento dei dati mancanti (NA):
→ se **na.last=TRUE** i dati mancanti sono identificati con NA e messi in coda ai dati;
→ se **na.last=FALSE** i dati mancanti sono identificati con NA e posti in testa ai dati;
→ se **na.last=NA** i dati mancanti sono rimossi;

Copiate questo secondo script, incollatelo nella `Console di R` e premete `↲ Invio`:

```
# ORDINAMENTO ordina per nome della variabile
#
mydata <- read.table("c:/Rdati/peso_altezza.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata # mostra la tabella originaria
attach(mydata) # consente di impiegare direttamente i nomi delle variabili di mydata
#
mydata[order(id),] # ordina id in ordine crescente
#
mydata[order(-xtfrm(id)),] # ordina id in ordine decrescente
#
mydata[order(-anni),] # ordina gli anni in ordine decrescente
#
mydata[order(sesso, -altezza_m),] # ordina il sesso in ordine crescente, l'altezza in ordine crescente
#
mydata[order(-xtfrm(sesso), altezza_m),] # ordina il sesso in ordine decrescente, l'altezza in ordine decrescente
#
detach(mydata) # termina l'impiego diretto dei nomi delle variabili
#
mydata # la tabella originaria resta immodificata
#
```

Impieghiamo sempre la funzione **order()** contenuta nel pacchetto {base} ma risulta immediatamente evidente la differenza: nello script precedente le variabili erano indicate con *nomedellatabella$nomedellavariabile*, mentre ora dopo avere riportato nella seconda riga di codice

```
attach(mydata)
```

possiamo fare riferimento alle variabili riportando semplicemente il *nomedellavariabile*. Mentre le regole di ordinamento rimangono ovviamente le stesse, alla fine dello script viene riportato

```
detach(mydata)
```

per ripristinare la condizione di default [2].

Concludiamo quanto riguarda la funzione **order()** contenuta nel pacchetto {base} con questo terzo script, incollatelo nella `Console di R` e premete `↲ Invio`:

```
# ORDINAMENTO ordina per numero della colonna
```

```
#
mydata <- read.table("c:/Rdati/peso_altezza.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata # mostra la tabella originaria
#
mydata[order(mydata[,4]),] # ordina la colonna 4 in ordine crescente
#
mydata[order(-mydata[,4]),] # ordina la colonna 4 in ordine decrescente
#
mydata[order(mydata[,1]),] # ordina la colonna 1 in ordine crescente
#
mydata[order(-xtfrm(mydata[,1])),] # ordina la colonna 1 in ordine decrescente
#
mydata[order(mydata[,2], mydata[,5]),] # ordina la colonna 2 in ordine crescente, la colonna 5 in ordine crescente
#
mydata[order(-xtfrm(mydata[,2]), -mydata[,5]),] # ordina la colonna 2 in ordine decrescente, la colonna 5 in ordine decrescente
#
mydata # la tabella originaria resta immodificata
#
```

La differenza rispetto ai due primi script consiste, questa volta, semplicemente nell'ordinare la tabella impiegando il numero della colonna.

Vediamo ora le regole per l'ordinamento previste dalla funzione **arrange()** del pacchetto **dplyr**.

Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ORDINAMENTO con il pacchetto "dplyr"
#
library(dplyr) # carica il pacchetto
mydata <- read.table("c:/Rdati/peso_altezza.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata # mostra la tabella originaria
#
arrange(mydata, altezza_m) # ordina l'altezza in ordine crescente
#
arrange(mydata, desc(altezza_m)) # ordina l'altezza in ordine decrescente
#
arrange(mydata, sesso, desc(id)) # ordina il sesso in ordine crescente, id in ordine decrescente
#
mydata # la tabella originaria resta immodificata
#
```

Le cose da notare in merito alla funzione **arrange()** sono:
→ viene specificato ogni volta il nome della tabella da ordinare, per cui il nome della variabile risulta abbreviato;
→ i dati mancanti sono identificati con `NA` e messi in coda ai dati;
→ l'ordinamento di una variabile testo in ordine discendente con l'opzione **desc()** non incontra il problema che avevamo riscontrato con la funzione **order()**;
→ la tabella originaria resta immodificata.

Passiamo infine a vedere le regole per l'ordinamento previste dalla funzione **setorder()** del pacchetto **read.table**.

Copiate questo script, incollatelo nella `Console di R` e premete ↵ `Invio`:

```
# ORDINAMENTO con il pacchetto "data.table"
#
library(data.table) # carica il pacchetto
mydata <- read.table("c:/Rdati/peso_altezza.csv", header=TRUE, sep=";", dec=",") # importa i dati
mydata # mostra la tabella originaria
#
setorder(mydata, altezza_m) # ordina l'altezza in ordine crescente
mydata # mostra la tabella ordinata
#
setorder(mydata, -altezza_m) # ordina l'altezza in ordine decrescente
mydata # mostra la tabella ordinata
#
setorder(mydata, sesso, -id) # ordina il sesso in ordine crescente, id in ordine decrescente
mydata # mostra la tabella ordinata
#
```

Qui le cose da notare sono:
→ contrariamente a quanto visto finora, con la funzione **setorder()** la tabella originaria viene realmente ordinata e inoltre:
→ se si impiega l'argomento **na.last=TRUE** i dati mancanti sono identificati con `NA` e messi in coda ai dati;
→ se si impiega l'argomento **na.last=FALSE** i dati mancanti sono identificati con `NA` e posti in testa ai dati;
→ la funzione accetta per l'argomento **na.last** solamente i valori **TRUE** e **FALSE** con valore di default **FALSE** (che è quello qui esemplificato).

Conclusione: se volete evitare l'impiego di un pacchetto aggiuntivo, trovate nella funzione **order()** del pacchetto {base} che viene installato con il programma **R** tutte le modalità di ordinamento che si possono desiderare, anche se con qualche arzigogolo. Se invece vi va bene installare pacchetti aggiuntivi, la funzione **arrange()** e la funzione **setorder()**, con differenze veramente minime tra loro, offrono il vantaggio di impiegare una sintassi semplificata e più immediata. Il punto delicato è il fatto che **setorder()** ordina realmente la tabella originaria, mentre potrebbe essere più prudente e anche più ordinato lasciare, come accade con **arrange()**, la tabella originaria immodificata e al bisogno salvare i dati ordinati in una tabella con un nome differente. Ma ovviamente si tratta di scelte personali e insindacabili.

---

[1] Parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14
6	7	18	12
11	8	17	13
7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

[2] In termini un poco più tecnici: con la funzione **attach(***database***)** il database è collegato al percorso di ricerca di **R**, questo significa che quando si opera su una variabile **R** è informato del database che la contiene, e pertanto è possibile accedere alla variabile semplicemente fornendone il nome. Con la funzione **detach(***database***)** il database viene rimosso dal percorso di ricerca di **R**.

domenica 5 marzo 2023

Gestione dei dati mancanti

Per illustrare il tema, che è strettamente collegato all'ordinamento dei dati [1], vediamo tre cose:
→ come individuare i dati mancanti;
→ come i dati mancanti - identificati in **R** con la sigla `NA` (`Not Available`) - possono bloccare l'esecuzione di calcoli sulle variabili numeriche che li contengono e come in questo caso sia possibile omettere selettivamente dai calcoli i dati `NA`;
→ come al bisogno sia possibile eliminare per intero da un vettore o matrice o tabella [2] i casi con dati `NA`.

Per proseguire dovete, seguendo le istruzioni fornite alla pagina Dati:
→ effettuare il download del file `importa_csv.csv`
→ salvare il file nella cartella `C:\Rdati\`

In alternativa copiate le otto righe riportate qui sotto, incollatele in un editor di file di testo aggiungendo un `↲ Invio` al termine dell'ultima riga, e salvate il tutto in `C:\Rdati\` in un file di testo denominato `importa_csv.csv` (attenzione: assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`):

```
id;sesso;anni;peso_kg;altezza_m
MT;M;69;76;1,78
GF;F;56;63;
MC;F;53;71;1,60
SB;M;28;73;1,78
FE;F;61;54;1,54
AB;M;46;92;1,84
RF;F;31;81;1,56
```

Vediamo come individuare i dati mancanti. Copiate e incollate nella `Console di R` questo script e premete `↲ Invio`:

```
# IDENTIFICA E CONTEGGIA I DATI MANCANTI
#
# importa i dati, notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",",
row.names="id")
#
mydata # mostra i dati importati
#
is.na(mydata) # identifica i dati mancanti
#
colSums(is.na(mydata)) # conteggia i dati mancanti per colonna/variabile
#
which(is.na(mydata$altezza_m)) # identifica la posizione dei dati mancanti
#
```

Dopo avere importato i dati, richiamando il nome dell'oggetto **mydata** che li contiene questi sono mostrati con la sigla `NA` riportata automaticamente da **R** in corrispondenza di ciascuno dei dati mancanti.

```
> mydata # mostra i dati importati
   sesso anni peso_kg altezza_m
```

```
MT      M    69       76       1.78
GF      F    56       63         NA
MC      F    53       71       1.60
SB      M    28       73       1.78
FE      F    61       54       1.54
AB      M    46       92       1.84
RF      F    31       81       1.56
```

Nel caso di piccole tabelle come questa non serve altro: ma nel caso di estesi database ci servono funzioni che forniscano qualche soluzione più pratica e immediata.

Impieghiamo quindi la funzione **is.na()** che identifica nella tabella **mydata** i dati mancanti riportando `TRUE` in corrispondenza di ciascuno di essi

```
> is.na(mydata) # identifica i dati mancanti
   sesso  anni peso_kg altezza_m
MT FALSE FALSE   FALSE     FALSE
GF FALSE FALSE   FALSE      TRUE
MC FALSE FALSE   FALSE     FALSE
SB FALSE FALSE   FALSE     FALSE
FE FALSE FALSE   FALSE     FALSE
AB FALSE FALSE   FALSE     FALSE
RF FALSE FALSE   FALSE     FALSE
```

I dati della tabella **mydata** trasformati nei corrispettivi valori logici con la funzione **is.na()** possono allora essere impiegati come argomento della funzione **colSums()** che effettua il conteggio dei dati mancanti per ciascuna colonna/variabile

```
colSums(is.na(mydata)) # conteggia i dati mancanti per colonna/variabile
    sesso      anni   peso_kg altezza_m
        0         0         0         1
```

A questo punto abbiamo individuato in `altezza_m` la variabile con i dati mancanti e abbiamo conteggiato il loro numero.

Non ci resta quindi che impiegare i dati della variabile **mydata$altezza_m** trasformati nei corrispettivi valori logici con **is.na()** come argomento della funzione **which()** per identificare la posizione dei dati mancanti

```
> which(is.na(mydata$altezza_m)) # identifica la posizione dei dati mancanti
[1] 2
```

che nel nostro caso risultano essere il dato della riga numero 2 della variabile `mydata$altezza_m`.

Per l'effetto che i dati mancanti possono determinare sui calcoli copiate e incollate nella `Console di R` questo script e premete `↲ Invio`:

```
# ESEMPIO DI MEDIA ARITMETICA NON COMPUTABILE A CAUSA DI DATI MANCANTI
#
# importa i dati, notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",",
row.names="id")
#
mydata # mostra i dati importati
#
mean(mydata$peso_kg) # calcola la media aritmetica dei valori di peso
```

mean(mydata$altezza_m) # calcola la media aritmetica dei valori di altezza
#

Il risultato che compare nella `Console di R` è il seguente:

```
> mydata # mostra i dati importati
   sesso anni peso_kg altezza_m
MT     M    69      76      1.78
GF     F    56      63        NA
MC     F    53      71      1.60
SB     M    28      73      1.78
FE     F    61      54      1.54
AB     M    46      92      1.84
RF     F    31      81      1.56
> #
> mean(mydata$peso_kg) # calcola la media aritmetica dei valori di peso
[1] 72.85714
> mean(mydata$altezza_m) # calcola la media aritmetica dei valori di altezza
[1] NA
```

Come potete vedere il calcolo della media effettuato con la funzione **mean()** ha avuto successo solamente per il peso. Per l'altezza in luogo del valore della media è stata riportata la sigla `NA`, evidente conseguenza del valore mancante dell'altezza nel caso `GF`.

Copiate e incollate nella `Console di R` queste due ulteriori righe di codice e premete `↲ Invio`:

la corretta gestione dei dati mancanti consente di calcolare la media dei valori di altezza
#
mean(mydata$peso_kg) # ricalcola la media aritmetica dei valori di peso
mean(mydata$altezza_m, na.rm=TRUE) # ricalcola la media aritmetica dei valori di altezza
#

Il risultato dei calcoli della media che compare nella `Console di R` ora è questo:

```
> mean(mydata$peso_kg) # ricalcola la media aritmetica dei valori di peso
[1] 72.85714
> mean(mydata$altezza_m, na.rm=TRUE) # ricalcola la media aritmetica dei valori di
altezza
[1] 1.683333
```

Quindi con l'aggiunta nella funzione **mean()** dell'argomento **na.rm=TRUE**, che rimuove dal calcolo i dati mancanti (in questo caso uno solo, ma potrebbero essere anche molti), è stato reso possibile il calcolo della media anche per l'altezza.

Con la funzione **na.omit()** è infine possibile eliminare definitivamente i casi con dati mancanti: anche un solo dato mancante/campo vuoto comporta la cancellazione dell'intera riga/caso. Copiate e incollate nella `Console di R` questo script e premete `↲ Invio`:

ELIMINA I CASI CON DATI MANCANTI
#
importa i dati, notare / invece di \ su windows
mydata <- read.table("C:/Rdati/importa_csv.csv", header=TRUE, sep=";", dec=",", row.names="id")
#
mydata # mostra i dati importati
#

```
newdata <- na.omit(mydata) # elimina da mydata i casi con dati mancanti
newdata # mostra i dati dopo eliminazione dei casi con dati mancanti
#
mean(newdata$peso_kg) # calcola la media aritmetica dei valori di peso
mean(newdata$altezza_m) # calcola la media aritmetica dei valori di altezza
#
```

Dai dati importati nella tabella **mydata** mediante la funzione **na.omit()** viene generata una nuova tabella denominata **newdata** dalla quale sono esclusi i casi con dati mancanti. Con l'eliminazione dalla tabella del caso `GF` i calcoli vanno subito a buon fine sia per il peso sia per l'altezza.

```
> mydata # mostra i dati importati
   sesso anni peso_kg altezza_m
MT     M    69      76      1.78
GF     F    56      63        NA
MC     F    53      71      1.60
SB     M    28      73      1.78
FE     F    61      54      1.54
AB     M    46      92      1.84
RF     F    31      81      1.56
> #
> newdata <- na.omit(mydata) # elimina da mydata i casi con dati mancanti
> newdata # mostra i dati dopo eliminazione dei casi con dati mancanti
   sesso anni peso_kg altezza_m
MT     M    69      76      1.78
MC     F    53      71      1.60
SB     M    28      73      1.78
FE     F    61      54      1.54
AB     M    46      92      1.84
RF     F    31      81      1.56
> #
> mean(newdata$peso_kg) # calcola la media aritmetica dei valori di peso
[1] 74.5
> mean(newdata$altezza_m) # calcola la media aritmetica dei valori di altezza
[1] 1.683333
```

Da notare che impiegando la funzione **na.omit()** la media dei valori di peso, che con lo script precedente era `72.85714` kg, ora è cambiata ed è diventata `74.5` kg in quanto in seguito all'eliminazione dell'intero caso `GF` è stato eliminato anche il suo valore di peso `63` kg che contribuiva a determinare la media di quest'ultimo.

---

[1] Vedere il post Ordinamento dei dati.

[2] Parliamo di **array** o **vettore** nel caso di dati numerici monodimensionali, disposti su una sola riga,

8	6	11	7

di **matrice** nel caso di **dati numerici** disposti su più righe e più colonne

8	9	15	14

6	7	18	12
11	8	17	13
7	4	19	17

e di **tabella** nei casi in cui il contenuto, disposto su più righe e più colonne, è rappresentato oltre che da **dati numerici**, anche da **testo** e/o **operatori logici**

M	7	9	VERO
F	3	12	VERO
F	5	10	FALSO

lunedì 8 maggio 2023

Analisi dei gruppi (clustering non gerarchico)

L'analisi dei gruppi si applica a *dati multivariati* ed è un metodo statistico di *tassonomia numerica* che riveste un ruolo importante nella *analisi esplorativa dei dati*.

L'obiettivo dell'**analisi dei gruppi** (*cluster analysis* o *clustering*) è concettualmente semplice: verificare la possibile esistenza, in un insieme di oggetti, di sottoinsiemi di oggetti particolarmente simili tra loro (gruppi/cluster).

Nonostante alla base dell'analisi dei gruppi vi sia un'idea semplice e logica, vi sono numerosi modi per realizzarla [1], qui ci occupiamo dell'implementazione del *clustering non gerarchico* con *metodi esclusivi* nelle due versioni:
→ clustering con il **metodo di MacQueen** (*k-means*);
→ clustering con il **metodo di Rousseew** (*k-medoids*).

Come dati impieghiamo i valori di BMI (indice di massa corporea) rilevati a livello europeo alcuni anni fa e pubblicati dall'Istat [2].

Per proseguire è necessario:
→ effettuare il download del file di dati `bmi.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7

Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2

Inoltre è necessario scaricare dal **CRAN** il pacchetto aggiuntivo **cluster** [3], il pacchetto aggiuntivo **factoextra** [4] e il pacchetto aggiuntivo **ggplot2** [5].

Iniziamo con il clustering con il metodo di MacQueen che impiega l'algoritmo *k-means*.

Copiate questo script, incollatelo nella `Console di R` e premete `↵ Invio`:

```
# CLUSTERING (NON GERARCHICO) con il metodo di MacQueen (k-means)
#
library(cluster) # carica il pacchetto
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # standardizza le variabili
windows() # apre e inizializza una nuova finestra grafica
#
myclust <- kmeans(z, 4, algorithm="MacQueen", nstart=50) # genera i 4 gruppi/cluster
clusplot(z, myclust$cluster, color=TRUE, labels=2, lines=0, main="Grafico dei cluster -
metodo di MacQueen (k.means)", xlab="Componente principale 1", sub="",
ylab="Componente principale 2", cex=0.6, col.txt="black", col.p="black") # traccia il grafico
dei cluster
#
```

Con prima cosa viene caricato il pacchetto **cluster**, quindi i dati sono importati in **mydata**.

Con la funzione **scale()** della terza riga per ciascun dato viene calcolata la *deviata normale standardizzata z*. Questa funzione calcola per i dati di ciascuna colonna/variabile la `media` e la `deviazione standard`, poi calcola per ciascun dato `x` la corrispondente deviata normale standardizzata `z` come

```
z = (x - media) / deviazione standard
```

I valori di `z` sono salvati nel nuovo oggetto qui denominato per comodità mnemonica **z**.

Se nella `Console di R` digitate **z**

```
> z
```

sono mostrati i dati standardizzati, in coda ai quali sono riportate la `media`

```
attr(,"scaled:center")
 sottopeso    normale sovrappeso      obeso
  2.103704  44.537037  36.240741  17.111111
```

e la `deviazione standard`

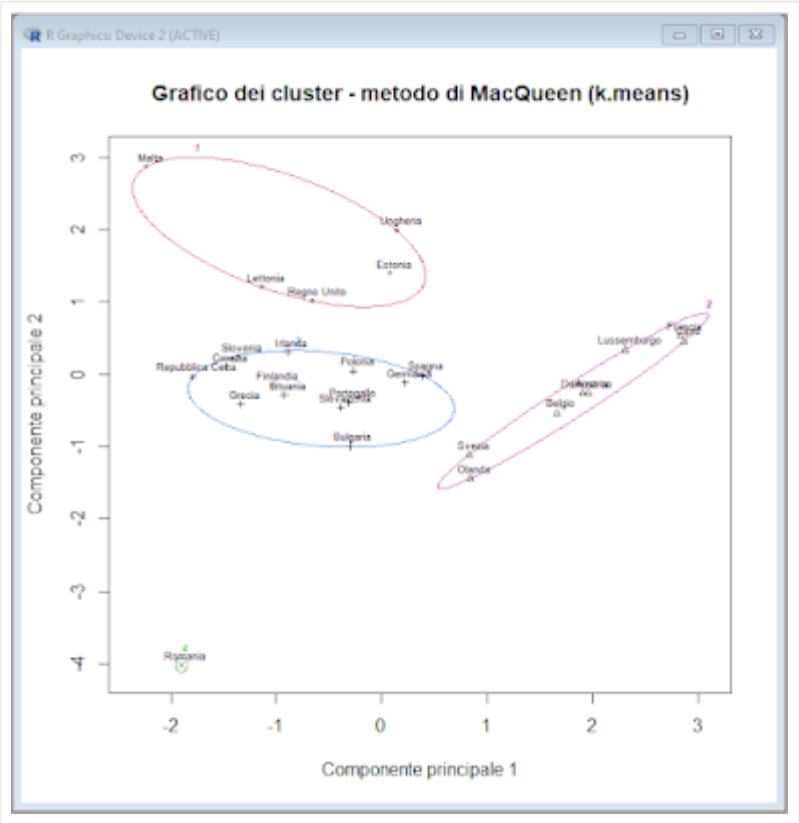
```
attr(,"scaled:scale")
 sottopeso    normale sovrappeso      obeso
 0.6111033  3.3717318  2.8989732  3.2322097
```

impiegate per effettuare la standardizzazione dei dati.

Quindi viene aperta e inizializzata una nuova finestra grafica con **windows()**.

La funzione **kmeans()** impiega i dati standardizzati **z** per generare **4** cluster, impiegando l'algoritmo **"MacQueen"** e 50 iterazioni (**nstart=50**) dell'algoritmo di scelta iniziale dei cluster. I risultati sono salvati in **myclust**.

I risultati del clustering salvati in **myclust** sono quindi impiegati per tracciare il grafico con la funzione **clusplot()** e i seguenti argomenti:
→ l'oggetto **z** contenente i dati standardizzati;
→ l'oggetto **myclust$cluster** contenente i cluster generati con **kmeans()**;
→ **color=TRUE** per riportare i cluster in colore;
→ **labels=2** per riportare la numerazione assegnata ai cluster;
→ **lines=0** per non riportare le linee che collegano i cluster;
→ **sub=""** che elimina il sottotitolo previsto di default dalla funzione;
→ **cex=0.6** per rimpicciolire i caratteri del testo applicato;
→ **col.txt** che definisce il colore del testo che compare all'interno del grafico;
→ **col.p** che definisce il colore impiegato per rappresentare i punti all'interno del grafico.



Per semplicità nella funzione **clusplot()** sono stati lasciati i valori di default per numerosi altri argomenti, digitate **help(*clusplot*)** nella `Console di R` per un approfondimento del tema.

Se nella `Console di R` digitate **mydata[order(-mydata$normale),]**

```
> mydata[order(-mydata$normale),]
```

sono mostrati i dati ordinati in ordine decrescente per la colonna che contiene la percentuale di soggetti con peso normale

```
                sottopeso normale sovrappeso obeso
Danimarca             2.2    50.0       32.9  14.9
Austria               2.4    49.6       33.3  14.7
Francia               3.2    49.6       31.9  15.3
Lussemburgo           2.8    49.3       32.4  15.6
Olanda                1.6    49.0       36.0  13.3
```

Svezia	1.8	48.3	35.9	14.0
Belgio	2.7	48.0	35.3	14.0
Cipro	3.9	47.8	33.8	14.5
Germania	1.8	46.1	35.2	16.9
Spagna	2.2	45.4	35.7	16.7
Portogallo	1.8	44.6	36.9	16.6
Finlandia	1.2	44.1	36.4	18.3
Estonia	2.2	43.9	33.5	20.4
Bulgaria	2.2	43.8	39.2	14.8
Slovacchia	2.1	43.6	38.0	16.3
Polonia	2.4	42.9	37.5	17.2
Romania	1.3	42.9	46.4	9.4
Lituania	1.9	42.5	38.3	17.3
Irlanda	1.9	42.3	37.0	18.7
Regno Unito	2.1	42.2	35.6	20.1
Repubblica Ceca	1.1	42.1	37.6	19.3
Ungheria	2.9	41.9	34.0	21.2
Lettonia	1.7	41.8	35.2	21.3
Slovenia	1.6	41.8	37.4	19.2
Grecia	1.9	41.3	39.4	17.3
Croazia	1.9	40.7	38.7	18.7
Malta	2.0	37.0	35.0	26.0

e potete vedere subito come - giusto per chiarire con un esempio - Danimarca, Austria, Francia, Lussemburgo, Olanda, Svezia, Belgio e Cipro, graficamente incluse nello stesso cluster, numericamente si distinguono dalle altre nazione per avere la maggior percentuale di soggetti con peso normale e contemporaneamente una ridotta percentuale di soggetti obesi.

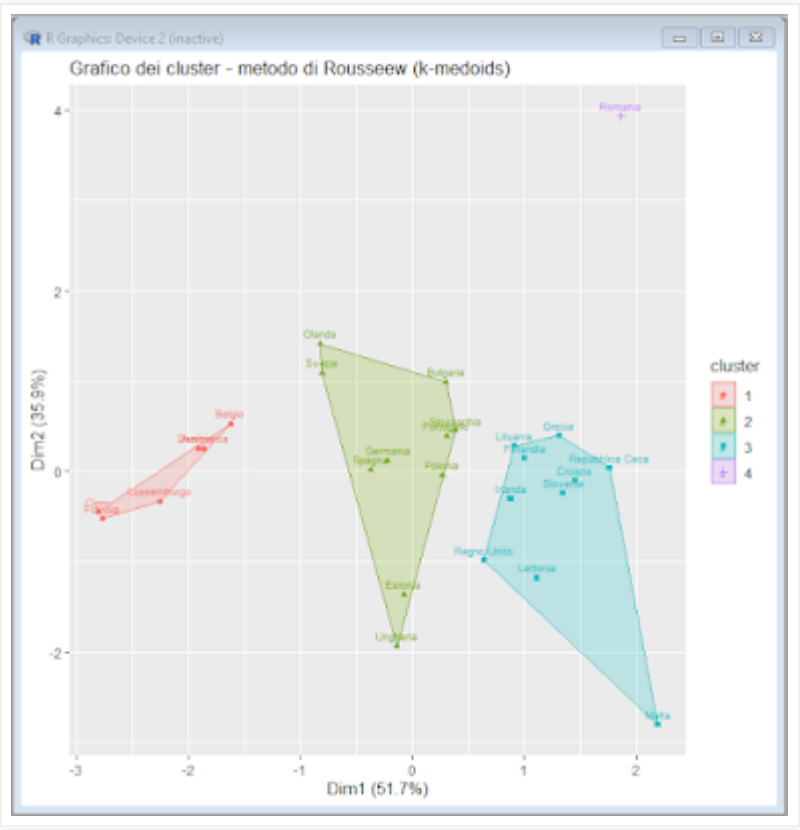
Vediamo ora il clustering con il metodo di Rousseew che impiega l'algoritmo *k-medoids*.

