Comandi che seguono nella Console di R

- ## Uscire da R !

```
q()
```

- ## Chiedere aiuto ad R

```
help()    # chiede l'help generale
help(q)   # chiede l'help del comando "q"

help.start() # lancia l'help in un browser
```

- ## Esempi e dimostrativi

```
example(ls)  # mostra l'esempio, se disponibile del comando "ls"

example(image) # mostra gli esempi del comando image
example(persp)
example(contour)

demo(graphics)
demo(persp)
demo(image)
```

- ## Assegnazione di oggetti

```
x <- 4
y <- c(2,7,4,1)
x
y
```

- ## Il verso della freccia indica chi viene assegnato a cosa

```
y <- c(2,7,4,1)
y
c(2,7,4,1) -> y
y
```

- ## Assegnazione esplicita, più flessibile

```
assign("y",c(2,7,4,1))
y
for(i in 1:9)  assign( paste("V",i,sep="") , i)
ls()
```

- ## Contenuto e manipolazione del workspace

```
ls()                    # mostra il contenuto del workspace
# il simbolo "#" indica l'inizio di una linea di commento. R
ignorera' quanto segue
rm(i)                   # elimina uno o piu' oggetti, in questo
caso la variabile "i"
```

```
ls(pattern="V")        # mostra gli oggetti che contengono "V"
nel nome
ls(pat="V")            # i nomi dei parametri dei comandi
possono essere abbreviati

save(list=ls(),file="prova.rda")   # salviamo il workspace in un
file chiamato prova.rda
                                   # rda e' l'estensione
standard per indicare un file dati di R
                                   # ma si puo' usare qualsiasi
altra estensione o ometterla
rm(list=ls(pat="V"))   # rimuoviamo tutte le variabili che
contengono "V" nel nome
ls()                   # non e' rimasto molto nel workspace
rm(list=ls())          # cancelliamo tutto il workspace
ls()                   # non c'e' piu' nulla

load("prova.rda")      # ricarichiamo i dati salvati in
precedenza
ls()                   # fiuuuuu

save(V1,V2, file="prova2.rda")  # si possono salvare
selettivamente solo alcuni oggetti
save.image()                    # o salvare tutto il workspace
in un file di default di R
                                # chiamato .RData
load(".RData")                  # ricarichiamo tutto
ls()
```

- ## Operazioni elementari su stringhe

```
paste("A","B",2,"c",sep="*")

paste("A","B",2,"c",sep="")

paste("A","B",2,"c",sep=" * ")
```

- ## Matrici e operazioni algebriche

```
rm(list=ls())
x <- 4
y <- c(2,7,4,1)

x * Y            # R e' un linguaggio case-sensitive. Nel nostro
caso esiste "y" ma non "Y"

x * y            # scalare per vettore

y * y            # y_i * y_i

t(y) %*% y       # vettore riga * vettore colonna = scalare
                 # Il simbolo %*% e' l'operatore prodotto
matriciale
z <- y %*% t(y)  # vettore colonna * vettore riga = matrice


a <- matrix(1:30, 5, 6)  # costruzione di matrici per
riempimento
a
```

```
matrix(0,2,3)            # matrice di 0
matrix(1,2,3)            # matrici di 1
matrix(,2,3)             # riempimento senza assegnazione di
valori (NA)


t(a)                     # "t()" funzione "trasposto di"

x <- matrix(c(1,2,3,4),1,4) # "x" definito come vettore riga
y <- matrix(c(1,2,3,4),4,1) # "y" definito come vettore colonna

x * y                       # messaggio d'errore
x %*% y
y %*% x
x * t(y)
t(x) * y


x <- c(1,2,3,4)
y <- c(2,4,6,8)
v1 <- x + y        # operazioni termine a termine
v2 <- x - y
v3 <- x * y
v4 <- x / y
```