```
# CLUSTERING NON GERARCHICO con il metodo di Rousseew (k-medoids)
#
library(factoextra) # carica il pacchetto
library(cluster) # carica il pacchetto
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
windows() # apre e inizializza una nuova finestra grafica
#
myclust <- pam(mydata, 4, metric=c("euclidean"), stand=TRUE) # genera un oggetto pam
che contiene i dati dei cluster
fviz_cluster(myclust, myclust$cluster, labelsize=7, main = "Grafico dei cluster - metodo di
Rousseew (k-medoids)") # grafico dei cluster per le prime due componenti principali
#
windows() # apre e inizializza una nuova finestra grafica
distance <- get_dist(mydata, method="euclidean", stand=TRUE) # genera la matrice delle
distanze
fviz_dist(distance, order=TRUE, show_labels=TRUE, gradient = list(low="#00AFBB",
mid="white", high="#FC4E07")) + theme(axis.text.x=element_text(angle=90,
vjust=0.3))+ theme(axis.text.y=element_text(angle=0, vjust=0.3)) # grafico della matrice
delle distanze
#
windows() # apre e inizializza una nuova finestra grafica
fviz_nbclust(mydata, FUNcluster=cluster::pam, method="wss") # calcola il numero ottimale
di cluster
#
```

I preliminari sono gli stessi dello script precedente a parte il fatto che in questo caso viene impiegato anche il pacchetto **factoextra** che a sua volta si basa sul pacchetto **ggplot2** che deve anch'esso essere stato installato.

La prima funzione utilizzata è **pam()** che deriva il suo nome da "*Partitioning Around Medoids*", impiega l'algoritmo di clustering dei dati *k-medoids* (un versione robusta dell'algoritmo *k-means*) e prevede come argomenti:
→ i dati **mydata** da impiegare;
→ il numero di cluster, in questo caso **4**, in cui classificare i dati;
→ la metrica da impiegare che può essere **"euclidean"** o **"manhattan"**;
→ **stand=TRUE** che impone la standardizzazione dei dati, che nel caso del clustering con il metodo di MacQueen che abbiamo visto sopra deve invece essere eseguita in via preliminare con la funzione **scale()**.

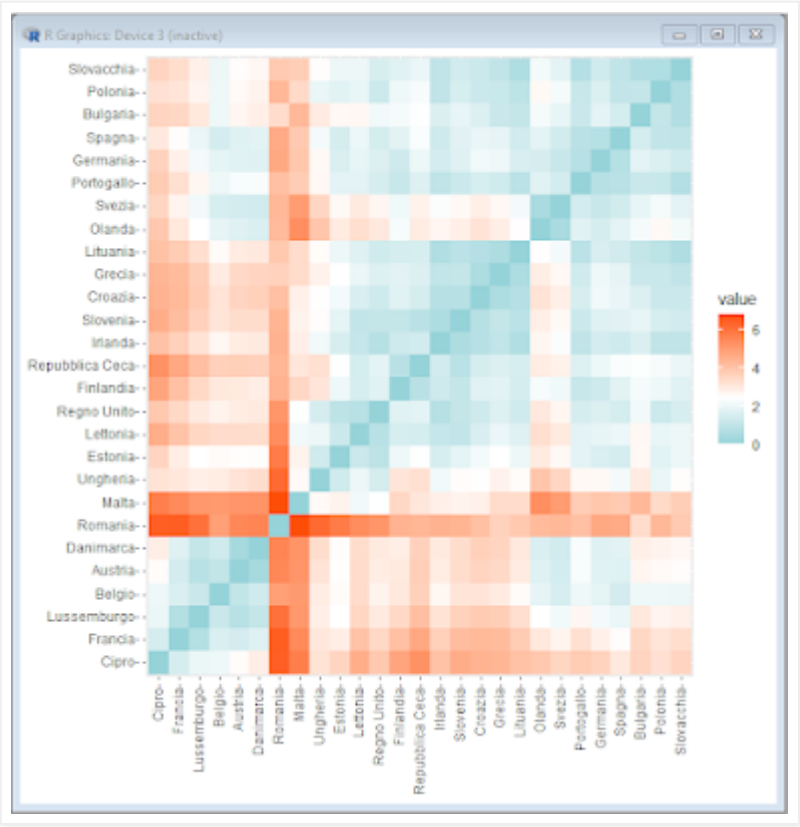
Con la successiva funzione **fviz_cluster()** è generato il grafico dei cluster (**myclust$cluster**) ottenuti con l'algoritmo *k-medoids* e contenuti nell'oggetto **myclus**t.



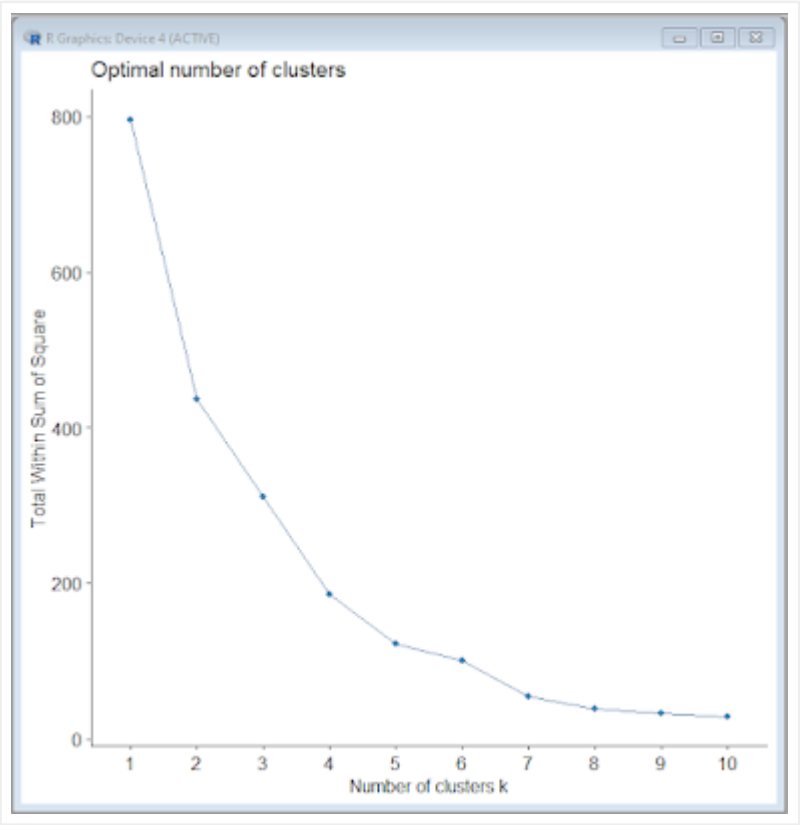
La matrice delle distanze **distance** generata con la funzione **get_dist()** sui dati (**mydata**), previa la loro standardizzazione (**stand=TRUE**), viene impiegata dalla funzione **fviz_dist()** per generare il grafico delle distanze tra gli oggetti clusterizzati, un grafico facoltativo e accessorio rispetto al precedente, ma che potrebbe interessare.

In diagonale compare in azzurro la distanza `0` della relazione di identità degli oggetti con se stessi. Il colore azzurro si va attenuando via via che inizia la dissimilarità tra gli oggetti, fino a trasformarsi in un rosso sempre più intenso al suo progressivo aumentare.

Il colore rosso, ad esempio, diventa la dominante nell'incrocio della Romania con tutte le altre Nazioni, a conferma della sua peculiarità: se guardate i dati, ha la percentuale in assoluto più elevata di sovrappeso (46.4%) e la percentuale in assoluto più bassa di obesi (9.4%). In termini di dissimilarità dalle restanti Nazioni, la Romania è immediatamente seguita da Malta, come risulta evidente anche nel grafico dei cluster.



Infine con la funzione **fviz_nbclust()** viene generato il grafico che in teoria consentirebbe di valutare il numero ottimale di cluster da imporre nella funzione **pam()**.



Il numero ottimale di cluster dovrebbe corrispondere ad un punto in cui cambia la curvatura: qui vediamo il primo in corrispondenza di 2 cluster, e il secondo in corrispondenza di 4 cluster. Personalmente non ho mai trovato particolarmente utili gli automatismi di questo genere, preferisco ragionare sul quadro generale, ma lascio al lettore le valutazioni del caso.

Trovate il seguito e le strategie alternative di clustering e di analisi dei dati multivariati nei post:
→ Analisi delle componenti principali

→ Analisi dei gruppi (clustering gerarchico)
→ Analisi dei gruppi (clustering non esclusivo)

---

[1] Nonostante alla base dell'analisi dei gruppi vi sia un'idea semplice e logica, vi sono numerosi modi per realizzarla, infatti esistono:
→ *metodi gerarchici*, che danno luogo a una suddivisione ad albero (dendrogramma) in base alla distanza tra i singoli oggetti dell'insieme;
→ *metodi non gerarchici*, nei quali l'appartenenza di un oggetto (dell'insieme) ad uno specifico sottoinsieme/gruppo/cluster viene stabilita sulla base della sua distanza dal centro o dalla media dei dati o da un punto rappresentativo del cluster;
→ metodi *bottom-up* noti anche come *metodi agglomerativi* nei quali all'inizio del processo di classificazione ad ogni oggetto viene fatto corrispondere un cluster. In questo stadio gli oggetti sono considerati tutti dissimili tra di loro. Al passaggio successivo i due oggetti più simili sono raggruppati nello stesso cluster. Il numero dei cluster risulta quindi pari al numero di oggetti diminuito di uno. Il procedimento viene ripetuto ciclicamente, fino ad ottenere (all'ultimo passaggio) un unico cluster;
→ metodi *top-down* noti anche come *metodi divisivi* nei quali inizialmente tutti gli oggetti sono considerati come appartenenti ad un unico cluster, che viene via via suddiviso in cluster fino ad avere un numero di cluster uguale al numero degli oggetti;
→ *metodi esclusivi*, che prevedono che un oggetto possa appartenere esclusivamente a un cluster;
→ metodi *non esclusivi* (*fuzzy*) che prevedono che un oggetto possa appartenere, in modo quantitativamente diverso, a più di un cluster.

[2] Vedere il post Indice di massa corporea (BMI).

[3] Trovate la documentazione nel manuale di riferimento del pacchetto: Package ‘cluster’. URL consultato il 06/01/2023.

[4] Trovate la documentazione nel manuale di riferimento del pacchetto: Package ‘factoextra’. URL consultato il 06/01/2023.

[5] Trovate la documentazione nel manuale di riferimento del pacchetto: Package ‘ggplot2’. URL consultato il 06/01/2023.

venerdì 21 luglio 2023

Analisi delle componenti principali

L'**analisi delle componenti principali** (Principal Component Analysis da cui l'acronimo **PCA**), insieme all'analisi dei gruppi (cluster analysis o clustering), fa parte delle tecniche di *analisi esplorativa dei dati* e viene impiegata nell'analisi di dati multivariati [1].

Dato un campione descritto da `n` variabili, le sue componenti principali [2, 3]:
→ sono ciascuna una combinazione lineare delle `n` variabili;
→ sono nello stesso numero `n` delle variabili, essendo la *prima componente principale* la variabile formata dalla combinazione lineare delle variabili originali che spiega la maggior quantità di varianza, la *seconda componente principale* la variabile che spiega la maggior quantità di varianza in ciò che rimane una volta rimosso l'effetto della prima componente principale, e così via, fino a spiegare tutta la varianza osservata;
→ sono non correlate, come dimostrato dal fatto che i coefficienti di correlazione tra le componenti principali sono uguali a `0` (zero);
→ concentrano di solito la maggior parte dell'informazione del campione in 2 componenti principali, con una perdita di informazione contenuta;
→ consentono una migliore introspezione nei dati in quanto 2 dimensioni sono facilmente rappresentabili e analizzabili.

Per un corretto utilizzo della **PCA** è opportuno ricordare che:
→ la tecnica prevede l'assunto che i dati abbiano una distribuzione gaussiana;
→ tra le variabili deve esistere una correlazione lineare;
→ le componenti principali sono una combinazione lineare delle variabili analizzate;
→ le componenti principali non sono invarianti rispetto alla scala, pertanto è necessario standardizzare le variabili che descrivono il campione sul quale sono calcolate le componenti principali;
→ a causa della standardizzazione di cui al punto precedente le nuove variabili calcolate non possono essere interpretate come i dati originari, dei quali sono una "sintesi" lineare [2, 3].

Come dati impieghiamo i valori di BMI (indice di massa corporea) rilevati a livello europeo alcuni anni fa e pubblicati dall'Istat. Si tratta di una tabella che, per ciascuna delle Nazioni elencate, riporta la percentuale di soggetti sottopeso, con peso normale, sovrappeso e obesi [4].

Per proseguire è necessario:
→ effettuare il download del file di dati `bmi.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

```
Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
```

Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7
Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2

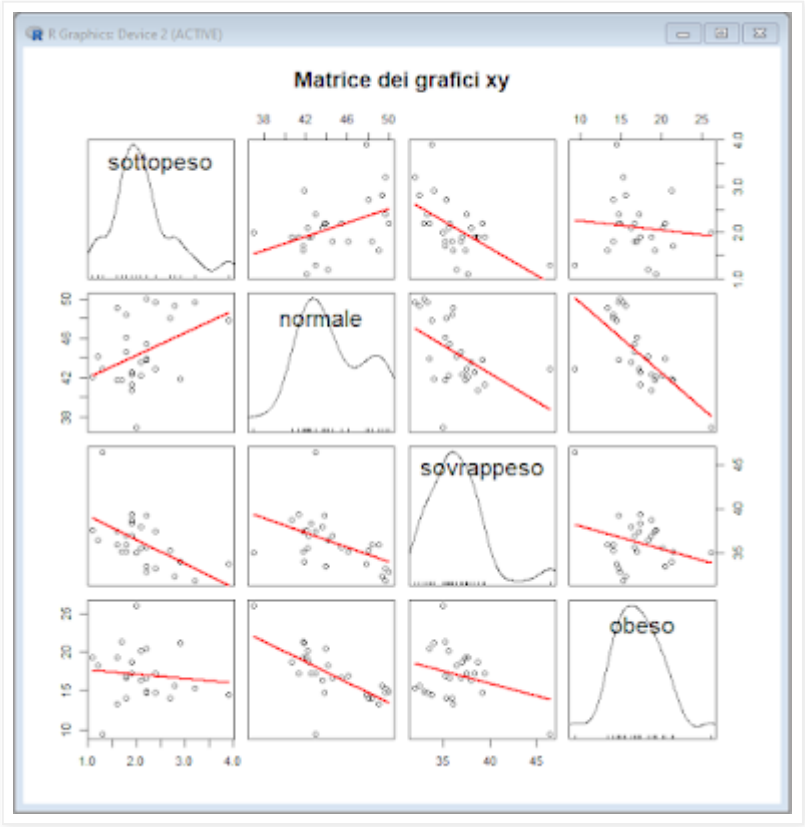
Infine è necessario scaricare dal **CRAN** il pacchetto aggiuntivo **car** e il pacchetto aggiuntivo **cluster**.

Copiate e incollate questo primo script nella `Console di R` e premete `↵ Invio`:

```
# ANALISI DELLE COMPONENTI PRINCIPALI - ispezione dei dati
#
library(car) # carica il pacchetto necessario per la grafica
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
windows() # apre e inizializza una nuova finestra grafica
#
scatterplotMatrix(~sottopeso+normale+sovrappeso+obeso, regLine=list(method=lm,
lty=1, lwd=2, col="red"), smooth=FALSE, diagonal=list(method="density",
kernel="gaussian", adjust=1), col = "black", main="Matrice dei grafici xy",
data=mydata) # traccia il grafico di dispersione xy che incrocia tutte le variabili
#
```

Le prime tre righe servono solamente a caricare il pacchetto necessario per la grafica e a importare i dati e si commentano da sole.

Per effettuare una prima ispezione dei dati impieghiamo la funzione **scatterplotMatrix** che genera i grafici `xy` che incrociano le quattro variabili numeriche della tabella, aggiunge con **regLine=-...** la retta di regressione e riporta sulla diagonale (**diagonal=...**) i kernel density plot (**method="density"**) delle distribuzioni dei dati.



Il grafico documenta una proporzionalità inversa tra la percentuale di soggetti con peso normale e le percentuali di soggetti sovrappeso e obesi (all'aumentare della percentuale di soggetti con peso normale le ultime due diminuiscono), e una proporzionalità diretta tra soggetti con peso normale e soggetti sottopeso.

L'idea ora è di semplificare il quadro impiegando la **PCA**. Per effettuarne l'analisi numerica copiate e incollate questo secondo script nella `Console di R` e premete `↵ Invio`:

```
# ANALISI DELLE COMPONENTI PRINCIPALI - analisi numerica
#
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
#
options(scipen=999) # esprime i numeri in formato fisso
#
z <- scale(mydata) # calcola la deviata normale standardizzata dei dati
cbind(mydata, z) # mostra i dati originari e la loro deviata normale standardizzata z
#
mypca <- princomp(z, cor=TRUE) # calcola le componenti principali sulle variabili standardizzate
cbind(mydata, mypca$scores[,1:4]) # mostra i dati originari e le componenti principali
#
round(cor(cbind(mydata, mypca$scores[,1:4])), digits=3) # calcola i coefficienti di
correlazione delle  componenti principali e dei dati originari
summary(mypca) # sintetizza l'importanza delle componenti principali
mypca$sdev^2 # si valuta il criterio di Kraiser che prevede varianza > 1
#
options(scipen=0) # ripristina la notazione scientifica
#
```

Dopo avere importato i dati, con la funzione **options()** e l'argomento **scipen=999** imponiamo di rappresentare i risultati numerici in formato fisso (di default **R** prevede la notazione scientifica) per facilitarne la lettura [5].

Le componenti principali non sono invarianti rispetto alla scala: pertanto è necessario standardizzare le variabili che descrivono il campione sul quale sono calcolate le componenti principali. Questo viene fatto calcolando per i dati di ciascuna variabile la `media` e la `deviazione standard`, poi calcolando per ciascuno dato `x` la corrispondente deviata normale standardizzata `z` come

```
z = (x - media) / deviazione standard
```

Con la funzione **scale()** [6] sono calcolate automaticamente le deviate normali standardizzate `z` dei dati e con la funzione **cbind()** accanto ai dati originari (**mydata**) sono riportati in una tabella le rispettive deviate normali standardizzate (**z**).

```
> cbind(mydata, z) # mostra i dati originari e la loro deviata normale standardizzata z
                sottopeso normale sovrappeso obeso    sottopeso     normale  sovrappeso       obeso
Austria               2.4    49.6       33.3  14.7  0.484854650  1.50159124 -1.01440770 -0.74596370
Belgio                2.7    48.0       35.3  14.0  0.975769982  1.02705765 -0.32450826 -0.96253381
Bulgaria              2.2    43.8       39.2  14.8  0.157577761 -0.21859302  1.02079566 -0.71502512
Cipro                 3.9    47.8       33.8  14.5  2.939431313  0.96774095 -0.84193284 -0.80784088
Croazia               1.9    40.7       38.7  18.7 -0.333337572 -1.13800184  0.84832080  0.49157977
Danimarca             2.2    50.0       32.9  14.9  0.157577761  1.62022463 -1.15238759 -0.68408653
Estonia               2.2    43.9       33.5  20.4  0.157577761 -0.18893467 -0.94541776  1.01753574
Finlandia             1.2    44.1       36.4  18.3 -1.478806681 -0.12961797  0.05493644  0.36782542
Francia               3.2    49.6       31.9  15.3  1.793962204  1.50159124 -1.49733732 -0.56033218
Germania              1.8    46.1       35.2  16.9 -0.496976016  0.46354901 -0.35900323 -0.06531479
Grecia                1.9    41.3       39.4  17.3 -0.333337572 -0.96005175  1.08978561  0.05843955
Irlanda               1.9    42.3       37.0  18.7 -0.333337572 -0.66346826  0.26190627  0.49157977
Lettonia              1.7    41.8       35.2  21.3 -0.660614460 -0.81176000 -0.35900323  1.29598302
Lituania              1.9    42.5       38.3  17.3 -0.333337572 -0.60415156  0.71034091  0.05843955
Lussemburgo           2.8    49.3       32.4  15.6  1.139408427  1.41261619 -1.32486245 -0.46751642
Malta                 2.0    37.0       35.0  26.0 -0.169699127 -2.23536077 -0.42799317  2.75009660
Olanda                1.6    49.0       36.0  13.3 -0.824252904  1.32364114 -0.08304345 -1.17910392
Polonia               2.4    42.9       37.5  17.2  0.484854650 -0.48551816  0.43438113  0.02750097
Portogallo            1.8    44.6       36.9  16.6 -0.496976016  0.01867378  0.22741130 -0.15813055
Regno Unito           2.1    42.2       35.6  20.1 -0.006060683 -0.69312661 -0.22102334  0.92471998
Repubblica Ceca       1.1    42.1       37.6  19.3 -1.642445126 -0.72278496  0.46887610  0.67721129
Romania               1.3    42.9       46.4   9.4 -1.315168237 -0.48551816  3.50443367 -2.38570880
Slovacchia            2.1    43.6       38.0  16.3 -0.006060683 -0.27790972  0.60685599 -0.25094631
Slovenia              1.6    41.8       37.4  19.2 -0.824252904 -0.81176000  0.39988616  0.64627270
Spagna                2.2    45.4       35.7  16.7  0.157577761  0.25594057 -0.18652837 -0.12719197
Svezia                1.8    48.3       35.9  14.0 -0.496976016  1.11603270 -0.11753842 -0.96253381
Ungheria              2.9    41.9       34.0  21.2  1.303046871 -0.78210165 -0.77294290  1.26504444
```

Se invece nella `Console di R` digitate **z**

```
> z
```

sono mostrati i soli dati standardizzati, in coda ai quali sono riportate la `media`

```
attr(,"scaled:center")
 sottopeso     normale sovrappeso       obeso
  2.103704   44.537037   36.240741   17.111111
```

e la `deviazione standard`

```
attr(,"scaled:scale")
 sottopeso     normale sovrappeso       obeso
 0.6111033   3.3717318   2.8989732   3.2322097
```

impiegate per effettuare la standardizzazione dei dati.

La funzione **princomp()** calcola le componenti principali sulle variabili standardizzate (**z**) impiegando la matrice di correlazione (**cor=TRUE**).

Quindi mediante la funzione **cbind()** i dati originari (**mydata**) sono combinati con le quattro colonne contenenti i valori della prima componente principale (**mypca$scores[,1]**), della seconda componente principale (**mypca$scores[,2]**), della terza componente principale (**mypca$scores[,3]**) e della quarta componente principale (**mypca$scores[,4]**) e sono così visualizzati:

```
> cbind(mydata, mypca$scores[,1:4]) # mostra i dati originari e le componenti principali
               sottopeso normale sovrappeso obeso     Comp.1      Comp.2      Comp.3         Comp.4
Austria              2.4    49.6       33.3  14.7  1.9568592 -0.25836896 -0.57968660 -0.00008584140
Belgio               2.7    48.0       35.3  14.0  1.6573003 -0.53681742  0.34757492  0.00090365951
Bulgaria             2.2    43.8       39.2  14.8 -0.3065287 -1.00537387  0.76399423  0.00304859104
Cipro                3.9    47.8       33.8  14.5  2.8676299  0.45690930  1.71080991 -0.00033835804
Croazia              1.9    40.7       38.7  18.7 -1.4782942  0.10560338  0.51186402  0.00259366777
Danimarca            2.2    50.0       32.9  14.9  1.9030622 -0.26177070 -0.96268268 -0.00023905961
Estonia              2.2    43.9       33.5  20.4  0.0772148  1.39278775 -0.34702817 -0.00035012011
Finlandia            1.2    44.1       36.4  18.3 -1.0242159 -0.14902653 -1.16654614  0.00167352686
Francia              3.2    49.6       31.9  15.3  2.8277112  0.53285016  0.24082546 -0.00114263428
Germania             1.8    46.1       35.2  16.9  0.2246564 -0.11800412 -0.74397641  0.00094526687
Grecia               1.9    41.3       39.4  17.3 -1.3355279 -0.40371977  0.60210826 -0.01534678459
Irlanda              1.9    42.3       37.0  18.7 -0.8930939  0.30605551  0.05586142 -0.01670407396
Lettonia             1.7    41.8       35.2  21.3 -1.1340925  1.20428268 -0.52824564  0.00064173257
Lituania             1.9    42.5       38.3  17.3 -0.9256566 -0.28759995  0.28705908  0.00249320079
Lussemburgo          2.8    49.3       32.4  15.6  2.3063902  0.33950282 -0.18750396  0.01764811897
Malta                2.0    37.0       35.0  26.0 -2.2351068  2.86564834  0.23542898  0.00007011548
Olanda               1.6    49.0       36.0  13.3  0.8396220 -1.43885040 -1.09559132 -0.01670331763
Polonia              2.4    42.9       37.5  17.2 -0.2687562  0.04968007  0.78163159  0.00190832184
Portogallo           1.8    44.6       36.9  16.6 -0.3104836 -0.39590962 -0.28827071 -0.01655924560
Regno Unito          2.1    42.2       35.6  20.1 -0.6497827  1.00578052  0.06207928  0.00080150891
Repubblica Ceca      1.1    42.1       37.6  19.3 -1.7935828 -0.03541460 -0.91094768  0.02066918164
Romania              1.3    42.9       46.4   9.4 -1.8998797 -4.01115995  1.00294656  0.00752978219
Slovacchia           2.1    43.6       38.0  16.3 -0.3885290 -0.45558544  0.41177524  0.00234357074
Slovenia             1.6    41.8       37.4  19.2 -1.3688625  0.24227630 -0.23901380  0.00198400366
Spagna               2.2    45.4       35.7  16.7  0.3802247 -0.01464965 -0.04421772  0.00108011741
Svezia               1.8    48.3       35.9  14.0  0.8261779 -1.10531918 -0.78871246  0.00153297266
Ungheria             2.9    41.9       34.0  21.2  0.1455442  1.97619331  0.86846432 -0.00039790369
```

Alla riga di codice successiva con la funzione **cor()** sono calcolati i coefficienti di correlazione tra tutte le variabili, arrotondando i risultati a tre decimali con la funzione **round(.... , digits=3)**:

```
> round(cor(cbind(mydata, mypca$scores[,1:4])), digits=3) # calcola i coefficienti di correlazione delle  componenti principali e dei dati originari
           sottopeso normale sovrappeso  obeso Comp.1 Comp.2 Comp.3 Comp.4
sottopeso      1.000   0.412     -0.567 -0.108  0.755  0.340  0.560  0.001
normale        0.412   1.000     -0.481 -0.688  0.902 -0.327 -0.283  0.005
sovrappeso    -0.567  -0.481      1.000 -0.290 -0.679 -0.666  0.309  0.004
obeso         -0.108  -0.688     -0.290  1.000 -0.473  0.877 -0.088  0.005
Comp.1         0.755   0.902     -0.679 -0.473  1.000  0.000  0.000  0.000
Comp.2         0.340  -0.327     -0.666  0.877  0.000  1.000  0.000  0.000
Comp.3         0.560  -0.283      0.309 -0.088  0.000  0.000  1.000  0.000
Comp.4         0.001   0.005      0.004  0.005  0.000  0.000  0.000  1.000
```

L'analisi numerica si conclude con la sintesi fornita dalla funzione **summary()**:

```
> summary(mypca) # sintetizza dell'importanza delle componenti principali
Importance of components:
                          Comp.1    Comp.2    Comp.3        Comp.4
Standard deviation     1.4380589 1.1980885 0.7046275 0.00840918288
Proportion of Variance 0.5170033 0.3588540 0.1241250 0.00001767859
Cumulative Proportion  0.5170033 0.8758574 0.9999823 1.00000000000
```

che fornisce una indicazione importante.

Con la quota di varianza (`Proportion of Variance`) fornita da ciascuna delle quattro componenti principali vediamo che la prima componente principale (`Comp.1`) spiega il 51.7% della variabilità del campione, la seconda componente principale (`Comp.2`) spiega il 35.9% della variabilità del campione, la terza componente principale (`Comp.3`) spiega il 12.4% della variabilità del campione, che sommati danno il 100%, essendo trascurabile la variabilità spiegata dalla quarta componente principale (`Comp.4`).

La quota cumulativa di varianza (`Cumulative Proportion`) ci conferma che nella prima e nella seconda componente principale è contenuta la maggior parte della variabilità del campione (87.6%). Se siamo disposti a sacrificare una piccola quota dell'informazione (varianza) contenuta nel campione (il 12.4%), una analisi del campione può essere effettuata impiegando due sole variabili, la prima componente principale (`Comp.1`) e la seconda componente principale (`Comp.2`), invece delle quattro variabili originali (sottopeso, normale, sovrappeso, obeso), consentendo quindi una migliore introspezione nei dati.

La penultima riga di codice (**mypca$sdev^2**) riporta la varianza spiegata dalle componenti principali, per consentire di applicare il criterio di Kraiser, che prevede che siano significative le componenti principali con una varianza maggiore di 1. Che si confermano essere le prime due:

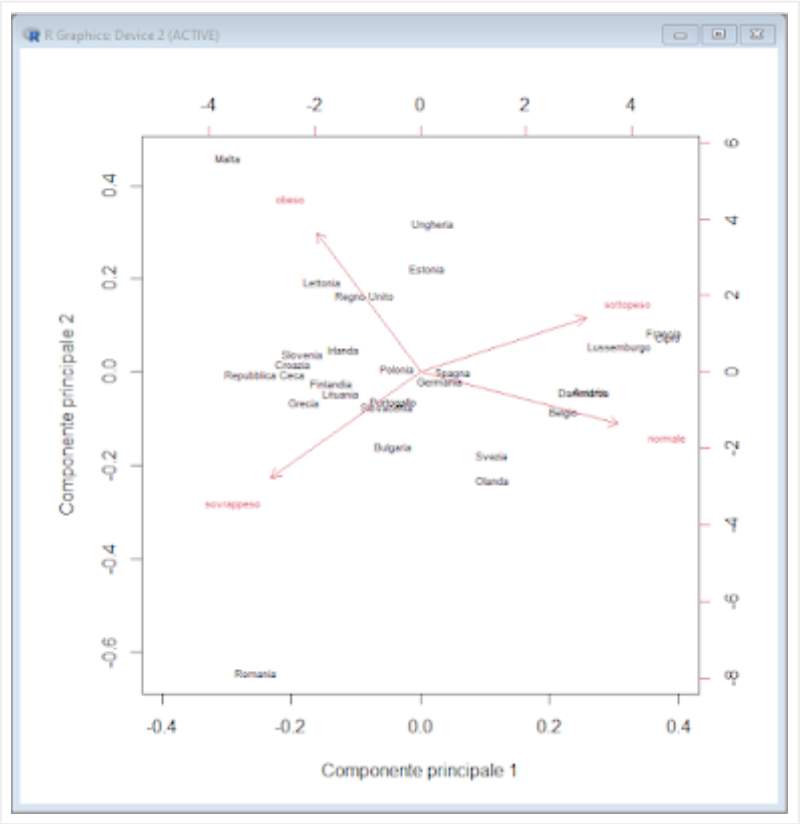
```
> mypca$sdev ^ 2 # si valuta il criterio di Kraiser che prevede varianza > 1
       Comp.1        Comp.2        Comp.3        Comp.4
2.06801325718 1.43541616920 0.49649985926 0.00007071436
```

Infine con **options(scipen=0**) viene ripristinata la notazione scientifica.

Dato che quindi possiamo considerare adeguata un'analisi del campione che impiega solamente la prima e la seconda componente principale, passiamo all'analisi grafica rappresentando le due componenti in un grafico cartesiano `xy` con la funzione **biplot()**.

Copiate e incollate questo terzo script nella `Console di R` e premete `↵ Invio`:

```
# ANALISI DELLE COMPONENTI PRINCIPALI - analisi grafica
#
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # standardizza le variabili
windows() # apre e inizializza una nuova finestra grafica
#
mypca <- princomp(z, cor=TRUE) # calcola le componenti principali sulle variabili standardizzate
#
biplot(mypca, xlim=c(-0.4,0.4), xlab="Componente principale 1", ylab="Componente
principale 2", cex=0.6) # traccia il grafico con i vettori delle componenti principali
#
```



Da notare che nel biplot:
→ l'asse inferiore riporta la scala nella quale sono espressi i valori della componente principale 1
→ l'asse di sinistra riporta la scala nella quale sono espressi i valori della componente principale 2
→ l'asse superiore riporta la scala nella quale sono espressi i pesi dei vettori sulla componente principale 1
→ l'asse di destra riporta la scala nella quale sono espressi i pesi dei vettori sulla componente principale 2
→ la proiezione di un vettore sull'asse di una specifica componente principale fornisce il peso che la variabile rappresentata nel vettore ha su quella componente principale;
→ vettori separati tra loro da angoli piccoli indicano variabili con correlazione positiva;
→ vettori che divergono molto tra loro (fino a 180 gradi) indicano variabili con correlazione negativa.

Dall'ispezione del biplot si ricava che:
→ alcune Nazioni (Svezia, Olanda, Belgio, Danimarca, Austria, Lussemburgo, Francia, Cipro) si proiettano prevalentemente nella dimensione "normale" e in alcuni casi (Francia e Cipro) con una possibile rilevanza della dimensione "sottopeso";
→ altre Nazioni (Regno Unito, Lettonia, Estonia, Ungheria, Malta) si proiettano prevalentemente nella dimensione "obeso", con un dato particolarmente rilevante che riguarda Malta;
→ le Nazioni rimanenti si proiettano nella dimensione del "sovrappeso", con un dato particolarmente rilevante che riguarda la Romania.

Sulla base dei rilievi forniti dall'analisi delle componenti principali è possibile ricondurre le singole osservazioni ai dati originari, mediante i quali, per esempio, si può confermare che Malta, con il 26% di obesi, e la Romania, con il 46.4% di sovrappeso e il 9.4% di obesi, si distaccano in modo importante dalle altre Nazioni.

La cosa molto interessante è che l'analisi delle componenti principali fornisce il collegamento con le tecniche complementari, e precisamente con l'analisi dei gruppi (clustering). Lo vediamo applicando ai dati di BMI il clustering non gerarchico con il metodo di *MacQueen* e l'algoritmo *k-means*.

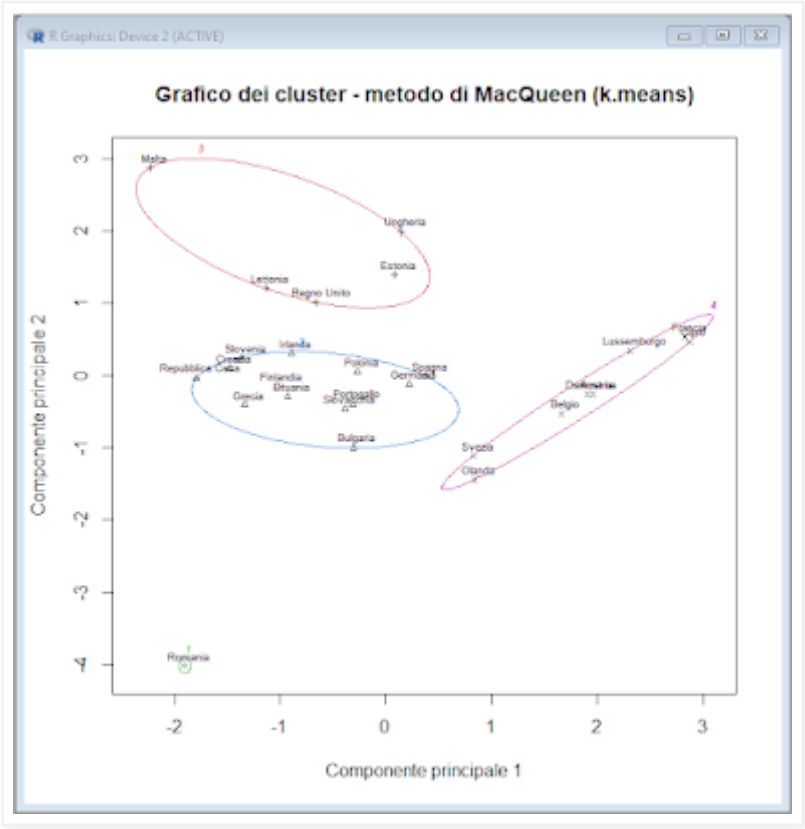
Copiate e incollate questo quarto e ultimo script nella `Console di R` e premete `↵ Invio`:

```
# CLUSTERING (NON GERARCHICO) con il metodo di MacQueen (k-means)
#
library(cluster) # carica il pacchetto
mydata <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";",
row.names="Nazione") # importa i dati
z <- scale(mydata) # standardizza le variabili
windows() # apre e inizializza una nuova finestra grafica
#
myclust <- kmeans(z, 4, algorithm="MacQueen", nstart=50) # genera i 4 gruppi/cluster
clusplot(z, myclust$cluster, color=TRUE, labels=2, lines=0, main="Grafico dei cluster -
metodo di MacQueen (k.means)", xlab="Componente principale 1", sub="",
ylab="Componente principale 2", cex=0.6, col.txt="black", col.p="black") # traccia il grafico
dei cluster
#
```

Dopo i soliti preliminari, con la funzione **kmeans()** sono generati **4** cluster impiegando l'algoritmo **"MacQueen"**.

Quindi viene impiegata per tracciare il grafico la funzione **clusplot()** con i seguenti argomenti:
→ l'oggetto **z** contenente i dati (standardizzati);
→ l'oggetto **myclust$cluster** contenente i cluster;
→ **color=TRUE** per riportare i cluster in colore;
→ **labels=2** per riportare la numerazione assegnata ai cluster;
→ **lines=0** per non riportare le linee che collegano i cluster;
→ **sub=""** che elimina il sottotitolo previsto di default dalla funzione;
→ **cex=0.6** per rimpicciolire i caratteri del testo applicato;
→ **col.txt** che definisce il colore del testo che compare all'interno del grafico;
→ **col.p** che definisce il colore impiegato per rappresentare i punti all'interno del grafico.

Per semplicità nella funzione **clusplot()** sono stati lasciati i valori di default per numerosi altri argomenti, digitate **help(*clusplot*)** nella `Console di R` per un approfondimento del tema.



La distribuzione delle Nazioni che si vede in questo grafico si sovrappone perfettamente a quella realizzata con l'analisi delle componenti principali e in aggiunta identifica quattro cluster ben separati l'uno dall'altro e ben individuati.

In un caso ancora abbastanza semplice come questo possiamo anche pensare di validare i risultati fin qui ottenuti rianalizzando i dati originali: per questo li ordiniamo in senso decrescente prima per la prevalenza di peso normale, poi per la prevalenza di obesi, infine per la prevalenza di sovrappeso, con questo risultato

	A	B	C	D	E
1	Nazione	sottopeso	normale	sovrappeso	obeso
2	Danimarca	2,2	50,0	32,9	14,9
3	Austria	2,4	49,6	33,3	14,7
4	Francia	3,2	49,6	31,9	15,3
5	Lussemburgo	2,8	49,3	32,4	15,6
6	Olanda	1,6	49,0	36,0	13,3
7	Svezia	1,8	48,3	35,9	14,0
8	Belgio	2,7	48,0	35,3	14,0
9	Cipro	3,9	47,8	33,8	14,5
10	Malta	2,0	37,0	35,0	26,0
11	Lettonia	1,7	41,8	35,2	21,3
12	Ungheria	2,9	41,9	34,0	21,2
13	Estonia	2,2	43,9	33,5	20,4
14	Regno Unito	2,1	42,2	35,6	20,1
15	Romania	1,3	42,9	46,4	9,4
16	Grecia	1,9	41,3	39,4	17,3
17	Bulgaria	2,2	43,8	39,2	14,8
18	Croazia	1,9	40,7	38,7	18,7
19	Lituania	1,9	42,5	38,3	17,3
20	Slovacchia	2,1	43,6	38,0	16,3
21	Repubblica Ceca	1,1	42,1	37,6	19,3
22	Polonia	2,4	42,9	37,5	17,2
23	Slovenia	1,6	41,8	37,4	19,2
24	Irlanda	1,9	42,3	37,0	18,7
25	Portogallo	1,8	44,6	36,9	16,6
26	Finlandia	1,2	44,1	36,4	18,3
27	Spagna	2,2	45,4	35,7	16,7
28	Germania	1,8	46,1	35,2	16,9

che, considerata la Romania con i suoi valori estremi come un caso a sé stante, conferma l'esistenza nei dati di sottoinsiemi numericamente ben caratterizzati, corrispondenti a quelli rilevati con la cluster analysis.

In conclusione nel caso di dati multivariati l'analisi delle componenti principali (**PCA**), pur con gli aspetti un po' delicati ricordati all'inizio:
→ consente di ridurre il numero delle variabili;
→ consente di semplificare la rappresentazione grafica dei dati;
→ comporta una perdita di informazione tuttavia controllata e misurabile;
→ fornisce le basi per realizzare il clustering.

Trovate il seguito e le strategie alternative di clustering e di analisi dei dati multivariati nei post:
→ Analisi dei gruppi (clustering gerarchico)
→ Analisi dei gruppi (clustering non gerarchico)
→ Analisi dei gruppi (clustering non esclusivo)

---

[1] Vedere Using R for Multivariate Analysis. URL consultato il 02/01/2023.

[2] Vedere Principal component analysis. URL consultato il 02/01/2023.

[3] Vedere Principal Component Analysis in R. URL consultato il 02/01/2023.

[4] I dati sono illustrati nel post Indice di massa corporea (BMI).

[5] Vedere il post Esprimere in numeri in formato fisso.

[6] Digitate **help(scale)** nella `Console di R` per la documentazione della funzione **scale()**.

lunedì 2 ottobre 2023

Istogrammi affiancati, sovrapposti e contrapposti

Abbiamo già visto [1] che per affiancare due istogrammi è sufficiente disporli in due colonne affiancate, ma qui lo facciamo con un piccolo trucco che può essere utile conoscere perché può essere impiegato in varie situazioni per migliorare l'estetica dei grafici, copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

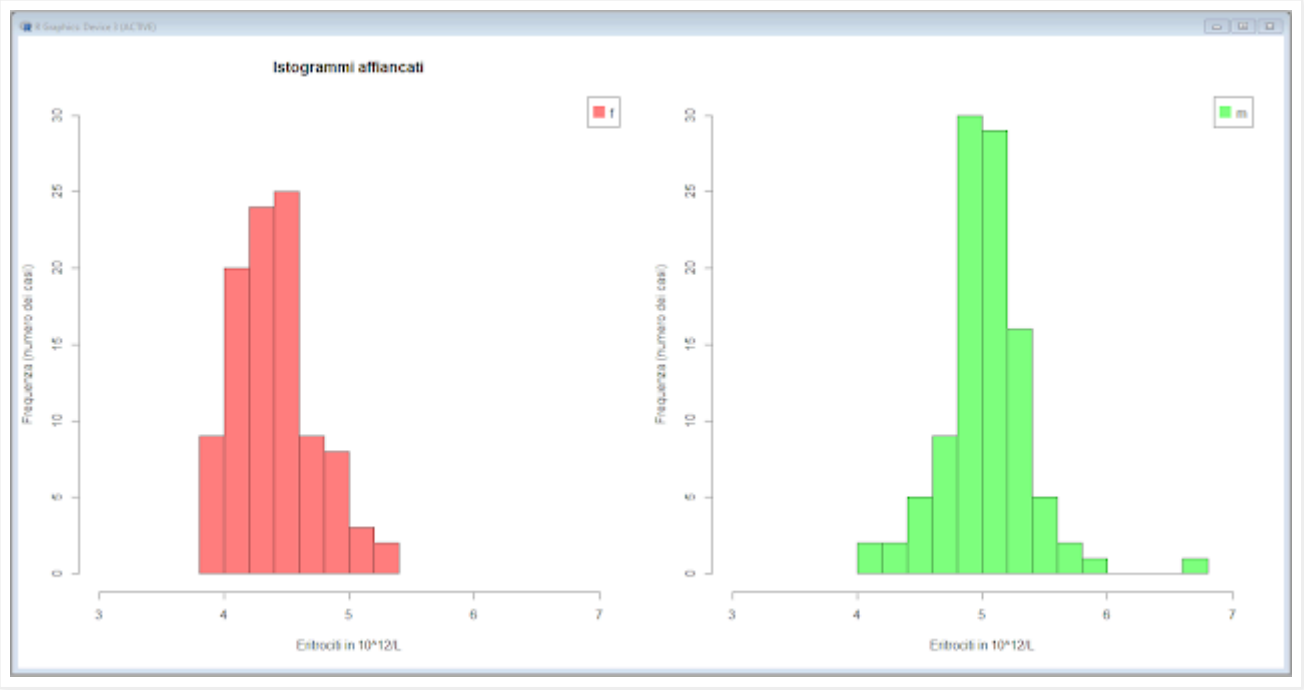
```
# DUE ISTOGRAMMI AFFIANCATI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
#
windows() # apre e inizializza una nuova finestra grafica
dev.new(unit="cm", width=30, height=15) # nuova dimensione della finestra grafica
par(mfrow=c(1,2)) # grafici organizzati in una riga e due colonne
#
hist(f, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), col=rgb(1,0,0,0.5), main="Istogrammi affiancati", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di f
legend("topright", legend=c("f"), col=c(rgb(1,0,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
hist(m, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(m)), col=rgb(0,1,0,0.5), main="", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di m
legend("topright", legend=c("m"), col=c(rgb(0,1,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
#
```

Dopo avere caricato il pacchetto **DAAG** che contiene il set di dati **ais** [2], a partire dai valori di concentrazione dei globuli rossi nel sangue che che sono stati determinati in 202 atleti, 100 donne e 102 uomini, contenuti nella variabile **rcc** del set di dati **ais**, riportiamo (**<-**) nell'oggetto **f** sotto forma di matrice (**as.matrix**) – con la funzione **subset()** che li estrae selezionandoli con **select=c(rcc)** – i valori di concentrazione dei globuli rossi dei soggetti di sesso femminile (**sex=="f"**).

Ripetiamo la cosa riportando nell'oggetto **m** i valori di concentrazione dei globuli rossi dei soggetti di sesso maschile (**sex=="m"**).

Dopo avere aperto una nuova finestra grafica con **windows()** ecco il trucco:
→ con **dev.new()** impostiamo nuove dimensioni per la finestra grafica;
→ impieghiamo come unità di misura i centimetri (**unit="cm"**);
→ portiamo la larghezza della finestra a 30 cm (**width=30**);
→ fissiamo l'altezza della finestra a 15 cm (**height=15**);
→ con **par(mfrow=c(1,2))** nella finestra grafica così ridimensionata organizziamo i grafici in una riga e due colonne.

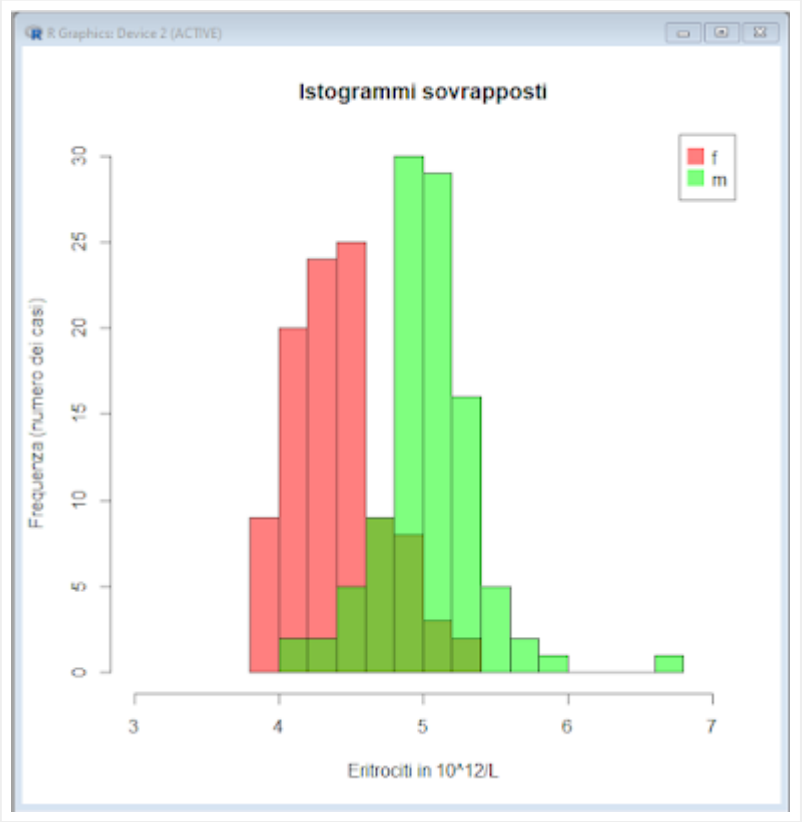
E questo è il risultato ottenuto tracciando i due istogrammi con la funzione **hist()** e aggiungendo loro la legenda con la funzione **legend()** [3].



Ora vediamo nuovamente come sovrapporre due istogrammi sfruttando la trasparenza del colore, copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# DUE ISTOGRAMMI SOVRAPPOSTI CON TRASPARENZA DEL COLORE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
#
windows() # apre e inizializza una nuova finestra grafica
#
hist(f, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), col=rgb(1,0,0,0.5),
main="Istogrammi sovrapposti", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di f
hist(m, add=TRUE, xlim=c(3,7), ylim=c(0,30),
breaks=sqrt(length(m)), col=rgb(0,1,0,0.5), ) # istogramma di m
legend("topright", legend=c("f","m"), col=c(rgb(1,0,0,0.5), rgb(0,1,0,0.5)), pt.cex=2,
pch=15 ) # riporta la legenda
#
```

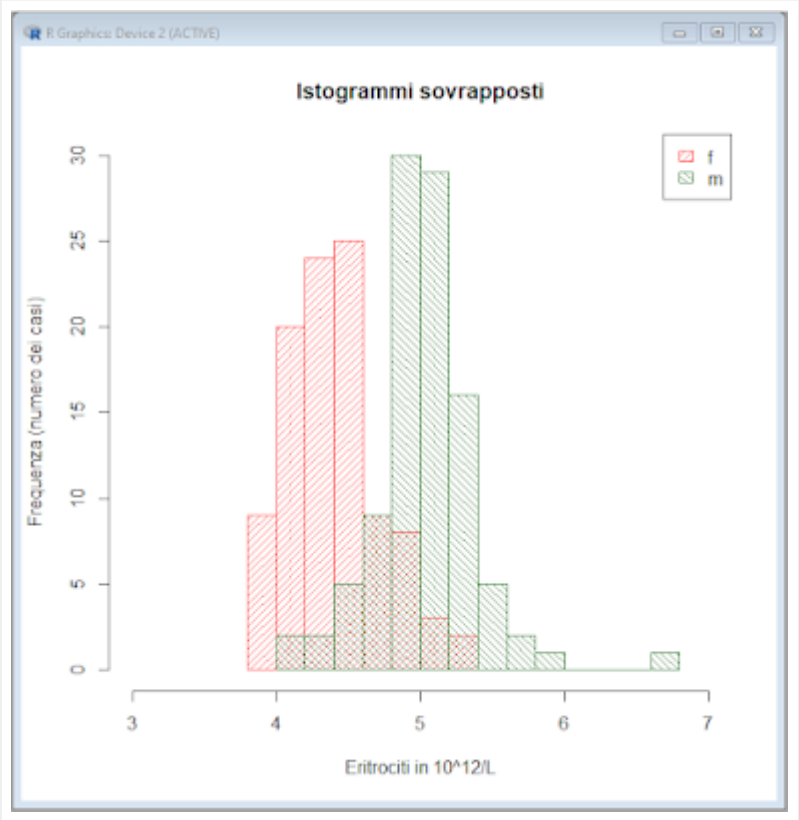
Qui le novità sono l'argomento **add=TRUE** che consente di sovrapporre il secondo istogramma al primo, e il quarto valore della funzione **rgb()** che viene posto uguale a **0.5** per assicurare la trasparenza al colore.