- **Accedere agli elementi dei vettori**

```
-1:3        # successione di valori da -1 a +3
x <- 1:4
x
x <- -3:8
x
seq(-3,6,2)     # successione di valori da -3 a 6 passo 2
seq(-3,-1,.33)  # successione di valori da -3 a -1 passo .33
seq(-3,-1,len=11)  # successione di valori da -3 a -1 di
lunghezza 11


sequenza <- -3:8
A <- matrix(sequenza,2,6)
A
diag(1,3,3)     # crea una matrice diagonale di 1 e dimensioni
3x3
matrix("A",2,3) # crea una matrice di simboli "A"

sequenza[3]   # elemento 3 del vettore sequenza

A[2,3]        # elemento (2,3) della matrice A

A[2,]         # seconda riga della matrice A

A[,6]         # sesta colonna della matrice A
```

- **Ricerca di elementi all'interno di un vettore**

```
x <- -3:8

which(x<2)                  # posizioni degli elementi che
verificano x<2
```

```
which((x >= -1) & (x < 5))  #   ""              ""
""     x >= -1 e x < 5

which((x < -2) | (x > 1))   #   ""              ""
""     x < -2  o x > 1

indici<-which((x >= -1) & (x < 5))  # mettiamo gli indici in un
vettore

x[indici]                           # estraiamo da "x" i valori
che ci interessano
x[-indici]                          # e i loro "complementari"
```

- **Gli oggetti e le tipologie di dati**

```
cmp <- complex(real=1:10,imaginary=-1:9)     # numeri complessi

str <- c("tizio", "caio", "sempronio")       # stringhe

log <- c(TRUE, TRUE, FALSE, FALSE, FALSE)    # booleani

A <- matrix(1, 4, 2)                         # matrici

elle <- list(CPLX = cmp, NOMI = str, BOOL = log,  matrice = A)
# liste
elle

elle2<- list( cmp,  str,  log,   A)
elle2
```

- **Accedere agli elementi di oggetti complessi**

```
elle$BOOL         # il simbolo $ e' cruciale

elle[[3]]         # stesso risultato ma meno intuitivo
elle2[[3]]

elle$CPLX[4]      # accediamo all'elemento 4 del vettore "CPLX"
che si trova nella lista "elle"

elle$C[4]         # al solito, se non ci sono ambiguita'
possiamo abbreviare le notazioni
```

- **Strutture degli oggetti**

```
str(cmp)

str(log)

str(str)

str(A)

str(elle)

str(elle2)
```

- **Le funzioni e i controlli di flusso**

```
somma <- function(a,b) a+b   # "function" e' un nome (di comando
R) riservato

somma(2,3)

somma <- function(a=1,b) {
 if( b>0 )
  a+b
 else
  print("errore")
}

somma(2,3)
somma(b=3)    # si puo' omettere il parametro "a" impostato ad 1
se non specificato
somma(3)      # Errore! Manca un argomento, in questo caso il
secondo, cioe' "b". Si noti la differenza rispetto
all'istruzione precedente.
somma(b=-2)   # la nostra fuzione produce un errore poiche
"b<0"

somma("ciao","come va")  # se non espressamente previsto le
funzioni non effettuano controlli sui parametri

log(9)    # "log" e' un nome ridefinibile, ma non e' opportuno
ridefinire funzioni standard di R
log <- seq(-3,3,len=30)
log[9]
log(3)
log <- 2
log <- function(x) x
log(9)
rm(log)                # rimettiamo tutto a posto


pow <- function(A,n){    # funzione "potenza di matrice"
 tmp <- A
 if(n>1)
  for(i in 2:n)
   tmp <- tmp %*% A
 tmp
}
```

- **Termini riservati**

```
FALSE TRUE Inf NA NaN NULL
break else for functions if in next repeat while
```

- **I cicli: meglio evitare**

Stuttura di un ciclo **for**

for(*variabile a valori* **in** *un insieme* **)** *esegui i comandi*

```
x <- 0
for(i in 1:50000) x <- x + i   # un ciclo e' dispendioso in
termini di tempo
x
```

```
sum(1:50000)                    # se e' possibile usare funzioni
interne ad R
```