In alternativa possiamo sovrapporre due istogrammi impiegando il tratteggio del colore, copiate e incollate questo script nella `Console di R` e premete `↵ Invio`:

```
# DUE ISTOGRAMMI SOVRAPPOSTI CON TRATTEGGIO DEL COLORE
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
#
windows() # apre e inizializza una nuova finestra grafica
#
hist(f, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), border="red",
col="red", density=20, angle=45, main="Istogrammi sovrapposti", xlab="Eritrociti in 10^12/L", ylab="Frequenza (numero dei casi)") # istogramma di f
hist(m, add=TRUE, xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(m)), border="green4",
col="green4", density=20, angle=-45) # istogramma di m
legend("topright", legend=c("f","m"), fill=c("red","green4"), border=c("red","green4"),
col=c("red","green4"), density=c(20,20), angle=c(45,-45)) # riporta la legenda
#
```

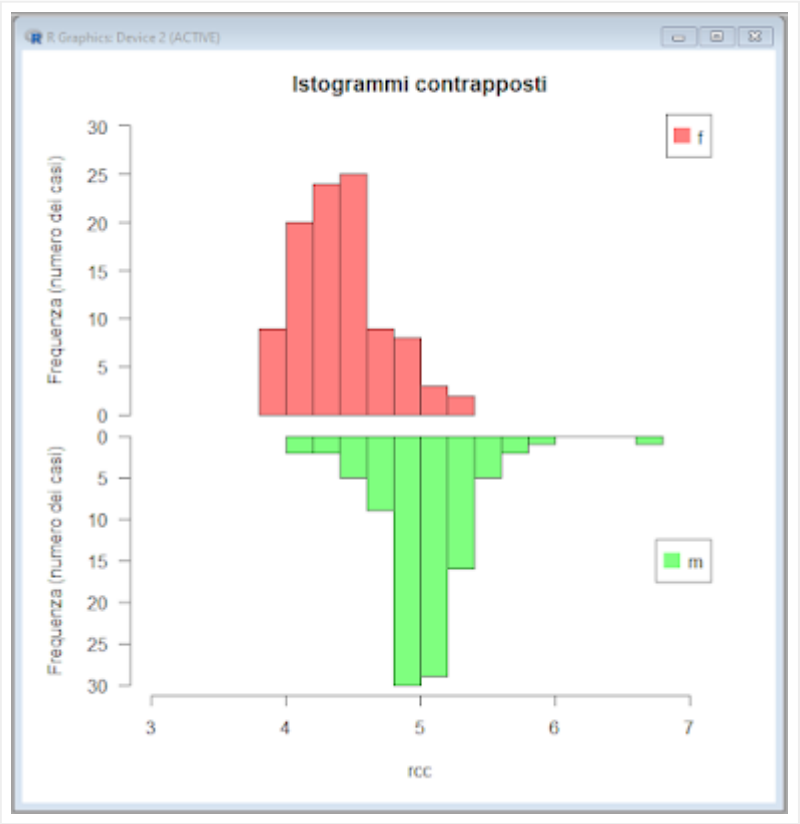
In questo, caso stabilito che il colore deve occupare il 20% della superficie da colorare con **density=20**, per distinguere i due istogrammi nell'area di sovrapposizione è sufficiente porre l'inclinazione di un tratteggio a 45 gradi (**angle=45**) in un istogramma a - 45 gradi (**angle=-45**) nell'altro (ma anche a qualche altra angolazione a piacere).



Possiamo anche contrapporre due istogrammi, copiate e incollate questo script nella `Console di R` e premete ↵ `Invio`:

```
# DUE ISTOGRAMMI CONTRAPPOSTI
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
f <-as.matrix(subset(ais, sex=="f", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=f
m <-as.matrix(subset(ais, sex=="m", select=c(rcc))) # estrae il sottoinsieme dei dati con sesso=m
#
windows() # apre e inizializza una nuova finestra grafica
par(mfrow=c(2,1)) # grafici organizzati in due righe e una colonna
#
par(mar=c(0,5,3,3)) # adatta i margini al grafico superiore
hist(f, xaxt="n", xlim=c(3,7), ylim=c(0,30), breaks=sqrt(length(f)), col=rgb(1,0,0,0.5), las=1, main="Istogrammi contrapposti", xlab="", ylab="Frequenza (numero dei casi)") # istogramma di f
legend("topright", legend=c("f"), col=c(rgb(1,0,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
par(mar=c(5,5,0,3)) # adatta i margini al grafico inferiore
hist(m, xlim=c(3,7), ylim=c(30,0), breaks=sqrt(length(m)), col=rgb(0,1,0,0.5), las=1, main="", xlab="rcc", ylab="Frequenza (numero dei casi)") # istogramma di m
legend("right", legend=c("m"), col=c(rgb(0,1,0,0.5)), pt.cex=2, pch=15) # riporta la legenda
par(mar=c(5.1,4.1,4.1,2.1)) # ripristina i margini di default
#
```

Otteniamo questo risultato



ed è più facile di quanto possa sembrare, perché a parte la necessità di organizzare i grafici in due righe e una colonna con **par(mfrow=c(2,1))** e non rappresentare l'asse delle ascisse del primo istogramma (**xaxt="n"**), il punto che sembra più complesso, cioè ribaltare la rappresentazione del secondo istogramma, lo si risolve semplicemente invertendo le coordinate dell'asse delle y con **ylim=c(30,0)**. Il resto è rappresentato dal semplice adattamento dei margini dei due istogrammi con **par(mar=....)** assegnando loro opportuni valori [4].

---

[1] Vedere il post Istogrammi.

[2] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza installare il pacchetto **DAAG**.

[3] Per gli argomenti impiegati vedere i post che trattano degli istogrammi e dell'aggiunta delle legende ai grafici, o digitare **help(*nomedellafunzione*)** nella `Console di R` per la documentazione ufficiale delle funzioni.

[4] Come personalizzare l'ampiezza dei margini lo trovate nella documentazione di **R** digitando **help(par)** nella `Console di R`.

sabato 25 novembre 2023

Come arrotondare i numeri

L'operazione di **arrotondamento** – che si rende necessaria a causa del fatto che i risultati di una elaborazione statistica sono generalmente ottenuti con un numero di cifre in eccesso rispetto a quelle significative – prevede quanto segue:
→ se il numero termina per 1, 2, 3 o 4 si arrotonda per difetto, così ad esempio 143.274 arrotondato a due decimali diventa 143.27 [1];
→ se il numero termina per 6, 7, 8 o 9 si arrotonda per eccesso, così ad esempio 87.136 arrotondato a due decimali diventa 87.14;
→ se il numero termina per 5 si arrotonda in modo che l'ultima cifra del numero dopo l'arrotondamento risulti pari, così ad esempio 8.25 arrotondato a un decimale diventa 8.2 e 12.75 diventa 12.8 (notare come in questo modo la media dei due numeri risulti immodificata e uguale a 10.5 sia prima sia dopo l'arrotondamento). La regola è riportata nella norma ISO/IEC 60559 (IEEE Std 754) come si trova documentato anche in **R** digitando **help(round)** che tra le altre informazioni sulla funzione restituisce:
"*... Si noti che per arrotondare un 5, dovrebbe essere utilizzato lo standard IEC 60559 (vedi anche "IEEE 754"), "vai alla cifra pari". Pertanto round(0.5) è 0 e round(-1.5) è -2. Tuttavia, ciò dipende dai servizi del sistema operativo e dall'errore di rappresentazione (poiché ad esempio 0.15 non è rappresentato esattamente, la regola di arrotondamento si applica al numero rappresentato e non al numero stampato, quindi round(0.15, 1) potrebbe essere 0.1 oppure 0.2)*" [2].

Copiate lo script quindi incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ARROTONDAMENTO a un dato numero di cifre decimali
#
mydata <- cbind(c(1.8246e-02, 6.91635e-01, 3.2385e-02), c(8.37544e-01, 5.2175e-02, 2.4565e-02)) # tabella con i dati di esempio
#
myroun <- round(mydata, digits=5) # arrotonda i dati a 5 cifre decimali
#
mydata # mostra i dati originali
myroun # mostra i dati arrotondati a 5 cifre decimali
#
```

```
> mydata # mostra i dati originali
          [,1]     [,2]
[1,] 0.018246 0.837544
[2,] 0.691635 0.052175
[3,] 0.032385 0.024565
> myroun # mostra i dati arrotondati a 5 cifre decimali
        [,1]    [,2]
[1,] 0.01825 0.83754
[2,] 0.69164 0.05218
[3,] 0.03238 0.02456
```

Come si vede è possibile arrotondare a un dato numero di cifre decimali impiegando la funzione **round()** con l'argomento **digits=n** ove **n** specifica il numero di cifre decimali. Da notare che qui sopra nei numeri che terminano con 5 dopo l'arrotondamento l'ultima cifra è sempre pari.

Se volete invece arrotondare a un dato numero di cifre significative dovete impiegare la funzione **signif()** con l'argomento **digits=n** ove **n** specifica il numero di cifre significative.

Copiate questo script quindi incollatelo nella `Console di R` e premete `↵ Invio`:

```
# ARROTONDAMENTO a un dato numero di cifre significative
#
mydata <- cbind(c(1.8246e-02, 6.91635e-01, 3.2385e-02), c(8.37544e-01, 5.2175e-02,
2.4565e-02)) # tabella con i dati di esempio
#
mysign <- signif(mydata, digits=3) # arrotonda i dati a 3 cifre significative
#
mydata # mostra i dati originali
mysign # mostra i dati arrotondati a 3 cifre significative
#
```

```
> mydata # mostra i dati originali
          [,1]     [,2]
[1,] 0.018246 0.837544
[2,] 0.691635 0.052175
[3,] 0.032385 0.024565
> mysign # mostra i dati arrotondati a 3 cifre significative
       [,1]   [,2]
[1,] 0.0182 0.8380
[2,] 0.6920 0.0522
[3,] 0.0324 0.0246
```

Il razionale per decidere quale deve essere il numero di cifre significative da impiegare nella rappresentazione di un numero è riportato nel post Come stabilire il giusto numero di cifre significative.

Infine, se i numeri sono espressi in formato esponenziale, potete anche aggiungere all'arrotondamento dei numeri la loro espressione in formato fisso impiegando la funzione **options()** con l'argomento **scipen**, seguendo le indicazioni del post Come riportare i numeri in formato fisso.

---

[1] Nota bene: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata negli script, nei file di dati e in tutto il testo di questo sito.

[2] Round {base} R Documentation. Rounding of Numbers. "*Note that for rounding off a 5, the IEC 60559 standard (see also 'IEEE 754') is expected to be used, 'go to the even digit'. Therefore round(0.5) is 0 and round(-1.5) is -2. However, this is dependent on OS services and on representation error (since e.g. 0.15 is not represented exactly, the rounding rule applies to the represented number and not to the printed number, and so round(0.15, 1) could be either 0.1 or 0.2)*".

sabato 23 dicembre 2023

AuguRi di Buone Feste!

```
# X-Mas Tree with 10 Lines of R Code
# Credits: Michael Mayer
# Posted on December 20, 2021 by Michael Mayer in "R bloggers"
# https://bit.ly/3H2gWP9
#
library(rgl)
t <- seq(0, 100, by=0.7)^0.6
x <- t * c(sin(t), sin(t + pi))
y <- t * c(cos(t), cos(t + pi))
z <- -2 * c(t, t)
color <- rep(c("darkgreen", "gold"), each=length(t))
open3d(windowRect=c(100, 100, 600, 600), zoom=0.9)
bg3d("red")
spheres3d(x, y, z, radius=0.3, color=color)
#
# On screen (skip if Export)
play3d(spin3d(axis=c(0,0,1), rpm=4), duration=30)
#
# Export (requires 3rd party tool "ImageMagick" resp. magick-package)
# movie3d(spin3d(axis=c(0,0,1), rpm=4), duration=30, dir=getwd())
#
```

Per riavviare l'animazione chiudere e riaprire la pagina web.

domenica 7 gennaio 2024

Scale nominali, scale ordinali e scale numeriche

Le relazioni che intercorrono tra il tipo di *variabile*, il *processo* mediante il quale viene ottenuto il dato, la *modalità di espressione* del dato e il tipo di *scala* nella quale il dato viene espresso forniscono la **definizione operativa** delle scale di misura.

Questa definizione deriva dalla loro **definizione metrologica** prevista nel *Sistema internazionale di unità* (**SI**), che l'Italia nel 1982 ha adottato per legge, aggiornandolo successivamente con le modifiche via via apportate [1].

Definizione operativa				Definizione metrologica (VIM)
Tipo di variabile	**Dato ottenuto mediante un processo di...**	**Modalità di espressione del dato**	**Scala nella quale il dato viene espresso**	
Qualitativa	Classificazione qualitativa	Descrizione verbale (maschio / femmina)	Scala nominale	Scala di riferimento convenzionale
	Classificazione semiquantitativa	Attributo ordinabile (neonato < bambino < adulto)	Scala ordinale	Scala ordinale
Quantitativa	Conteggio	Numero naturale | 1 | 2 | 3 | ...	Scala numerica discreta	Scala di valori o Scala di misura
	Misura	Numero razionale |1.74|152.9|0.32| ...	Scala numerica continua	

Le generalità del **SI** sono riportate nella brochure pubblicata dal *Bureau international des poids et mesures* (BIPM) [2] mentre il glossario del processo di misura è riportato nel **VIM**, il *Vocabolario Internazionale di Metrologia* [3], nel quale le scale di misura sono così definite:
→ scala di riferimento convenzionale – *scala di valori definita mediante un accordo ufficiale* [VIM 1.29]
→ scala ordinale – *scala di valori per una grandezza ordinale* [VIM 1.28]
→ scala di valori o scala di misura – *insieme ordinato dei valori di grandezze di una data natura, utilizzato per classificare le grandezze di quella natura in ordine crescente o decrescente in base alle loro espressioni quantitative* [VIM 1.27]

Quindi la *definizione operativa* delle scale di misura, a parte semplificare, ma senza cambiarne il senso, la definizione della "*scala di riferimento convenzionale*" riportandola da un "*accordo ufficiale*" al caso delle meno burocratiche, universalmente impiegate e più immediatamente intuitive **scale nominali**, riprende tal quale la definizione delle **scale ordinali** e semplicemente suddivide le **scale numeriche**, lasciandone immodificata la *definizione metrologica* riportata dal VIM, in **discrete** e in **continue** in base al processo mediante il quale è ottenuto il dato: conteggio o misura [4]. Questa suddivisione trova ulteriore supporto nel fatto che il *processo di conteggio* fornisce risultati espressi sotto forma di numeri interi e di una sola unità di misura ("*uno*") basata sulla grandezza "*numero di entità*" che non ha dimensioni [VIM 1.4 e 1.8], mentre il *processo di misura* fornisce risultati espressi sotto forma di numeri razionali e di differenti unità di misura (metro, secondo, kilogrammo, eccetera) basate su grandezze (lunghezza, tempo, massa e quant'altro) che hanno una dimensione fisica [VIM 1.7].

Ma a questo punto la domanda è inevitabile: se esistono definizioni ufficiali, ampiamente condivise e addirittura normate, dalle quali è possibile trarre una definizione operativa piuttosto chiara che le collega ai processi di misura impiegati nella pratica, perché si trova riportato quasi dappertutto che le scale di misura sono classificate in: scale nominali, scale ordinali, scale a intervalli, scale a rapporti ?

La risposta è: per qualche ragione imperscrutabile, probabilmente potenziata dalla disponibilità del copia-e-incolla, un meme è riuscito a moltiplicarsi in modo virale fino ad occupare quasi completamente l'infosfera concernente questo argomento. E senza ragione, senza un razionale.

La proposta di suddivisione in scale nominali, scale ordinali, scale a intervalli, scale a rapporti è riportata in un lavoro del 1946 di S. S. Stevens [5], che proprio all'inizio del lavoro scrive: "*PER SETTE ANNI UN COMITATO dell'Associazione britannica per l'avanzamento della scienza ha dibattuto il problema della misurazione. Nominato nel 1932 a rappresentare la Sezione A (Matematica e scienze fisiche) e Sezione J (Psicologia), il comitato è stato incaricato di considerare e riferire sulla possibilità di "stime quantitative di eventi sensoriali", che significano semplicemente: è possibile misurare le sensazioni umane? La riflessione ha portato solo al disaccordo, soprattutto su cosa si intende con il termine misurazione*" [6].

La domanda che Stevens si pone "*è possibile misurare le sensazioni umane?*" sarà pure lecita, ma non si possono avere dubbi "*su cosa si intende con il termine misurazione*".

Una misurazione o procedimento di *misura* consiste nell'esprimere una grandezza in modo quantitativo, mediante un "*valore numerico*" che è un numero puro uguale al rapporto tra il "*(valore della) grandezza (in esame)*" e il valore di una grandezza di riferimento ad essa omogenea definita come "*unità di misura*", ed è quindi

(valore della) grandezza (in esame) / unità di misura = valore numerico

Corollario: il risultato di una misura ovvero il "*(valore della) grandezza (in esame)*" è dato dal prodotto di un "*valore numerico*" per la "*unità di misura*"

(valore della) grandezza (in esame) = valore numerico · unità di misura

Per indicare il prodotto, rendendo "*valore numerico*" e "*unità di misura*" un tutt'uno inscindibile, il simbolo del prodotto [·] per convenzione viene sostituito con lo spazio unificatore [7], come avviene ad esempio in 2.5 m, 12 s, 75 kg (unica eccezione sono le unità angolari, per le quali lo spazio tra il valore numerico e il simbolo dell'unità non è previsto, pertanto un angolo verrà riportato, ad esempio, come 41°54'39).

Il lavoro di Stevens è viziato da un errore concettuale di fondo: nel tentativo di "*misurare le sensazioni umane*" e dare dignità di "*misura*" ai risultati dei metodi di ricerca in psicologia, ha cercato di adattare alla psicologia "... *il termine misurazione...*" – che ha e deve avere una definizione univoca – pensando di risolvere il problema con nuove definizioni delle scale di misura, mentre avrebbe dovuto ricercare la "... *possibilità di "stime quantitative di eventi sensoriali"...*" in una qualche grandezza [VIM 1.1] e nella sua unità di misura [VIM 1.9], perché senza il fondamento di grandezze [misurabili] e delle loro unità di misura, le scale di misura [VIM 1.27] sono una espressione vuota.

Quindi mentre per *misura* [VIM 2.1] e per tutti gli altri termini che devono essere impiegati per esprimersi correttamente in campo scientifico (ma anche nella vita di tutti i giorni) rimando al **SI** e alle definizioni del **VIM**, qui mi limito a rilevare che la forzatura impiegata da Stevens per accreditare le misure in campo psicologico ha determinato solamente confusione terminologica, anticamera della confusione concettuale [8].

Se proprio si vuole ricollegare la **definizione di Stevens** delle scale di misura – inutile dire che è fortemente sconsigliata anche in campo psicologico, al quale nulla aggiunge rispetto alle scale ufficiali qui trattate: in ogni caso il VIM semplicemente la ignora – alla **definizione operativa** delle scale di misura in campo scientifico, e con questa alla loro **definizione metrologica**, basta notare che:
- se i dati sono espressi in una scala numerica discreta possiamo avere:

[A] quella che Stevens denomina *scala a intervalli* quando, come accade ad esempio per le date, non sono possibili rapporti (non posso dividere il 28 febbraio per il 2 febbraio) ma posso solamente calcolare l'intervallo che le separa (nel caso specifico 26 giorni);
[B] quella che Stevens denomina *scala a rapporti* quando, come accade ad esempio per il numero di auto ferme al semaforo, se prima ne ho contate 5 e dopo ne ho contate 4, oltre a calcolare la differenza posso calcolare anche il rapporto prima-dopo che è 1.25;
- se i dati sono espressi in una scala numerica continua possiamo avere:
[A] quella che Stevens denomina *scala a intervalli* quando, come accade ad esempio per la temperatura in gradi Celsius (°C), posso dire che tra la temperatura di +20 °C e la temperatura di -10 °C c'è un intervallo di 30 °C, ma non posso dividere la prima per la seconda;
[B] quella che Stevens denomina *scala a rapporti* quando, come accade ad esempio per la concentrazione del colesterolo nel siero, posso calcolare non solo l'intervallo tra due valori (la mia concentrazione di colesterolo è 220 mg/dL, la tua 110 mg dL, la differenza è di 110 mg/dL) ma anche il rapporto (la mia concentrazione del colesterolo è 2.0 volte la tua).

Questo significa che (tolte le scale nominali e le scale ordinali sulle quali tutti concordano) quelle di Stevens non sono scale di misura ma, nella migliore e proprio nella migliore delle ipotesi, potrebbero (e uso il condizionale) essere considerate un attributo aggiuntivo (ma privo di interesse metrologico) delle scale numeriche. Riprendiamo quindi la nostra **definizione operativa** delle scale di misura, che trova un fondamento nel SI e nel VIM e che fornisce il razionale su cui si fondano le modalità di rappresentazione grafica e di analisi statistica dei dati.

Il tipo più semplice di scala è rappresentato dalla **scala nominale**. Corrisponde ai dati della natura più elementare, quella di dati qualitativi per i quali è disponibile solamente una descrizione verbale. Ne sono un esempio la *classificazione* maschio / femmina, la classificazione dei gruppi sanguigni ABO (O, A, B, AB), dei motori (endotermici, esotermici, elettrici, eccetera), delle specie animali (ad esempio: cane domestico, sciacallo, coyote, lupo, dingo) e così via. La misura più intuitiva e più utilizzata, nel caso della scala nominale, è costituita dalla percentuale (o dalla proporzione o frazione). In questo caso le rappresentazioni che si possono impiegare sono il *grafico a torta* (pie-chart) e, quando le grandezze da rappresentare sono indipendenti e non costituiscono la parte di un tutto, il *grafico a barre* (barplot) con le barre staccate e disposte orizzontalmente.

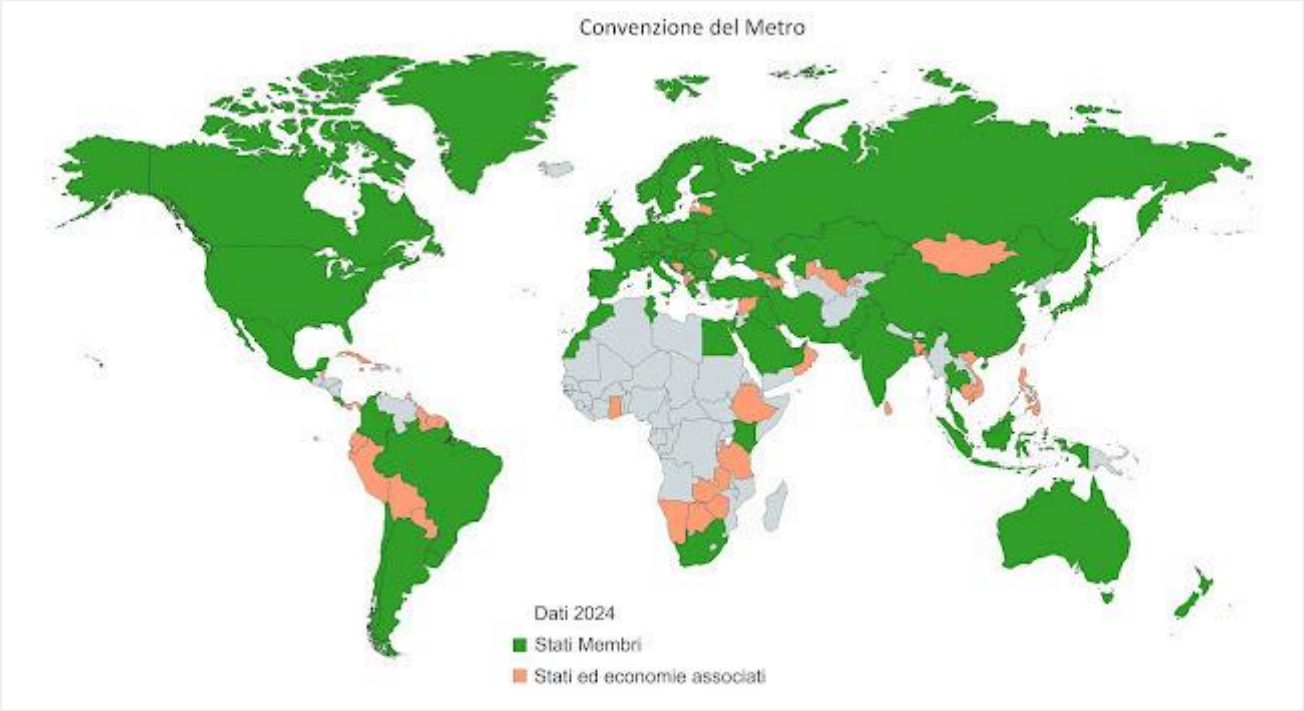
Nel caso di dati qualitativi per i quali è disponibile una descrizione sotto forma di attributo ordinabile si ha a che fare con una **scala ordinale**. Questo avviene per esempio per la *classificazione* in neonati, bambini e adulti, nella quale è appropriato applicare un ordinamento crescente per classi di età (neonato < bambino < adulto), per il livello di scolarità (scuola elementare < scuola media < diploma < laurea), eccetera. Un altro esempio è la classificazione per ranghi, nella quale il valore osservato viene trasformato nel corrispondente rango, cioè nel numero della posizione che il dato occupa nella lista ordinata dei dati. Così la classificazione inadeguato / parzialmente adeguato / adeguato / più che adeguato / eccellente può essere trasformata nella classificazione 1 / 2 / 3 / 4 / 5. Per le scale ordinali è possibile utilizzare un *grafico a torta* (pie-chart) oppure, per valorizzare la possibilità di ordinamento delle classi, un *grafico a barre* (barplot) con le barre staccate per indicare la discontinuità.

Una **scala numerica discreta** è tipicamente quella relativa a dati numerici ottenuti mediante operazioni di *conteggio* e riportati sotto forma di interi o *numeri naturali* (1, 2, 3, ….). Si prenda ad esempio il conteggio degli eritrociti (globuli rossi) nel sangue. La differenza minima tra due conteggi teoricamente misurabile è rappresentata da un globulo rosso: i dati sono numerici, ma tra l'uno l'altro non esistono valori intermedi. Altri esempi di scala numerica discreta sono il numero di controlli medici effettuati da una donna durante la gravidanza, il numero di insegnanti di ruolo nel liceo vicino a casa, il numero di auto ferme al semaforo e così via. Per le scale numeriche discrete si possono utilizzare un *grafico a barre* (barplot) con le barre staccate per indicare la discontinuità ma non l'istogramma che è basato sull'ipotesi di continuità dei dati, o anche, nel caso in cui si debbano rappresentare contemporaneamente due variabili, un *grafico di dispersione* (grafico xy o grafico cartesiano o scatterplot) precisando che i valori compresi tra due interi non hanno un corrispettivo reale.

La **scala numerica continua** è tipicamente quella impiegata per rappresentare dati numerici ottenuti mediante procedimenti di *misura* come quelli chimici e quelli fisici e i cui dati sono riportati sotto forma di *numeri razionali* (1.435, 123.9, 84.327, ...). Esempi ne sono la misura della concentrazione del colesterolo nel sangue, la misura della distanza tra due luoghi, la misura della velocità di un'auto, la misura del peso di una persona e quant'altro. L'intervallo tra due valori può essere suddiviso a piacere, anche se non oltre i limiti del potere di risoluzione degli strumenti di misura, che impongono nella pratica un numero molto limitato di cifre significative. Per le scale numeriche continue è possibile impiegare di volta in volta il classico *istogramma*, un grafico a linee spezzate come un *poligono di frequenza*, o ancora un *grafico di dispersione* (grafico xy o grafico cartesiano o scatterplot) nel caso in cui si debbano rappresentare contemporaneamente due variabili.

Dal punto di vista della statistica, ai dati espressi in forma di **scale nominali** sono applicabili i test basati sulla enumerazione dei casi (come test chi-quadrato, test di Fisher, test di McNemar), ai dati espressi in forma di **scale ordinali** sono applicabili in aggiunta i test basati sui ranghi, mentre ai dati espressi in forma di **scale numeriche** sono applicabili, con la necessaria appropriatezza e le dovute attenzioni, i test basati sulle ipotesi di continuità e gaussianità delle distribuzioni (test parametrici), i test non parametrici e i test basati su modelli bayesiani.

La conclusione? Molto semplice. Siglata per la prima volta a Parigi il 20 maggio 1875 da 17 Stati [9] la **Convenzione del metro**, che originariamente prevedeva l'impiego del *Sistema metrico decimale* che si è poi evoluto nel *Sistema internazionale di unità*, secondo i dati ufficiali forniti dal BIPM all'inizio del 2024 contava 64 Stati Membri [10] e 36 Stati ed economie associati [11].



Considerato che tutte le nazioni più popolose, incluse Cina e India, vi aderiscono, il **SI** con le sue grandezze e unità di misura è diventato un comune denominatore e un riferimento per miliardi di persone, nella scienza ma anche nella vita quotidiana [12].

Tra i documenti che il **SI** mette a disposizione perché ci si possa avvicinare in modo adeguato al mondo delle misure, il *Vocabolario Internazionale di Metrologia* riveste un ruolo cruciale: e questo post vuole essere un invito a consultarlo per formare, per informare, per adeguare i nostri modi di riportare i dati ed esprimerci con la necessaria appropriatezza quando parliamo di misure.

Nota bene: in **R** il separatore delle cifre decimali è il punto (.) e come già riportato altrove questa convenzione per ragioni di omogeneità viene adottata negli script, nei file di dati e nel testo dei post.

---

[1] Il *Sistema Internazionale di unità (SI)* diviene legale in Italia nel 1982, le successive modifiche sono recepite nel 2001, nel 2009 e nel 2020, questa è la normativa aggiornata a inizio 2024, che include la correzione riportata a livello comunitario nel 1984:
- DECRETO DEL PRESIDENTE DELLA REPUBBLICA 12 agosto 1982, n. 802. *Attuazione della direttiva (CEE) n. 80/181 relativa alle unita' di misura*. GU Serie Generale n.302 del 03-11-1982 - Suppl. Ordinario.
https://www.gazzettaufficiale.it/eli/id/1982/11/03/082U0802/sg
- DIRETTIVA DEL CONSIGLIO del 18 dicembre 1984 che modifica la direttiva 80/181/CEE per il ravvicinamento delle legislazioni degli Stati membri relative alle unità di misura.
https://eur-lex.europa.eu/legal-content/IT/TXT/PDF/?uri=CELEX:31985L0001
- DECRETO 29 gennaio 2001. *Attuazione della direttiva 1999/103/CE che modifica la direttiva 80/181/CEE sul riavvicinamento delle legislazioni degli Stati membri relative alle unita' di misura*. GU Serie Generale n.27 del 02-02-2001
https://www.gazzettaufficiale.it/eli/id/2001/02/02/001A1152/sg
- DECRETO 29 ottobre 2009. *Attuazione della direttiva 2009/3/CE del Parlamento europeo e del Consiglio dell'11 marzo 2009 che modifica la direttiva 80/181/CEE del Consiglio sul riavvicinamento delle legislazioni degli Stati membri riguardo alle unita' di misura*. (09A13580). GU Serie Generale n.273 del 23-11-2009
https://www.gazzettaufficiale.it/eli/id/2009/11/23/09A13580/sg
- DECRETO 7 aprile 2020. *Attuazione della direttiva (UE) 2019/1258 della Commissione del 23 luglio 2019 che modifica, ai fini dell'adattamento al progresso tecnico, l'allegato della direttiva 80/181/CEE del Consiglio per quanto riguarda le definizioni delle unita' SI di base*.
https://www.gazzettaufficiale.it/eli/id/2020/05/09/20A02529/sg

[2] Tutti i documenti del **SI** sono in versione bilingue, in francese e in inglese.
Brochure sur le SI: Le Système international d'unités.
https://www.bipm.org/fr/publications/si-brochure
SI Brochure: The International System of Units (SI).
https://www.bipm.org/en/publications/si-brochure

[3] *VIM: International Vocabulary of Metrology*. Alla pagina: *CGM Publications: Guides in Metrology*.
https://www.bipm.org/en/committees/jc/jcgm/publications

[4] Nel caso delle scale numeriche la *definizione metrologica* parla solo di *scala di misura* e non distingue tra scala numerica *continua* e scala numerica *discreta* in quanto quest'ultima viene considerata come caso limite nel quale l'unità di misura è "*uno*".

[5] S. S. Stevens. *On the Theory of Scales of Measurement*. Science, New Series, Vol. 103, No. 2684 (Jun. 7, 1946), pp. 677-680. American Association for the Advancement of Science.
http://www.jstor.org/stable/1671815

[6] "*FOR SEVEN YEARS A COMMITTEE of the British Association for the Advancement of Science debated the problem of measurement. Appointed in 1932 to represent Section A (Mathematical and Physical Sciences) and Section J (Psychology), the committee was instructed to consider and report upon the possibility of "quantitative estimates of sensory events"-meaning simply: Is it possible to measure human sensation? Deliberation led only to disagreement, mainly about what is meant by the term measurement*".

[7] Lo spazio unificatore non cambia ampiezza e non consente di andare a capo, lasciando quindi l'espressione sempre correttamente impaginata. In ambiente Windows può essere inserito tenendo premuti contemporaneamente i tasti <**ctrl**> e <**shift**> e premendo la **barra spaziatrice**, oppure

tenendo premuto il tasto <**alt**> e digitando **255** sul tastierino numerico, su Mac tenendo premuto il tasto <**alt**> e premendo la **barra spaziatrice**.

[8] Non è questa la sede per trattare un tema così complesso e delicato, ma un esempio di come la confusione terminologica possa sfociare in confusione concettuale lo si trova in: Paul C. Price, Rajiv S. Jhangiani, I-Chant A. Chiang, Dana C. Leighton, and Carrie Cuttler. *Research Methods in Psychology*. 3rd American Edition
https://opentext.wsu.edu/carriecuttler/
laddove gli autori si chiedono al punto 4.1 "*Cosa è la misura?*" rispondendo che in psicologia "*La misura è l'assegnazioni di punteggi agli individui...*" – secondo loro analogamente a come in fisica il risultato di una procedura per il calcolo dell'energia potenziale di un oggetto "... *è un punteggio che rappresenta l'energia potenziale dell'oggetto...*". Ma si tratta di una affermazione errata e fuorviante in quanto ignora il fatto che per "*rappresentare*" l'energia potenziale è necessario che il "*punteggio*" (ma forse sarebbe meglio chiamarlo "*valore numerico*" [VIM 1.20]) sia associato indissolubilmente alle sue dimensioni fisiche [VIM 1.7] attraverso una grandezza [VIM 1.4 e 1.5] e le relative unità di misura [VIM 1.9]. Mentre quella che un loro punteggio (score) in psicologia "*misura*" – se sono loro stessi ad affermare che "*... il punto importante qui è che la misurazione non richiede strumenti o procedure particolari. Non è necessario posizionare individui o oggetti sulla bilancia, porre davanti a loro dei regoli graduati o inserire termometri al loro interno. Ciò che richiede è una procedura sistematica per assegnare punteggi a individui o oggetti in modo che tali punteggi rappresentino la caratteristica di interesse*" – o è semplicemente una *valutazione* da esprimere in una scala nominale o al massimo può essere una *quantità* esprimibile in una scala ordinale (che il VIM al punto 1.26 definisce come una "*quantità definita mediante un procedimento di misurazione convenzionale, per la quale può essere stabilita una relazione di ordinamento totale, secondo la grandezza, con altre quantità dello stesso tipo, ma per la quale non esistono operazioni algebriche tra tali quantità*").

[9] Argentina, Austria-Ungheria, Belgio, Brasile, Danimarca, Francia, Germania, Italia, Perù, Portogallo, Russia, Spagna, Svezia e Norvegia, Svizzera, Turchia, Stati Uniti, Venezuela.
https://www.bipm.org/en/metre-convention

[10] Arabia Saudita, Argentina, Australia, Austria, Belgio, Bielorussia, Brasile, Bulgaria, Canada, Cile, Cina, Colombia, Costa Rica, Corea (Repubblica), Croazia, Danimarca, Ecuador, Egitto, Emirati Arabi Uniti, Estonia, Finlandia, Francia, Germania, Giappone, Grecia, India, Indonesia, Iran (Repubblica Islamica), Iraq, Irlanda, Israele, Italia, Kazakistan, Kenya, Lituania, Malaysia, Marocco, Messico, Montenegro, Norvegia, Nuova Zelanda, Olanda, Pakistan, Polonia, Portogallo, Regno Unito, Repubblica Ceca, Romania, Russia (Federazione), Serbia, Singapore, Slovacchia, Slovenia, Sudafrica, Spagna, Stati Uniti, Svezia, Svizzera, Thailandia, Tunisia, Turchia, Ucraina, Ungheria, Uruguay.
https://www.bipm.org/en/member-states

[11] Albania, Azerbaijan, Bangladesh, Bolivia, Bosnia-Erzegovina, Botswana, Cambogia, CARICOM (la Comunità caraibica), Etiopia, Georgia, Ghana, Hong Kong (Cina), Giamaica, Kuwait, Latvia, Lussemburgo, Malta, Mauritius, Moldavia (Repubblica), Mongolia, Namibia, Macedonia del Nord, Oman, Panama, Paraguay, Perù, Filippine, Qatar, Siria, Sri Lanka, Taiwan, Tanzania, Uzbekistan, Vietnam, Zambia, Zimbabwe.
https://www.bipm.org/en/associates

[12] Residui di storiche unità non-**SI** (miglia, acri, galloni, libbre, gradi Fahrenheit) permangono ancora nelle *United States Customary Units* e nel *Sistema Imperiale Britannico*, il cui uso è ammesso rispettivamente da USA e Regno Unito accanto al **SI** ufficialmente adottato.

domenica 21 gennaio 2024

Grafici 3D con i dati etichettati

Nello studio della relazione di funzione che lega tra loro tre variabili può essere utile la loro rappresentazione in un grafico tridimensionale (grafico 3D). Qui vediamo come sia possibile riportare i dati associando a ciascuno di essi una etichetta che ne consenta la immediata identificazione all'interno del grafico.

Come dati impieghiamo i valori di BMI (indice di massa corporea) rilevati a livello europeo alcuni anni fa e pubblicati dall'Istat [1].

Per proseguire è necessario:
→ effettuare il download del file di dati `bmi.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un `↵ Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

```
Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8
Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7
Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2
```

Inoltre dovete scaricare dal **CRAN** il pacchetto aggiuntivo **rgl** [2]. Copiate e incollate nella `Console di R` questo script, se necessario iconizzate la `Console di R` per visualizzare il grafico:

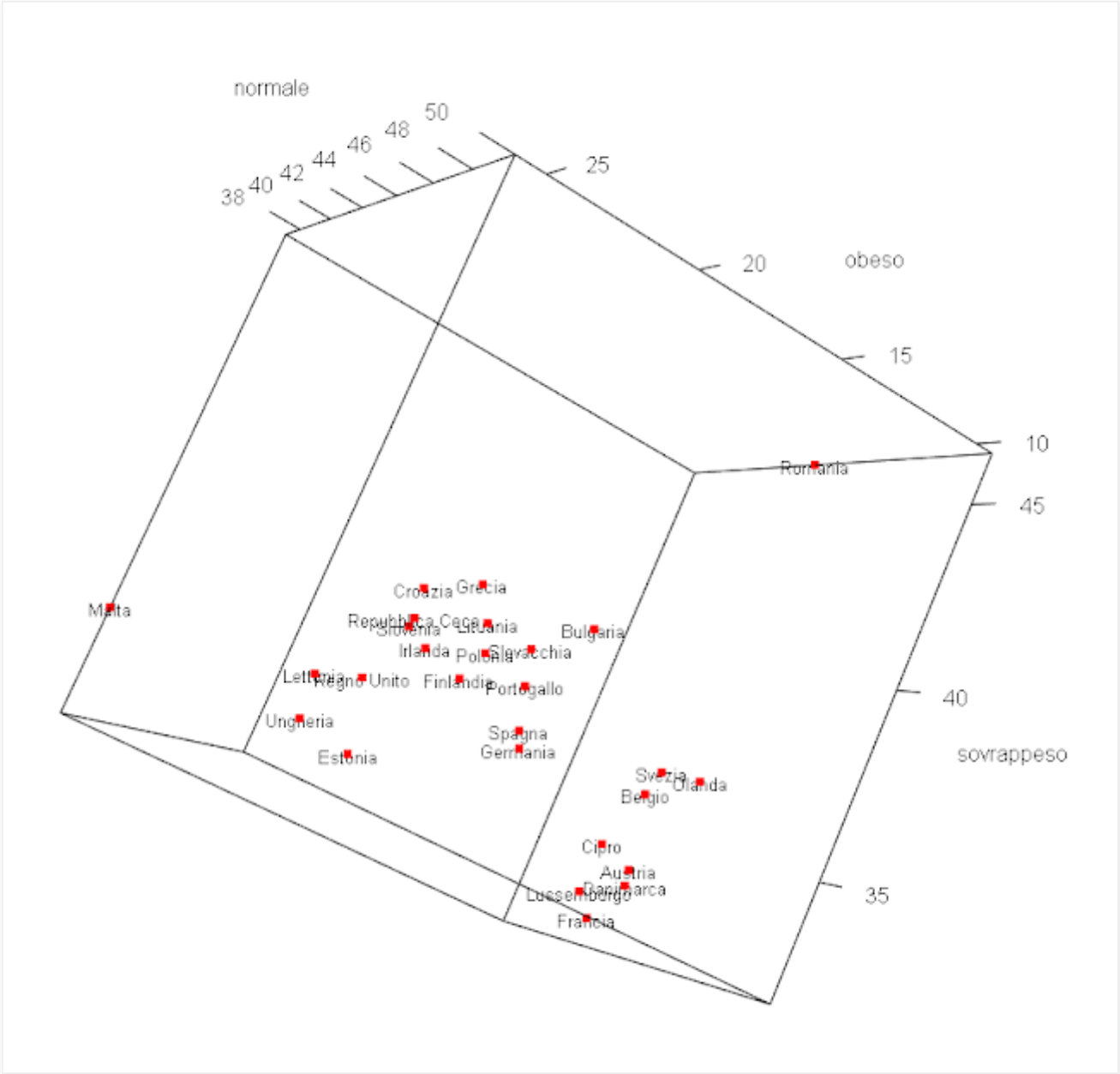
```
# GRAFICO 3D INTERATTIVO CHE PUO' ESSERE RUOTATO CON IL MOUSE
# accanto a ciascun punto viene riportata l'etichetta che lo identifica
#
library(rgl) # carica il pacchetto per la rappresentazione grafica 3D
data <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";") # importa i dati
#
with(data, plot3d(normale, sovrappeso, obeso, type="p", col="red", size=6)) # genera il grafico 3d
with(data, text3d(normale, sovrappeso, obeso, Nazione, cex=0.8)) # identifica ciascun punto con una etichetta
#
```

Nelle due prime righe con **library()** viene caricato il pacchetto **rgl** e con **read.table()** i dati sono importati nell'oggetto **data**.

Quindi viene impiegata due volte la funzione **with()** che specifica i dati da impiegare (**data**) e la funzione da applicare. La prima volta con la funzione **plot3d()** viene realizzato il grafico 3D, la seconda volta con la funzione **text3d()** accanto a ciascun dato viene riportata la scritta/etichetta contenuta nella variabile `Nazione`.

Se avete dubbi sui valori che possono assumere gli argomenti **type**, **col**, **size** e **cex** potete come sempre trovare aiuto digitando **help**(*nomedellafunzione*) nella `Console di R`.

Il grafico 3D è interattivo. Infatti se "afferrate" il grafico risultante facendo `click` su di esso e tenendo premuto il `tasto sinistro del mouse` senza rilasciarlo, potete ruotare il grafico a vostro piacimento muovendo il mouse. In questo modo è stata realizzata – dopo avere ingrandito la finestra grafica a tutto schermo – l'immagine seguente che fornisce una delle tante possibili viste [della distribuzione] dei dati.



Se infine volete una animazione che presenti il grafico sotto le varie possibili angolazioni copiate e incollate nella `Console di R` quest'altra riga di codice, se necessario iconizzate la `Console di R` per visualizzare il grafico:

```
#
with(data, play3d(spin3d(axis=c(0,0,1), rpm=3), duration=60)) # fa ruotare il grafico per 60 secondi
#
```

Si impiega di nuovo la funzione **with()** che specifica i dati da impiegare (**data**) e la funzione da applicare, che questa volta è la funzione **play3d()**, all'interno della quale:
→ la funzione **spin3d()** fa ruotare il grafico attorno all'asse specificato da **axis=c()** alla velocità di 3 giri al minuto (**rpm=3**);
→ il successivo argomento stabilisce per l'animazione una durata di 60 secondi (**duration=60**).

---

[1] Vedere il post Indice di massa corporea (BMI).

[2] Per la documentazione completa delle funzioni e dei rispettivi argomenti vedere il manuale di riferimento del pacchetto *rgl* sul repository della documentazione: *Available CRAN Packages By Name*. URL consultato il 10/11/2021: https://bit.ly/3zLwHWH

domenica 10 marzo 2024

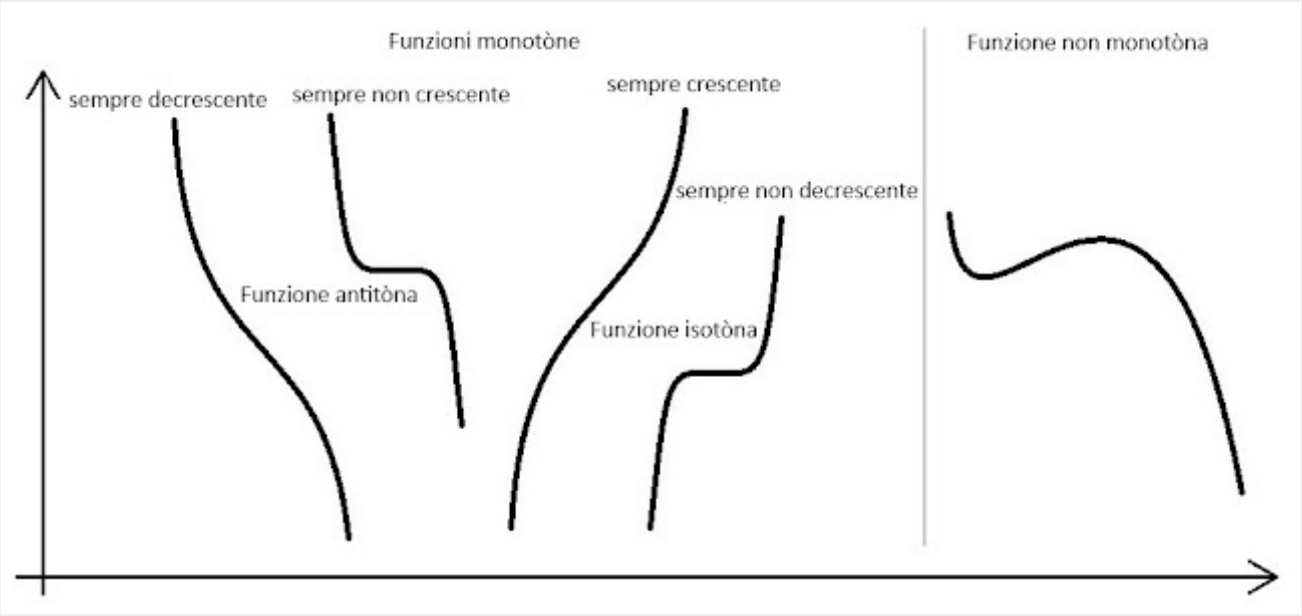
Regressione con una funzione monotòna (isotòna)

In assenza di un razionale che consenta la scelta di una funzione specifica (una retta o quant'altro) per approssimare una serie di dati, una possibile alternativa è rappresentata da una funzione con una forma determinata da condizioni poco restrittive.

Se partiamo dall'assunto minimo che essendo $x_1 < x_2 < \ldots < x_n$ all'aumentare di x la y corrispondente aumenti o al più resti uguale, siamo di fronte per definizione a una **funzione monotòna** (in quanto non presenta nè minimi né massimi) che è anche **isotòna** (in quanto sempre crescente o sempre non decrescente) ovvero:

→ *sempre crescente* quando $y_1 < y_2 < \ldots < y_n$

→ *sempre non decrescente* quando $y_1 \leq y_2 \leq \ldots \leq y_n$



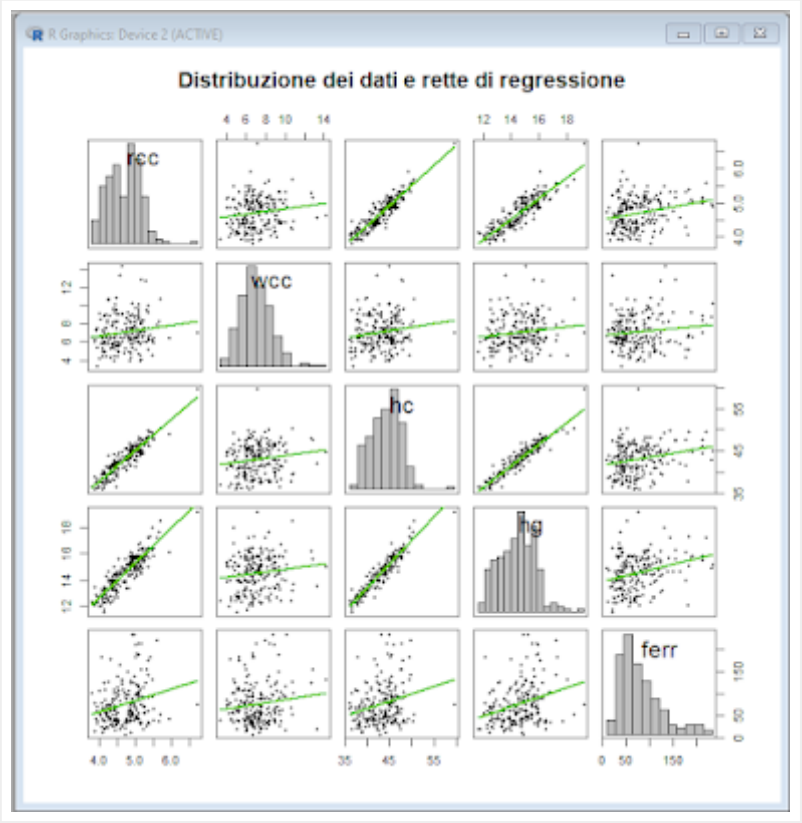
Vediamo ora come approssimare i dati con una **regressione isotòna** [1] impiegando le funzioni del pacchetto **isotone** [2] e del pacchetto **monreg** che dovete scaricare dal vostro **CRAN** preferito e dei quali è utile scaricare anche i rispettivi `Reference manual` [3].

Dovete inoltre accertarvi di avere installato – o dovete scaricare anche – il pacchetto **DAAG** [4] che contiene il set di dati **ais** impiegati nell'esempio e il pacchetto **car** che viene utilizzato per l'analisi preliminare dei dati.

Copiate e incollate nella `Console di R` questo script.

```
# ANALISI PRELIMINARE DEI DATI
#
library(DAAG) # carica il pacchetto incluso il set di dati ais
library(car) # carica il pacchetto per lo scatterplot
windows() # apre e inizializza una nuova finestra grafica
#
scatterplotMatrix(~rcc+wcc+hc+hg+ferr, col="black", pch=20, regLine=list(method=lm, lty=1, lwd=2, col="chartreuse3"), smooth=FALSE, diagonal=list(method="histogram", breaks="FD"), main="Distribuzione dei dati e rette di regressione", data=ais) # mostra il grafico
#
```

Dopo avere caricato il pacchetto (**DAAG**) che contiene i dati da analizzare e il pacchetto (**car**), con **windows()** viene aperta e inizializzata una nuova finestra nella quale comparirà il grafico predisposto con la funzione **scatterplotMatrix()** nella quarta e ultima riga di codice [5].



L'analisi preliminare dei dati ci dice che se in alcuni casi la regressione lineare sembra appropriata, in altri lo è molto meno. Qui ci focalizziamo su emoglobina (`hg`) e ferritina (`ferr`) [6] che mostrano distribuzioni dei dati non gaussiane, quindi in linea di principio inadatte all'impiego della regressione lineare ordinaria [7].

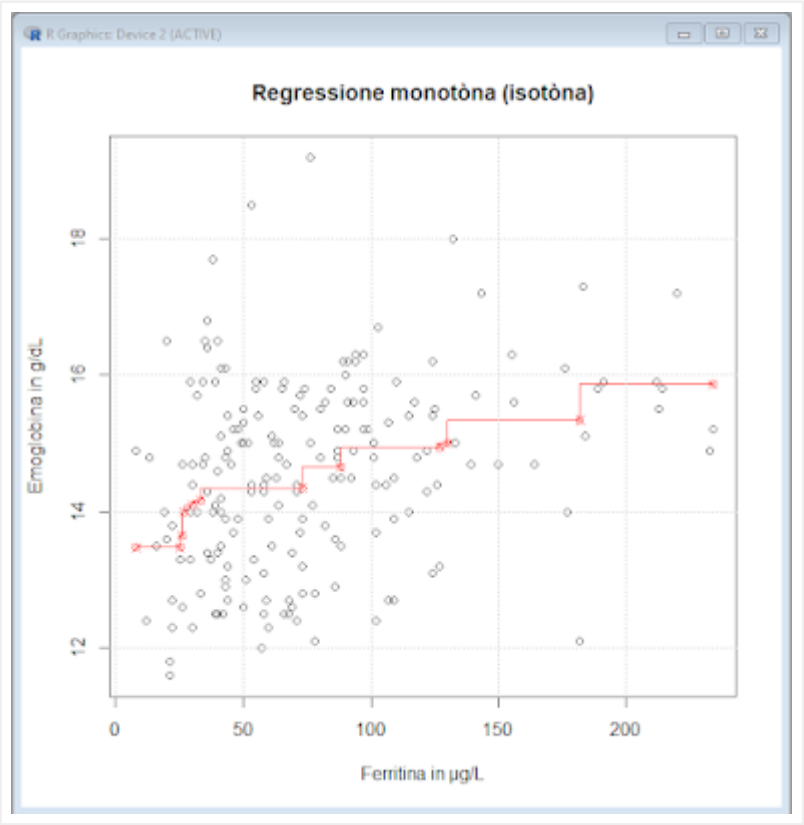
Ora copiate e incollate nella `Console di R` questo script.

```
# REGRESSIONE MONOTÒNA (ISOTÒNA) con il pacchetto isotone
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(isotone) # carica il pacchetto per la regressione
windows() # apre e inizializza una nuova finestra grafica
#
iso <- gpava(ais$ferr, ais$hg, solver=weighted.mean, ties="secondary") # calcola la regressione isotòna
#
plot(iso, main="Regressione monotòna (isotòna)", xlab="Ferritina in µg/L", ylab="Emoglobina in g/dL", col="red") # riporta il grafico con la distribuzione dei dati e sovrappone la regressione
#
```

Impieghiamo quindi il pacchetto **isotone** per calcolare la regressione isotòna con la funzione **gpava** che prevede come argomenti:
→ il valore in ascisse, nel nostro caso **ais$ferr**;
→ il valore in ordinate, nel nostro caso **ais$hg**;
→ l'argomento **solver** che consente di approssimare i dati impiegando la media (**weighted.mean**), la mediana (**weighted.median**), i frattili (**weighted.fractile**) pesati (qui i singoli pesi non sono specificati quindi per default sono uguali a 1);

→ l'argomento **ties** che consente di gestire l'ordinamento dei dati nel caso di valori identici, con **"primary"** l'ordinamento viene effettuato tra i blocchi di dati identici, non all'interno di questi, con **"secondary"** l'ordinamento viene effettuato tra i blocchi di dati identici, ma viene richiesta l'uguaglianza dei valori all'interno di questi, con **"tertiary"** viene effettuato solamente l'ordinamento delle medie dei blocchi di dati identici.

Nell'oggetto **iso** generato dalla funzione **gpava** sono contenute tutte le informazioni necessarie per riportare in un grafico con la funzione **plot()** dati e regressione, quindi è sufficiente aggiungere semplicemente titolo, legende e colore (**"red"**) della regressione.



Nel caso di una regressione una cosa che interessa è impiegare la funzione calcolata per interpolare nuovi valori, per farlo copiate e incollate nella `Console di R` questo breve addendum allo script.

```
# interpolazione lineare sulla regressione isotòna calcolata in precedenza
#
ris <- as.data.frame(cbind(z=iso$z, y=iso$x)) # riporta in una tabella i dati della regressione
risord <- ris[order(ris$z),] # ordina la tabella
#
reg <- approx(risord$z, risord$y, xout=c(15, 20, 40, 60, 80, 100, 110, 120, 150, 170, 190, 220), ties="ordered") # interpola i valori specificati
#
round(cbind(reg$x, reg$y), digits=1) # mostra i risultati dell'interpolazione
#
```

Dopo avere riportato i dati della regressione in una tabella e avere ordinato i valori in ascisse in ordine crescente (prerequisito indispensabile per effettuare una interpolazione), i valori di emoglobina corrispondenti a `15, 20, 40, 60, 80, 100, 110, 120, 150, 170, 190, 220` µg/L di ferritina sono calcolati mediante **interpolazione lineare** sui dati della regressione isotòna con la funzione **approx()** (terza riga di codice) e sono quindi mostrati (quarta riga) arrotondandoli a un decimale.

```
> round(cbind(reg$x, reg$y), digits=1) # mostra i risultati dell'interpolazione
      [,1] [,2]
```

```
 [1,]   15 13.5
 [2,]   20 13.5
 [3,]   40 14.4
 [4,]   60 14.4
 [5,]   80 14.7
 [6,]  100 14.9
 [7,]  110 14.9
 [8,]  120 14.9
 [9,]  150 15.3
[10,]  170 15.3
[11,]  190 15.9
[12,]  220 15.9
```

I valori da interpolare riportati nell'argomento **xout=c()** vanno ovviamente adattati al bisogno. In ogni caso se volete verificare la correttezza del calcolo potete farlo riportando sul grafico i risultati dell'interpolazione con questa semplice riga di codice.

```
# riporta sul grafico i punti interpolati
#
points(reg$x, reg$y, pch=16, col="blue")
#
```

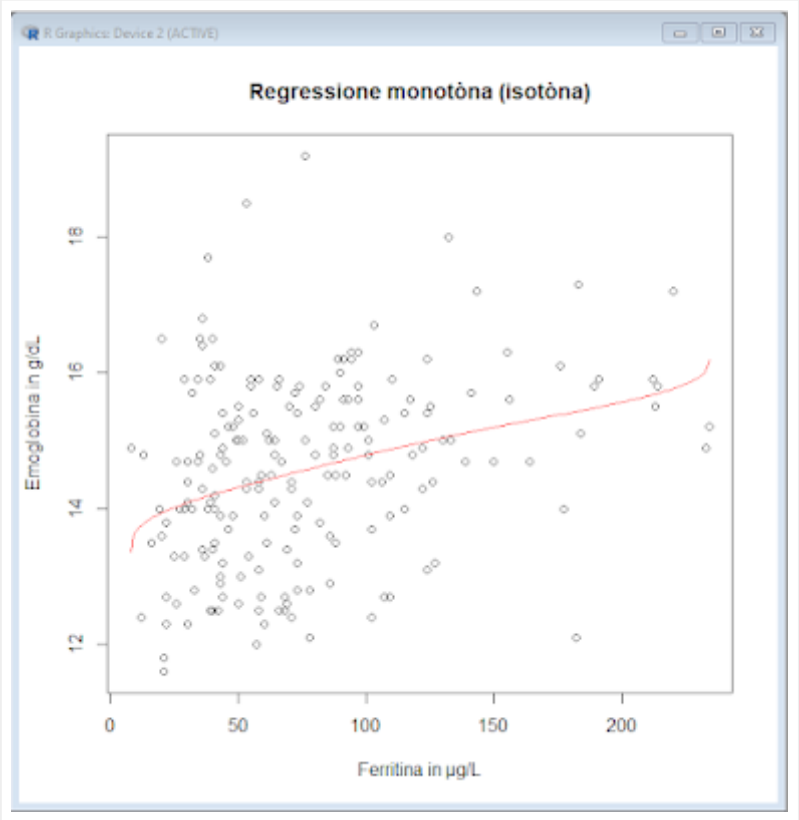
Per meglio sviluppare il tema vediamo ora la regressione calcolata con un metodo diverso, previsto nel pacchetto **monreg**, copiate e incollate nella `Console di R` questo script.

```
# REGRESSIONE MONOTÒNA (ISOTÒNA) con il pacchetto monreg
#
library(DAAG) # carica il pacchetto che include il set di dati ais
library(monreg) # carica il pacchetto per la regressione
windows() # apre e inizializza una nuova finestra grafica
#
mon <- monreg(ais$ferr, ais$hg, hd=.5, hr=.5, monotonie="isoton") # calcola la regressione isotòna
#
plot(ais$ferr, ais$hg, main="Regressione monotòna (isotòna)", xlab="Ferritina in µg/L",
ylab="Emoglobina in g/dL") # riporta il grafico con la distribuzione dei dati
lines(mon$t, mon$estimation, col="red") # sovrappone la regressione
#
```

In questo caso il passo più interessante si trova nella funzione **monreg()** ed è l'argomento **monotonie** che consente di selezionare le due soluzioni possibili per una **regressione monotòna**:
→ con **"isoton"** si calcola la **regressione isotòna** (crescente o sempre non decrescente) come facciamo in questo caso;
→ con **"antiton"** si calcola la **regressione antitòna** (decrescente o sempre non crescente), che è trattata a parte in un altro post [8].

Per gli altri argomenti della funzione rimando alla guida specifica richiamabile digitando **help(monreg)** nella `Console di R`.

Infine con **plot()** si riporta il grafico `xy` dei dati e con **lines()** si sovrappone la regressione le cui ascisse (**mon$t**) e ordinate (**mon$estimation**) sono state in precedenza salvate nell'oggetto **mon** generato con la funzione **monreg()**. Digitate **str(mon)** nella `Console di R` per visualizzare tutte le variabili salvate da **monreg()** nell'oggetto **mon**.



Ora copiate e incollate nella `Console di R` questo breve addendum allo script che riporta nuovamente – in quanto lievemente diverso e più semplice del precedente poiché la funzione **monreg()** restituisce i dati con la `x` già ordinata in ordine crescente – il codice necessario per effettuare l'interpolazione sulla regressione calcolata.

```
# interpolazione lineare sulla regressione isotòna calcolata in precedenza
#
reg <- approx(mon$t, mon$estimation, xout=c(15 , 20, 40, 60, 80, 100, 110, 120, 150, 170, 190, 220), ties="ordered") # interpola i valori specificati
#
round(cbind(reg$x, reg$y), digits=1) # mostra i risultati dell'interpolazione
#
```

Anche in questo caso i valori di emoglobina corrispondenti a `15, 20, 40, 60, 80, 100, 110, 120, 150, 170, 190, 220` µg/L di ferritina sono calcolati mediante interpolazione lineare sui dati della regressione isotòna con la funzione **approx()** (prima riga di codice), quindi sono mostrati (seconda riga).

```
> round(cbind(reg$x, reg$y), digits=1) # mostra i risultati dell'interpolazione
      [,1] [,2]
 [1,]   15 13.8
 [2,]   20 13.9
 [3,]   40 14.2
 [4,]   60 14.4
 [5,]   80 14.6
 [6,]  100 14.8
 [7,]  110 14.9
 [8,]  120 15.0
 [9,]  150 15.2
[10,]  170 15.3
[11,]  190 15.5
[12,]  220 15.8
```

Infine se volete verificare la correttezza del calcolo potete riportare sul grafico i risultati dell'interpolazione, per farlo copiate e incollate nella `Console di R` questa riga di codice, identica a quella già vista a conclusione dello script precedente.

```
# riporta sul grafico i punti interpolati
#
points(reg$x, reg$y, pch=16, col="blue")
#
```

---

[1] Nella letteratura statistica anglosassone trovate "*isotonic*" che talora viene erroneamente considerato sinonimo di "*monotonic*". Qui "*monotòna*" viene riportato con il significato matematico originario di "*funzione che non ha massimi e non ha minimi*" mentre "*isotòna*" e "*antitòna*" sono gli attributi aggiuntivi di una funzione monotòna quando è rispettivamente una "*funzione crescente o sempre non decrescente*" e quando è una "*funzione decrescente o sempre non crescente*".

[2] de Leeuw, J., Hornik, K., & Mair, P. (2009). *Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods*. Journal of Statistical Software, 32(5), 1–24. https://doi.org/10.18637/jss.v032.i05

[3] Vedere `Reference manual: isotone.pdf` al link:
https://cran.r-project.org/web/packages/isotone/index.html
e `Reference manual: monoreg.pdf` al link:
https://cran.r-project.org/web/packages/monoreg/index.html

[4] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati della tabella senza impiegare il pacchetto **DAAG**.

[5] Per i dettagli della funzione potete digitare **help(scatterplotMatrix)** nella `Console di R` ma trovate anche una breve spiegazione nel post Grafici di dispersione (grafici xy).

[6] L'emoglobina è la proteina che trasporta l'ossigeno contenuta nei globuli rossi del sangue, la sua concentrazione è espressa in grammi per decilitro di sangue (g/dL), la ferritina è la proteina collegata ai depositi di ferro presenti nell'organismo che contribuiscono a garantire una adeguata produzione di emoglobina, la sua concentrazione è espressa in microgrammi per litro di siero (µg/L).

[7] Vedere il post Regressione lineare semplice parametrica.

[8] Vedere il post Regressione con una funzione monotòna (antitòna) * in preparazione *

giovedì 18 aprile 2024

Multipli e sottomultipli delle unità di misura SI

Statistica significa analisi dei **dati**, cioè dei risultati di **processi di misura** espressi in termini di *grandezze*, di *unità di misura* e dei loro *multipli* e *sottomultipli*.

Ho riportato altrove [1] la storia che dalle antiche misure antropomorfe, attraverso la svolta che è seguita alla nascita della scienza moderna, ha condotto anche l'Italia a riconoscere la necessità di adottare un linguaggio delle misure rigoroso e ampiamente concordato e condiviso a livello scientifico, oltre che in grado di risolvere le esigenze pratiche quotidiane, prima con l'adesione nel 1875 al **Sistema metrico decimale** e successivamente adottando per legge nel 1982 il **Sistema internazionale di unità (SI)** che ne rappresenta la moderna evoluzione [2].

Così come si trova citato anche nella *Appendice 1* della *Brochure* del SI [3], nel 2022 la 27a CGPM [4] con una delle sue risoluzioni [5] ha aggiunto, ai precedenti, due altri multipli (ronna e quetta) e due altri sottomultipli (ronto e quecto), quindi per indicare multipli e sottomultipli delle unità di misura nel SI sono ad oggi previsti i *prefissi*, i *simboli* e i corrispondenti *fattori di moltiplicazione* riportati in questa tabella:

Prefisso	Simbolo	Fattore di moltiplicazione	Fattore di moltiplicazione per esteso
quetta	Q	10^{30}	1 000 000 000 000 000 000 000 000 000 000
ronna	R	10^{27}	1 000 000 000 000 000 000 000 000 000
yotta	Y	10^{24}	1 000 000 000 000 000 000 000 000
zetta	Z	10^{21}	1 000 000 000 000 000 000 000
exa	E	10^{18}	1 000 000 000 000 000 000
peta	P	10^{15}	1 000 000 000 000 000
tera	T	10^{12}	1 000 000 000 000
giga	G	10^{9}	1 000 000 000
mega	M	10^{6}	1 000 000
kilo	k	10^{3}	1 000
etto	h	10^{2}	100
deca	da	10^{1}	10
deci	d	10^{-1}	0,1
centi	c	10^{-2}	0,01
milli	m	10^{-3}	0,001
micro	µ	10^{-6}	0,000 001
nano	n	10^{-9}	0,000 000 001
pico	p	10^{-12}	0,000 000 000 001
femto	f	10^{-15}	0,000 000 000 000 001
atto	a	10^{-18}	0,000 000 000 000 000 001
zepto	z	10^{-21}	0,000 000 000 000 000 000 001
yocto	y	10^{-24}	0,000 000 000 000 000 000 000 001
ronto	r	10^{-27}	0,000 000 000 000 000 000 000 000 001
quecto	q	10^{-30}	0,000 000 000 000 000 000 000 000 000 001

Lo scopo è evidente: se il numero di cifre diventa eccessivo e si trova ostica per l'uso corrente e non si vuole impiegare la notazione scientifica [6], ma si preferisce esprimere in forma verbale i numeri superiori al milione, i prefissi SI consentono di evitare la trappola delle denominazioni che in tutta

Europa con la *scala lunga* [7], e negli USA e in molti altri Paesi con la *scala corta*, prevedono di impiegare gli stessi termini per indicare numeri completamente diversi, come qui evidenziato,

Prefisso	Simbolo	Fattore di moltiplicazione	Scala lunga (Europa)	Scala corta (Paesi anglosassoni)
quetta	Q	10^{30}	quintilione	nonilione (nonillion)
ronna	R	10^{27}	quadriliardo (mille quadrilioni)	ottilione (octillion)
yotta	Y	10^{24}	quadrilione	settilione (septillion)
zetta	Z	10^{21}	triliardo (mille trilioni)	sestilione (sextillion)
exa	E	10^{18}	trilione	quintilione (quintillion)
peta	P	10^{15}	biliardo (mille bilioni)	quadrilione (quadrillion)
tera	T	10^{12}	bilione	trilione (trillion)
giga	G	10^{9}	miliardo (mille milioni)	bilione (billion)
mega	M	10^{6}	milione	milione (million)

un fatto che determina confusione e possibili errori nella interpretazione dei numeri espressi in forma verbale quando questi sono riportati da una lingua all'altra [8].

Pertanto – giusto per fare un esempio – per la constatazione che "... a*s of April 2024 Apple has a market cap of $2.547 Trillion...*" [9] è opportuno precisare che i 2.547 trilioni di dollari di capitalizzazione dell'Apple sono espressi nella scala corta dei Paesi anglosassoni, e quindi sono $2.547 \cdot 10^{12}$ dollari, e non sono i $2.547 \cdot 10^{18}$ dollari corrispondenti ai trilioni della scala lunga.

Invece nessun dubbio vi può essere nel caso della larghezza di banda espressa in bit (b) di una rete a 100 Mb/s (100 megabit al secondo ovvero $100 \cdot 10^{6}$ bit al secondo) e della capacità espressa in byte (B) di una RAM di 16 GB (16 gigabyte ovvero $16 \cdot 10^{9}$ byte) o di un hard-disk di 4 TB (4 terabyte ovvero $4 \cdot 10^{12}$ byte), dato il significato univoco dei prefissi SI.

---

[1] *Grandezze e unità di misura. Breve storia dall'antichità al Sistema Internazionale di Unità (SI)*. Vedere il post:
https://impararfacendo.blogspot.com/2020/10/grandezze-e-unita-di-misura.html
Il testo può essere liberamente scaricato (e senza pubblicità diretta nè occulta) da questi siti
https://www.academia.edu/41041923/
https://books.google.it/books?id=GciVDwAAQBAJ
come pure dal mio sito personale
https://www.bayes.it/SI.pdf

[2] I riferimenti normativi completi altre che nel testo citato sono riportati nel post Scale nominali, scale ordinali e scale numeriche.

[3] *SI Brochure: The International System of Units (SI)* – 9a edizione, 2019.
https://www.bipm.org/en/publications/si-brochure

[4] Sigla della "Conferenza Generale dei Pesi e delle Misure", in sostanza il parlamento attraverso il quale si esprimono in tema di misure gli Stati che hanno adottato il SI.

[5] Vedere: *Resolution 3 of the 27th CGPM (2022). On the extension of the range of SI prefixes. The General Conference on Weights and Measures (CGPM), at its 27th meeting*.
https://www.bipm.org/en/cgpm-2022/resolution-3

[6] Vedere il post Come riportare i numeri in formato fisso.

[7] La scala lunga la si trova riportata anche in Italia nella legislazione, come ad esempio nella tabella a pagina 12 dell'Allegato al *DECRETO MINISTERIALE 4 settembre 1996 Attuazione della direttiva 94/55/CE del Consiglio concernente il ravvicinamento delle legislazioni degli Stati membri relative al trasporto di merci pericolose su strada. (GU Serie Generale n.282 del 02-12-1996 - Suppl. Ordinario n. 211)*.

I multipli e sottomultipli decimali di una unità possono essere formati mediante i prefissi o simboli seguenti, posti davanti il nome o davanti il simbolo della unità:

Fattore		Prefisso	Simbolo
$1\,000\,000\,000\,000\,000\,000 = 10^{18}$	trilione	esa	E
$1\,000\,000\,000\,000\,000 = 10^{15}$	biliardo	peta	P
$1\,000\,000\,000\,000 = 10^{12}$	bilione	tera	T
$1\,000\,000\,000 = 10^{9}$	miliardo	giga	G
$1\,000\,000 = 10^{6}$	milione	mega	M
$1\,000 = 10^{3}$	mille	chilo	k
$100 = 10^{2}$	cento	etto	h
$10 = 10^{1}$	dieci	deca	da
$0{,}1 = 10^{-1}$	decimo	deci	d
$0{,}01 = 10^{-2}$	centesimo	centi	c
$0{,}001 = 10^{-3}$	millesimo	milli	m
$0{,}000\,001 = 10^{-6}$	milionesimo	micro	μ
$0{,}000\,000\,001 = 10^{-9}$	miliardesimo	nano	n
$0{,}000\,000\,000\,001 = 10^{-12}$	bilionesimo	pico	p
$0{,}000\,000\,000\,000\,001 = 10^{-15}$	biliardesimo	femto	f
$0{,}000\,000\,000\,000\,000\,001 = 10^{-18}$	trilionesimo	atto	a

— 12 —

[8] Vedere: *Long and short scales*.
https://en.wikipedia.org/wiki/Long_and_short_scales

[9] Dati tratti da: *Largest Companies by Market Cap*
https://companiesmarketcap.com/

domenica 2 giugno 2024

Grafici di dispersione (scatterplot) con ggplot

I **grafici di dispersione** (*scatter plot* o *scatterplot*) forniscono un ottimo esempio delle potenzialità delle funzioni del pacchetto **ggplot2** [1] nella rappresentazione grafica dei dati. Funzioni che possono risultare ostiche a chi le affronta per la prima volta, ma che consentono risultati che non si riescono a ottenere impiegando esclusivamente le funzioni base di **R** [2].

Oltre al pacchetto **ggplot2** bisogna avere installato – o bisogna scaricare anche – il pacchetto **DAAG** [3] che contiene i dati impiegati nell'esempio e il pacchetto **gridExtra** che nel nostro caso è necessario per combinare più grafici in una sola figura.

Copiate e incollate nella `Console di R` questo script e premete `↵ Invio`.

```
# SCATTERPLOT con ggplot
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(ggplot2) # carica il pacchetto per la grafica
library(gridExtra) # carica il pacchetto per combinare i grafici in una sola figura
#
# scatterplot
plot1 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point()
#
# con regressione loess
plot2 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point() + geom_smooth()
#
# con regressione lineare
plot3 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point() + geom_smooth(method="lm")
#
# con la densità di distribuzione
plot4 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color="red")) +
geom_density_2d() + theme_classic() + labs(title="Densità di distribuzione \n tutti i
casi", x="Eritrociti in 10^6/µL", y="Emoglobina in g/dL") +
theme(plot.title=element_text(hjust=0.5), legend.position="none")
#
# combina i grafici in una sola figura
grid.arrange(plot1, plot2, plot3, plot4, nrow=2, ncol=2)
#
```

Lo script è semplice: dopo avere caricato i pacchetti necessari, sono generati quattro grafici che sono salvati separatamente per essere poi, al termine, combinati in una sola figura.

Per realizzare il primo grafico (**plot1**) viene impiegata la funzione **ggplot()** che inizializza l'oggetto al quale sono collegate con il segno più [**+**] le successive funzioni, che sviluppano la grafica e che prevede:
→ come primo argomento il nome della tabella che contiene i dati (**ais**) [3];
→ come secondo argomento la funzione **aes()** che specifica la variabile da riportare in ascisse (**rcc**) e la variabile da riportare in ordinate (**hg**), che sono rispettivamente la concentrazione degli eritrociti nel sangue espressa in milioni per microlitro di sangue (`10^6/µL`) e la concentrazione di emoglobina espressa in grammi per decilitro di sangue (`g/dL`).

Alla funzione **ggplot()** viene quindi collegata con il segno più [**+**] la funzione **geom_point()** che realizza il *grafico xy* lasciando per tutti gli argomenti i valori previsti di default.

Da notare subito che questa è la struttura base che rimane identica in tutti e quattro i grafici realizzati, che si differenziano tra loro per le componenti grafiche via via aggiunte.

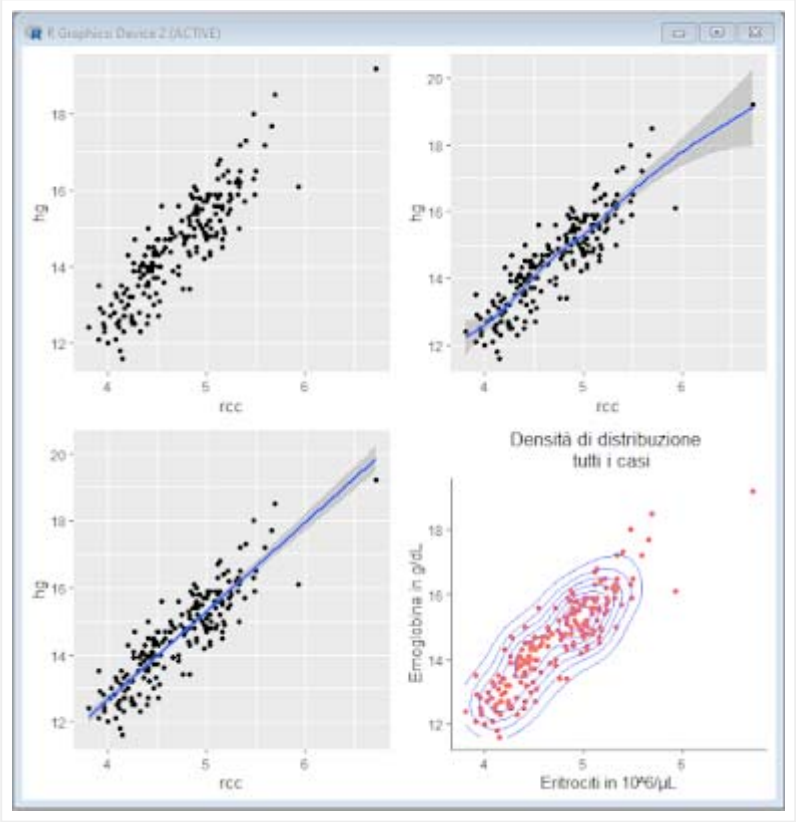
Il secondo grafico (**plot2**) quindi riprende tal quale la precedente struttura base e aggiunge semplicemente tramite la funzione **geom_smooth()**, la *regressione loess* prevista di default, con i suoi limiti di confidenza (la regressione loess è una regressione che collega tra loro una serie di regressioni polinomiali, quelle che localmente meglio si adattano ai dati).

Nel terzo grafico (**plot3**) impiegando l'argomento **method** la regressione loess è sostituita con la usuale retta di regressione (**"lm"**), riportando anche in questo caso i limiti i confidenza. Se non li volete rappresentare correggete la riga di codice sostituendola completamente con la seguente o semplicemente aggiungendo quanto qui riportato in colore

```
# con regressione lineare
plot3 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point() +
geom_smooth(method="lm", se=FALSE)
#
```

Il quarto grafico (**plot4**) riprende la struttura base aggiungendo:
→ con **aes(color="red")** un colore ai punti che rappresentano i dati;
→ con la funzione **geom_density_2d()** la rappresentazione sul piano dei livelli di densità nella distribuzione dei dati;
→ con la funzione **theme_classic()** il cambiamento di stile che risulta evidente nell'ultimo grafico;
→ con la funzione **labs()** il titolo e le etichette degli assi, Da notare il simbolo **\n** che all'interno di una stringa di testo determina il ritorno a capo;
→ con la funzione **theme()** il codice (abbastanza arzigogolato) che determina la centratura del titolo e il valore **"none"** per l'argomento **legend.position** che determina la posizione della legenda, che pertanto non viene riportata.

Infine con la funzione **grid.arrange()** i quattro grafici sono combinati in un'unica figura.



Ora copiate e incollate nella `Console di R` questo secondo script che ha come obiettivo evidenziare la semplicità con la quale possiamo effettuare in un grafico rappresentazioni separate in base a un

fattore che nel nostro caso è la variabile sesso (`sex`), in quanto il set di dati **ais** contiene dati ematologici e dati biometrici, rilevati in atleti australiani sia di sesso femminile sia di sesso maschile.

```
# SCATTERPLOT con ggplot (differenziaziati in base a un fattore)
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(ggplot2) # carica il pacchetto per la grafica
library(gridExtra) # carica il pacchetto per combinare i grafici in una sola figura
#
# scatterplot per sesso
plot1 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex))
#
# con simboli dei punti differenziati
plot2 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex, shape=sex)) +
theme(legend.position="none")
#
# con rette di regressione e limiti di confidenza
plot3 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex, shape=sex)) +
theme(legend.position="none") + geom_smooth(aes(group=sex, color=sex),
method="lm", se=TRUE)
#
# con rette di regressione estese
plot4 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex, shape=sex)) +
theme(legend.position="none") + geom_smooth(aes(group=sex, color=sex),
method="lm", se=FALSE, fullrange=TRUE)
#
# combina i grafici in una sola figura
grid.arrange(plot1, plot2, plot3, plot4, nrow=2, ncol=2)
#
```

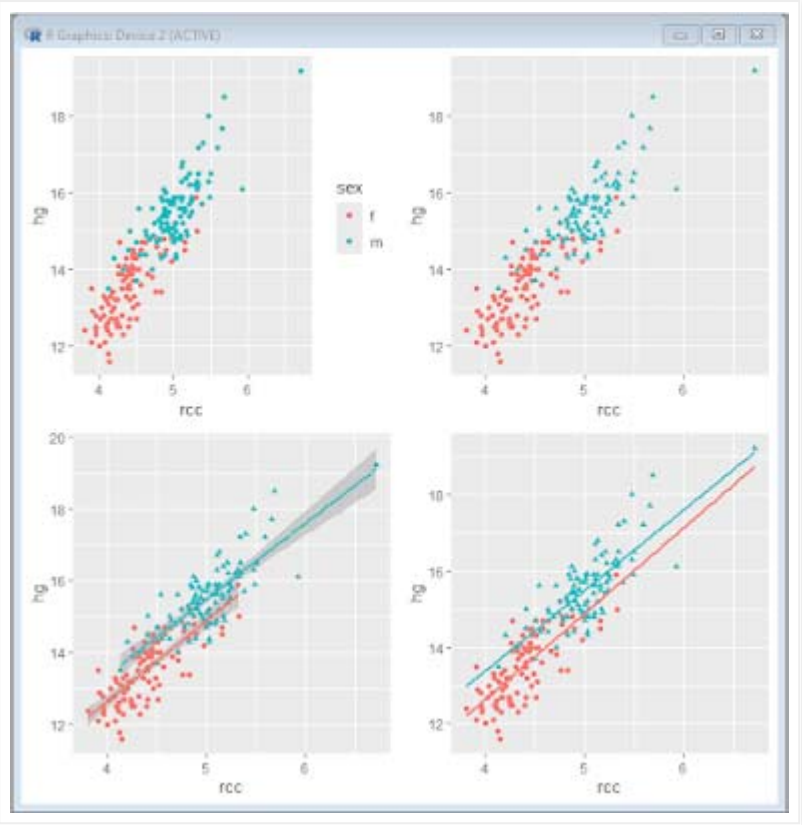
Per realizzare il primo grafico (**plot1**) viene impiegata la struttura base già vista nello script precedente, ma con la semplice aggiunta a **geom.point** della funzione **aes()** che con l'argomento **color** posto uguale a **sex** riporta i dati con un colore differenziato in base al sesso.

Il secondo grafico (**plot2**) riprende tal quale il codice precedente ma:
→ alla funzione **geom_point()** aggiunge l'argomento **shape=sex** che automaticamente riporta un simbolo differente in base al sesso;
→ con la funzione **theme()** esclude dal grafico la legenda.

Il terzo grafico (**plot3**) è identico al secondo ma con la funzione **geom_smooth()** aggiunge la retta di regressione ordinaria (**method="lm"**) con i limiti di confidenza (**se=TRUE**) separatamente (**group=sex**) e con un colore diverso (**color=sex**) per ciascun sesso.

Il quarto grafico (**plot4**) riprende esattamente il codice del terzo ma con **se=FALSE** esclude la rappresentazione dei limiti di confidenza e con **fullrange=TRUE** riporta la retta di regressione a tutto campo e al di la dei limiti dei dati osservati (per inciso si ricorda che, a meno di casi particolari e adeguatamente giustificati, l'estrapolazione non andrebbe mai fatta, ma viene qui riportata per completezza).

Infine con la funzione **grid.arrange()** i quattro grafici sono di nuovo combinati, per semplicità, in un'unica figura.



Ora copiate e incollate nella `Console di R` questo script e premete ↲ `Invio`.

```
# SCATTERPLOT con ggplot (densità delle distribuzioni)
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(ggplot2) # carica il pacchetto per la grafica
library(gridExtra) # carica il pacchetto per combinare i grafici in una sola figura
#
# scatterplot con ellissi
plot1 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex, shape=sex)) +
theme(legend.position="none") + stat_ellipse(aes(color=sex))
#
# con densità di distribuzione della y
plot2 <- ggplot(ais, aes(y=hg, fill=sex)) + geom_density(alpha=0.5) +
theme(legend.position="none")
#
# con densità di distribuzione della x
plot3 <- ggplot(ais, aes(x=rcc, fill=sex)) + geom_density(alpha=0.5) +
theme(legend.position="none")
#
# con livelli di densità separati per sesso
plot4 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex, shape=sex)) +
theme(legend.position="none") + geom_density_2d(aes(color=sex))
#
# combina i grafici in una sola figura
grid.arrange(plot1, plot2, plot3, plot4, nrow=2, ncol=2)
#
```

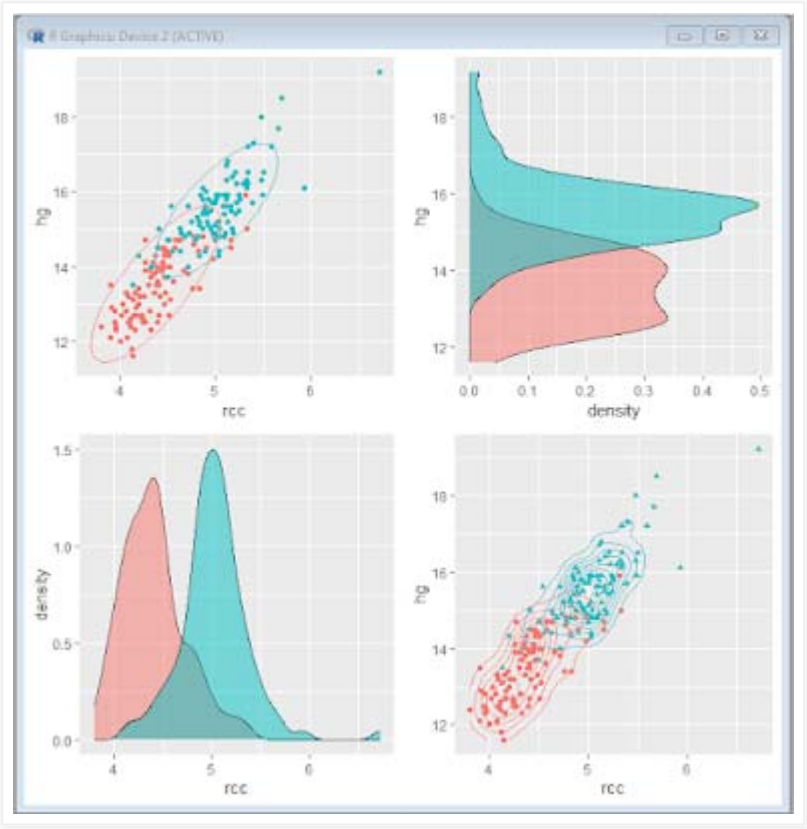
Il primo grafico (**plot1**) riprende le funzioni già note, ma aggiunge, con la funzione **stat_ellipse()**, le ellissi che riportano i limiti di confidenza al 95% delle distribuzioni dei dati, separate per ciascun sesso e ulteriormente analizzate con il secondo e il terzo grafico.

Il secondo grafico (**plot2**) riporta in ordinate per la variabile emoglobina (**y=hg**) e separatamente per ciascun sesso (**fill=sex**), con la funzione **geom_density()**, il kernel density plot, con un colore trasparente al 50% (**alpha=0.5**) per consentirne la sovrapposizione. Da notare due aspetti non banali gestiti automaticamente dalle funzioni qui impiegate: gli assi sono ruotati nella posizione corretta e per l'asse delle ordinate la scala nella quale sono espressi i valori di concentrazione dell'emoglobina (**hg**) è allineata con quella del primo grafico.

Il terzo grafico (**plot3**) riporta in modo analogo ma in ascisse il kernel density plot per sesso della variabile concentrazione degli eritrociti (**x=rcc**). Da notare che anche in questo caso gli assi sono automaticamente ruotati nella posizione corretta e per l'asse delle ascisse la scala nella quale sono espressi i valori di concentrazione degli eritrociti (**rcc**) è automaticamente allineata con quella del primo grafico.

In definitiva i primi tre grafici formano un tutt'uno organico – con le variabili rappresentate contemporaneamente sia singolarmente in forma di grafico xy sia in forma di grafici che ne riportano in continuo la densità (kernel density plot) – che fornisce un'analisi molto interessante dei dati a riprova dell'utilità delle funzioni incluse nel pacchetto.

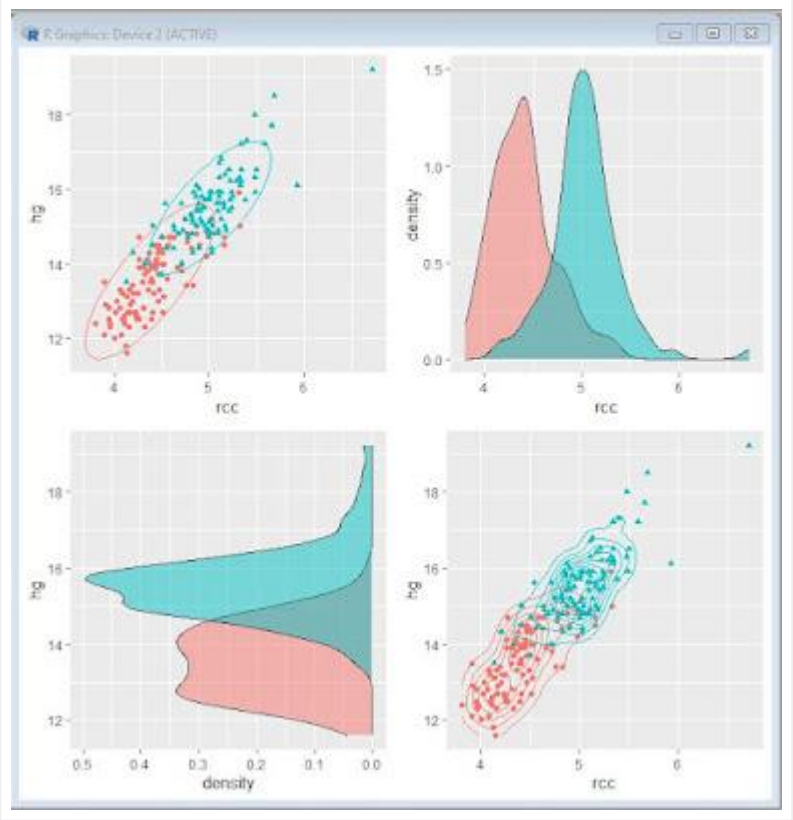
Il quarto grafico (**plot4**) riprende esattamente il primo aggiungendo questa volta al posto delle ellissi con la funzione **geom_density_2d()** la rappresentazione sul piano dei livelli di densità nella distribuzione dei dati.



Se vedendo il grafico notate che preferireste (io lo preferisco di gran lunga) che i due kernel density plot fossero riferiti al grafico in basso a destra, niente di più facile, copiate e incollate nella `Console di R` questo script e premete `↲ Invio`.

```
# SCATTERPLOT con ggplot (densità delle distribuzioni)
#
library(DAAG) # carica il pacchetto DAAG che include il set di dati ais
library(ggplot2) # carica il pacchetto per la grafica
library(gridExtra) # carica il pacchetto per combinare i grafici in una sola figura
#
# scatterplot con ellissi
plot1 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex, shape=sex)) +
theme(legend.position="none") + stat_ellipse(aes(color=sex))
#
# con densità di distribuzione della y
plot2 <- ggplot(ais, aes(y=hg, fill=sex)) + geom_density(alpha=0.5) +
theme(legend.position="none") + scale_x_reverse()
#
# con densità di distribuzione della x
plot3 <- ggplot(ais, aes(x=rcc, fill=sex)) + geom_density(alpha=0.5) +
theme(legend.position="none")
#
# con livelli di densità separati per sesso
plot4 <- ggplot(ais, aes(x=rcc, y=hg)) + geom_point(aes(color=sex, shape=sex)) +
theme(legend.position="none") + geom_density_2d(aes(color=sex))
#
# combina i grafici in una sola figura
grid.arrange(plot1, plot3, plot2, plot4, nrow=2, ncol=2)
#
```

Come si vede il codice è identico al precedente, a parte il fatto che dovete:
→ scambiare tra di loro i grafici (**plot3, plot2,**) quando li combinate in un'unica figura (ultima riga di codice), portando quindi il kernel density plot degli eritrociti (`rcc`) in alto a destra e quello dell'emoglobina (`hg`) in basso a sinistra;
→ aggiungere con (**+**) al **plot2** la funzione **scale_x_reverse()** per invertire la scala dell'asse delle ascisse e quindi la rappresentazione del kernel density plot.



Ricordate, perché prima o poi vi tornerà utile, che qualora fosse necessario potete anche invertire la scala dell'asse delle ordinale con **scale_y_reverse()** e scambiare di posizione gli assi con **coord_flip()**.

In questo e negli altri post nei quali il pacchetto è stato (e sarà) impiegato [4] ho cercato (e cercherò) di riportare alcuni esempi dell'interessante ausilio che una analisi grafica con **ggplot2** può fornire nella introspezione dei dati e nella comunicazione dell'informazione in essi contenuta.

Ovviamente potete procedere con ulteriori approfondimenti impiegando la documentazione del pacchetto [1], ma potete anche valutare l'opportunità di ricorrere a qualcuno dei corsi sull'argomento disponibili sul web [5].

---

[1] Vedere la documentazione e il manuale di riferimento su:
https://cran.r-project.org/web/packages/ggplot2/index.html

[2] Vedere ad esempio il post Grafici di dispersione (grafici xy).

[3] Vedere il post Il set di dati ais nel quale trovate anche come caricare i dati senza impiegare il pacchetto **DAAG**.

[4] Fate click su ggplot2 nelle **Parole chiave** o, per una ricerca meno selettiva, digitate ggplot nella casella **Cerca nel blog** quindi fate click su Cerca.

[5] Vedasi ad esempio "*Introduction to Data Visualization with ggplot2*":
https://www.datacamp.com/courses/introduction-to-data-visualization-with-ggplot2

venerdì 12 luglio 2024

Stacco estivo

"*Sai ched'è la statistica? È 'na cosa / che serve pe' fa' un conto in generale / de la gente che nasce, che sta male, / che more, che va in carcere e che spósa. / Ma pe' me la statistica curiosa / è dove c'entra la percentuale, / pe' via che, lì, la media è sempre eguale / puro co' la persona bisognosa. / Me spiego: da li conti che se fanno / secondo le statistiche d'adesso / risurta che te tocca un pollo all'anno: / e, se nun entra ne le spese tue, / t'entra ne la statistica lo stesso / perché c'è un antro che ne magna due.*"
(*Trilussa*)

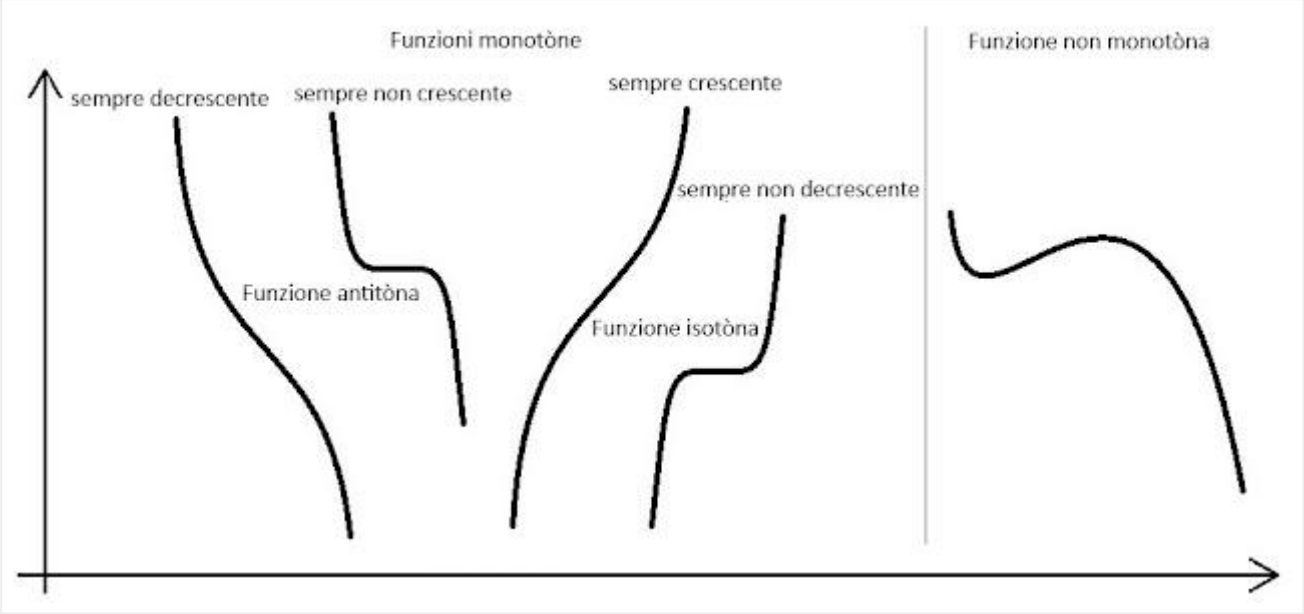
"*Nei tempi antichi non c'erano le statistiche, perciò era necessario ripiegare sulle menzogne.*"
(*Stephen Leacock*)

domenica 18 agosto 2024

Regressione con una funzione monotòna (antitòna)

Come riportato anche altrove [1], in assenza di un razionale che consenta la scelta di una funzione specifica (retta o altro) per approssimare una serie di dati, una possibile alternativa è rappresentata da una funzione con una forma determinata da condizioni poco restrittive.

Se partiamo dall'assunto minimo che essendo $x_1 < x_2 < \ldots < x_n$ all'aumentare di x la y corrispondente diminuisca o al più resti uguale, siamo di fronte per definizione a una **funzione monotòna** che è anche **antitòna** (in quanto sempre decrescente o sempre non crescente) ovvero:
→ *sempre decrescente* quando $y_1 > y_2 > \ldots > y_n$
→ *sempre non crescente* quando $y_1 \geq y_2 \geq \ldots \geq y_n$



Vediamo ora come approssimare i dati con una **regressione antitòna** [2] impiegando sia il pacchetto **fdrtool** sia il pacchetto **monreg** che dovete scaricare dal vostro **CRAN** preferito. Inoltre dovete scaricare anche il pacchetto **car** che viene utilizzato per l'analisi preliminare dei dati.

Come dati impieghiamo i valori di BMI (indice di massa corporea) rilevati a livello europeo alcuni anni fa e pubblicati dall'Istat [3].

Per proseguire è necessario:
→ effettuare il download del file di dati `bmi.csv`
→ salvare il file nella cartella `C:\Rdati\`

Per il file di dati trovate link e modalità di download alla pagina Dati, ma potete anche semplicemente copiare i dati riportati qui sotto aggiungendo un ↵ `Invio` al termine dell'ultima riga e salvarli in `C:\Rdati\` in un file di testo denominato `bmi.csv` (assicuratevi che il file sia effettivamente salvato con l'estensione `.csv`).

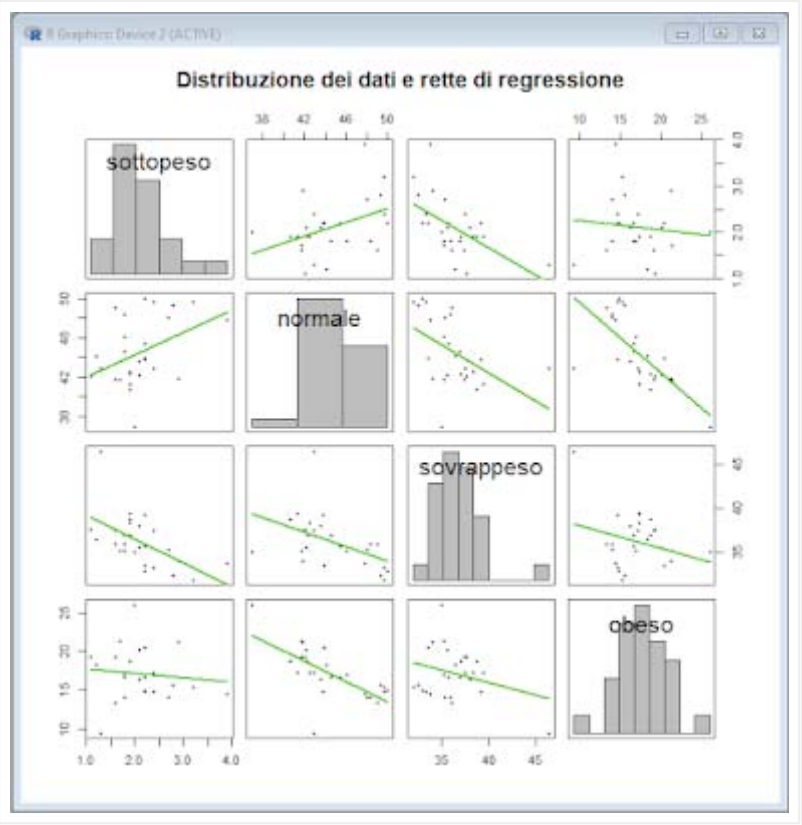
Nazione;sottopeso;normale;sovrappeso;obeso
Austria;2.4;49.6;33.3;14.7
Belgio;2.7;48.0;35.3;14.0
Bulgaria;2.2;43.8;39.2;14.8

Cipro;3.9;47.8;33.8;14.5
Croazia;1.9;40.7;38.7;18.7
Danimarca;2.2;50.0;32.9;14.9
Estonia;2.2;43.9;33.5;20.4
Finlandia;1.2;44.1;36.4;18.3
Francia;3.2;49.6;31.9;15.3
Germania;1.8;46.1;35.2;16.9
Grecia;1.9;41.3;39.4;17.3
Irlanda;1.9;42.3;37.0;18.7
Lettonia;1.7;41.8;35.2;21.3
Lituania;1.9;42.5;38.3;17.3
Lussemburgo;2.8;49.3;32.4;15.6
Malta;2.0;37.0;35.0;26.0
Olanda;1.6;49.0;36.0;13.3
Polonia;2.4;42.9;37.5;17.2
Portogallo;1.8;44.6;36.9;16.6
Regno Unito;2.1;42.2;35.6;20.1
Repubblica Ceca;1.1;42.1;37.6;19.3
Romania;1.3;42.9;46.4;9.4
Slovacchia;2.1;43.6;38.0;16.3
Slovenia;1.6;41.8;37.4;19.2
Spagna;2.2;45.4;35.7;16.7
Svezia;1.8;48.3;35.9;14.0
Ungheria;2.9;41.9;34.0;21.2

Ora copiate e incollate nella `Console di R` questo script e premete `↵ Invio`.

```
# ANALISI PRELIMINARE DEI DATI
#
library(car) # carica il pacchetto per la grafica
bmi <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";", row.names="Nazione") #
importa i dati
windows() # apre e inizializza una nuova finestra grafica
#
scatterplotMatrix(~sottopeso+normale+sovrappeso+obeso, col="black", pch=20, regLine
= list(method=lm, lty=1, lwd=2, col="chartreuse3"), smooth=FALSE,
diagonal=list(method ="histogram", breaks="FD"), main="Distribuzione dei dati e rette di
regressione", data=bmi) # mostra il grafico
#
```

Dopo avere caricato il pacchetto pacchetto (**car**) per l'analisi preliminare dei dati, questi sono importati nell'oggetto **bmi**, quindi con **windows()** viene aperta e inizializzata una nuova finestra nella quale comparirà il grafico predisposto con la funzione **scatterplotMatrix()** nella quarta e ultima riga di codice [4].



L'analisi preliminare ci dice che, tranne forse per la relazione obeso-normale, la regressione lineare mal si adatta a dati così dispersi. Qui ci focalizziamo sulla relazione che intercorre tra soggetti obesi (`obeso`) e soggetti in sovrappeso (`sovrappeso`) nella quale la dispersione dei dati è tale da mettere in dubbio l'opportunità di impiegare una regressione lineare.

Ora copiate e incollate nella `Console di R` questo script e premete `↵ Invio`.

```
# REGRESSIONE MONOTÒNA (ANTITÒNA) con il pacchetto fdrtool
#
library(fdrtool) # carica il pacchetto per la regressione
bmi <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";", row.names="Nazione") # importa i dati
windows() # apre e inizializza una nuova finestra grafica
#
fdr <- monoreg(bmi$obeso, bmi$sovrappeso, type="antitonic") # calcola la regressione antitòna
#
plot(bmi$obeso, bmi$sovrappeso, main="Regressione monotòna (antitòna)",
xlab="Soggetti obesi in %", ylab="Soggetti sovrappeso in %") # riporta il grafico con la distribuzione dei dati
lines(fdr$x, fdr$yf, col="red") # sovrappone la regressione
#
```

Dopo avere caricato il pacchetto e i dati, nella funzione **monreg()** l'argomento **type** consente di selezionare le due soluzioni possibili per una regressione monotòna:
→ con **"antitonic"** si calcola la **regressione antitòna** (decrescente o sempre non crescente) come facciamo in questo caso;
→ con **"isotonic"** si calcola la **regressione isotòna** (crescente o sempre non decrescente), che è trattata a parte in un altro post [1].

I risultati della regressione calcolata con la funzione **monoreg()** sono salvati nell'oggetto **fdr** mentre compare un messaggio che avverte che alcuni valori in ascisse sono duplicati (se osservate, nella variabile `obeso` compare due volte il valore `14.0` e due volte compaiono anche i

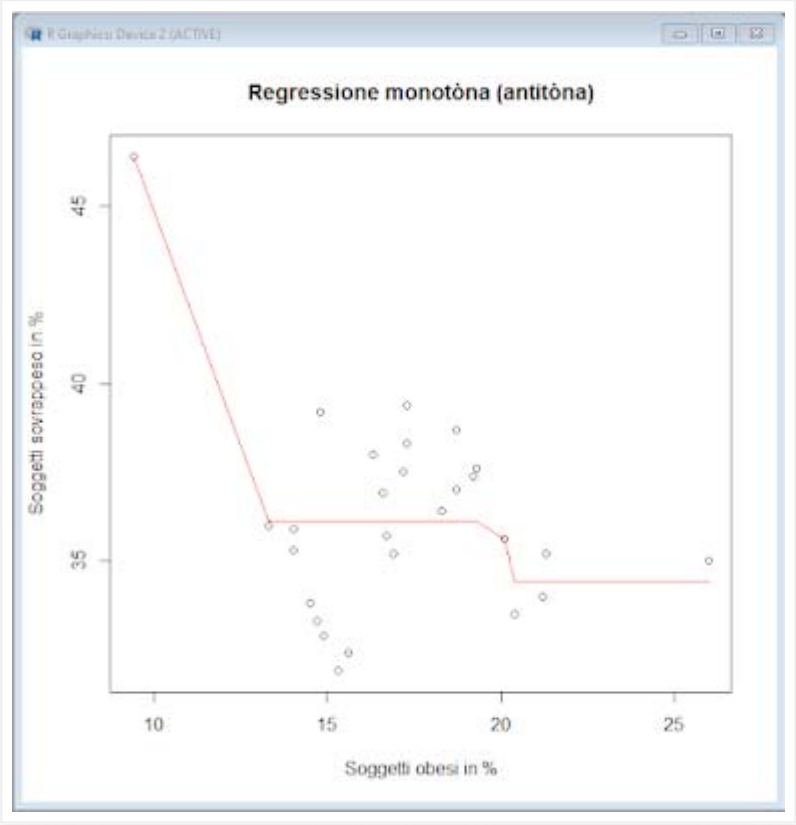
valori `17.3` e `18.7`): ma non si tratta di un problema in quanto per questi valori la funzione provvede automaticamente a unire le `y` corrispondenti come riportato nel manuale del pacchetto [5].

Quindi con la funzione **plot()** si riporta il grafico `xy` dei dati aggiungendo titolo ed etichette degli assi.

Infine con **lines()** si sovrappone la regressione le cui ascisse (**fdr$x**) e ordinate (**fdr$yf**) sono state in precedenza salvate nell'oggetto **fdr** generato con la funzione **monoreg()**. Se nella `Console di R` digitate

str(fdr)

potete visualizzare tutte le variabili salvate da **monoreg()** nell'oggetto **fdr**.



Ora copiate e incollate nella `Console di R` questo breve addendum allo script per effettuare l'interpolazione di alcuni valori sulla regressione calcolata e premete ↵ `Invio`.

```
# interpolazione lineare sulla regressione antitòna calcolata in precedenza
reg <- approx(fdr$x, fdr$yf, xout=c(10, 12, 14, 16, 18, 20, 22, 24), ties="ordered") #
interpola i valori specificati
round(cbind(reg$x, reg$y), digits=1) # mostra i risultati dell'interpolazione
#
```

Le percentuali (%) di soggetti sovrappeso corrispondenti al `10, 12, 14, 16, 8, 100, 20, 22, 24` % di obesi calcolate mediante interpolazione lineare sui dati della regressione antitòna (prima riga di codice) sono mostrate (seconda riga).

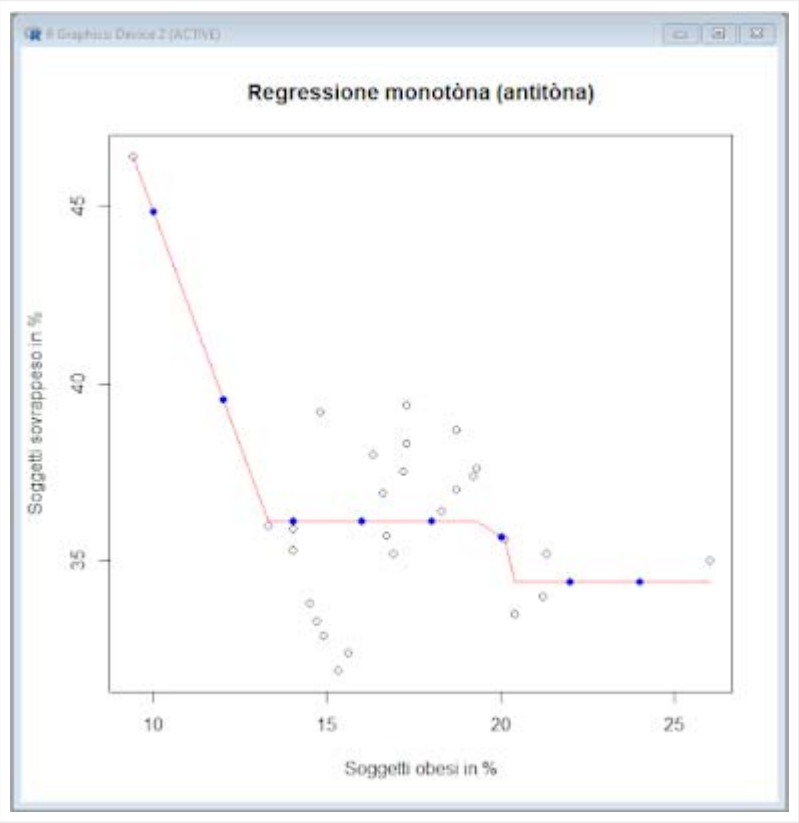
```
> round(cbind(reg$x, reg$y), digits=1) # mostra i dati e i valori interpolati
     [,1] [,2]
[1,]   10 44.8
[2,]   12 39.6
[3,]   14 36.1
```

```
[4,]   16 36.1
[5,]   18 36.1
[6,]   20 35.7
[7,]   22 34.4
[8,]   24 34.4
```

Se copiate e incollate nella `Console di R` questa riga di codice e premete `↵ Invio`

```
# riporta sul grafico i punti interpolati
#
points(reg$x, reg$y, pch=16, col="blue")
#
```

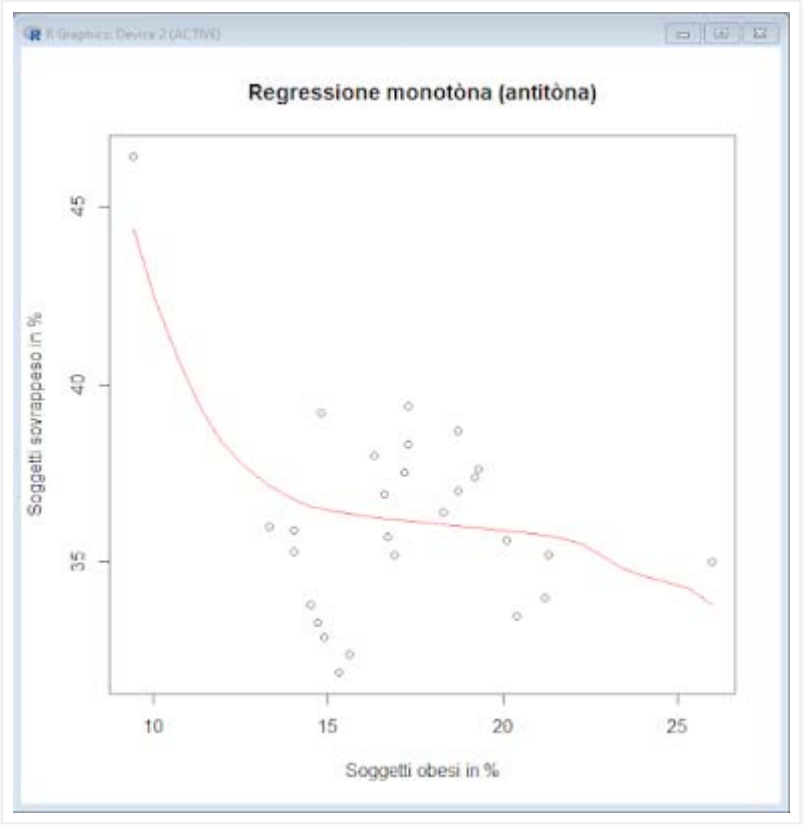
potete riportare sul grafico i risultati dell'interpolazione e confermare la correttezza del calcolo.



Vediamo ora in alternativa la regressione calcolata con il metodo previsto nel pacchetto **monreg**, copiate e incollate nella `Console di R` questo script e premete `↵ Invio`.

```
# REGRESSIONE MONOTÒNA (ANTITÒNA) con il pacchetto monreg
#
library(monreg) # carica il pacchetto per la regressione
bmi <- read.table("c:/Rdati/bmi.csv", header=TRUE, sep=";", row.names="Nazione") #
importa i dati
windows() # apre e inizializza una nuova finestra grafica
#
mon <- monreg(bmi$obeso, bmi$sovrappeso, hd=.5, hr=.5, monotonie="antiton") #
calcola la regressione antitòna
#
plot(bmi$obeso, bmi$sovrappeso, main="Regressione monotòna (antitòna)",
xlab="Soggetti obesi in %", ylab="Soggetti sovrappeso in %") # riporta il grafico con la
distribuzione dei dati
lines(mon$t, mon$estimation, col="red") # sovrappone la regressione
#
```

Lo script è quasi identico al precedente, salvo il fatto che la regressione viene calcolata con la funzione **monreg()** e con l'argomento **monotonie="antiton"** salvando i risultati nell'oggetto **mon** dal quale la funzione **lines()** ricava ascisse (**mon$t**) e ordinate (**mon$estimation**) per tracciare la regressione, che ha un andamento meno spigoloso della precedente.



Questo breve addendum allo script consente di effettuare sulla regressione l'interpolazione lineare di alcuni valori a titolo di esempio, per farlo copiatelo e incollatelo nella `Console di R` e premete ↲ `Invio`.

```
# interpolazione lineare sulla regressione antitòna calcolata in precedenza
reg <- approx(mon$t, mon$estimation, xout=c(10, 12, 14, 16, 18, 20, 22, 24),
ties="ordered") # interpola i valori specificati
round(cbind(reg$x, reg$y), digits=1) # mostra i risultati dell'interpolazione
#
```

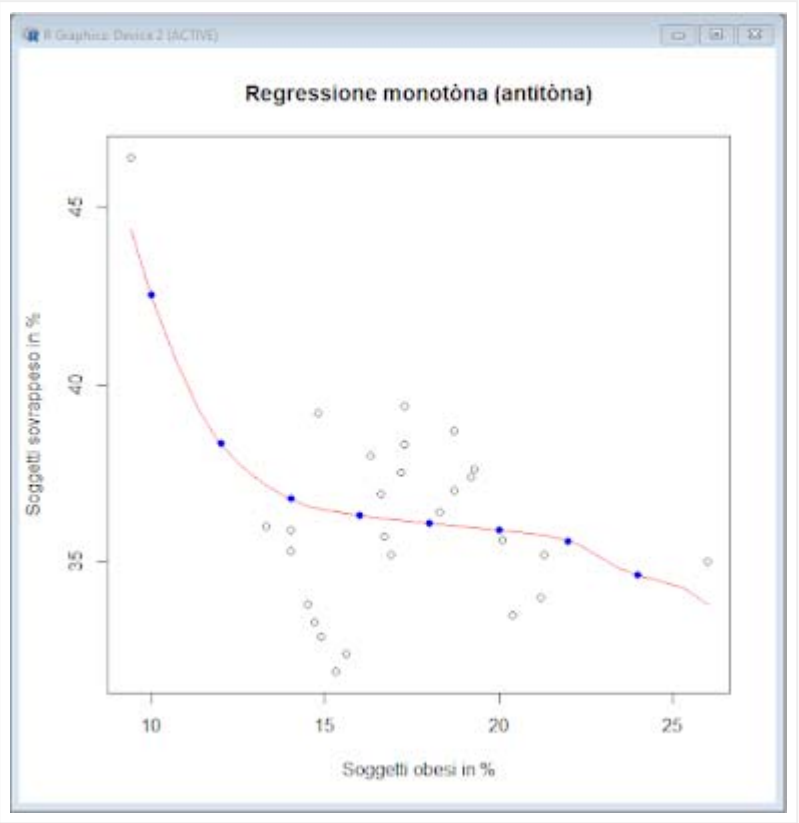
Le percentuali (%) di soggetti sovrappeso corrispondenti al `10, 12, 14, 16, 8, 100, 20, 22, 24` % di obesi calcolate mediante interpolazione lineare sui dati della regressione antitòna sono ora queste.

```
> round(cbind(reg$x, reg$y), digits=1) # mostra i dati e i valori interpolati
     [,1] [,2]
[1,]   10 42.5
[2,]   12 38.3
[3,]   14 36.8
[4,]   16 36.3
[5,]   18 36.1
[6,]   20 35.9
[7,]   22 35.6
[8,]   24 34.6
```

Se si desidera riutilizzare lo script, oltre a tutto il resto anche i valori da interpolare riportati nell'argomento **xout=c()** devono essere ovviamente adattati al bisogno. Anche in questo caso con una semplice riga di codice, identica a quella già vista a conclusione dello script precedente

```
# riporta sul grafico i punti interpolati
#
points(reg$x, reg$y, pch=16, col="blue")
#
```

potete verificare la correttezza del calcolo riportando sul grafico i risultati dell'interpolazione.



---

[1] Vedere il post Regressione con una funzione monotòna (isotòna).

[2] Nella letteratura statistica anglosassone trovate "*antitonic*". Qui "*monotòna*" viene riportato con il significato matematico originario di "*funzione che non ha massimi e non ha minimi*" mentre "*isotòna*" e "*antitòna*" sono gli attributi aggiuntivi di una funzione monotòna quando è rispettivamente una "*funzione crescente o sempre non decrescente*" e quando è una "*funzione decrescente o sempre non crescente*".

[3] Vedere il post Indice di massa corporea (BMI).

[4] Per i dettagli della funzione potete digitare **help(scatterplotMatrix)** nella `Console di R` ma trovate anche una breve spiegazione nel post Grafici di dispersione (grafici xy).

[5] A pagina 14 del *Reference manual* del pacchetto **fdrtool** è riportato: "*If several identical x values are given as input, the corresponding y values and the weights w are automatically merged, and a warning is issued*".
https://cran.r-project.org/web/packages/fdrtool/fdrtool.pdf