- **Cicli indefiniti: while**

  Stuttura di un ciclo **while**

  while(*la condizione risulta verificata* **)** *esegui i comandi*

```
x <- 0
while(x < 10) x <- x+1
x

x <- 0
while(x < 500000) x <- x+i  # anche questi cicli sono
dispendiosi in termini di tempo
x
```

- **Fenomeni qualitativi e ordinamenti**

```
sesso <- c("U","U","U","D","D","D","D")
eta <- c("giovane","giovane","adulto","adulto","anziano",
"giovane","anziano")
str(sesso)
str(eta)

sesso2 <- factor(sesso)  # factor: sinomino di "qualitativo",
"level" : valore della mutabile
str(sesso2)
sesso2

eta2 <- factor(eta)
str(eta2)
eta2

ordered(eta2,levels=c("giovane","adulto","anziano"))  #
ordinamento dei valori della mutabile

eta2 <- factor(eta,levels=c("giovane","adulto","anziano"),
ordered=TRUE)
eta2

eta2 <- ordered(eta,levels=c("giovane", "adulto", "anziano"))
eta2

sesso3 <- c(1,1,1,2,2,2,2)
sesso3
sesso4 <- factor(sesso3)
sesso4
levels(sesso4) <- c("U", "D")
sesso4

eta2
codes(eta2)     # "codes" : valori numerici attribuiti alle
etichette/livelli di una mutabile

eta <- c(15, 16, 45, 55, 75, 15, 70)
eta
str(eta)
```

- **Il dataframe ovvero la matrice dei dati**

```
x <- c(1, 4, 3, 3, 2, 1, 2, 2, 3, 1, 1, 1, 4, 2, 1, 2, 3, 4, 2,
2)
x <- factor(x)
levels(x) <- c("N","C","V","S")
x

y <- c(4, 2, 1, 2, 4, 3, 3, 2, 4, 2, 3, 1, 3, 3, 3, 4, 2, 2, 3,
3)
y <- factor(y)
levels(y) <- c("A","O","S","L")
y
y <- ordered(y)
y

z <- c(0, 1, 3, 4, 1, 1, 0, 2, 3, 0, 1, 0, 1, 4, 3, 0, 2, 2, 4,
4)
z

w <- c(72.5, 54.28, 50.02, 88.88, 62.3, 45.21, 57.5, 78.4,
       75.13, 58, 53.7, 91.29, 74.7, 41.22, 65.2, 63.58,
       48.27, 52.52, 69.5, 85.98)
w

dati <- data.frame(X=x, Y=y, Z=z, W=w)    # istruzione chiave:
creiamo una matrice dati
                                          # specificando i nomi
delle colonne
dati

save(file="dati1.rda", dati)

rm(list=ls())
ls()
load("dati1.rda")
ls()
dati
```

- **Accesso veloce alle colonne di un dataframe**

```
dati$X
dati[1,]
dati[,3]
dati[1,3]

attach(dati)       # "attach" : rende disponibili direttamente
le colonne di un dataframe
ls()
X
Y
detach(dati)       # "detach" : ripristina il comportamento
abituale
X
dati$X

attach(dati)
ls()
W
```

```
W <- 3 * Z        # dopo un "attach" ogni modifica delle colonne
di un dataframe
                  # produce una copia nel workspace di tale
colonna mentre la
                  # colonna originale del dataframe rimane
invariata
W             # copia di W nel workspace
ls()
dati$W        # colonna originale in "dati"
detach(dati)
W
dati$W
rm(W)
rm(list=ls())
```

- ## Creazione di tabelle a due vie

```
load("dati1.rda")
attach(dati)
table(X,Z)   # crea una tabella a doppia entrata
detach()

matrix(1:6,2,3)

VX <- c("basso","medio","alto")
VY <- c("corto", "lungo")
matrix(1:6,2,3, dimnames=list(VY,VX))
```

- ## Scambio di dati con altri software

```
load("dati1.rda")
write.table(dati, file = "dati.txt", sep = "\t",col.names = NA)
write.table(dati, file = "dati-excel.txt", sep = "\t",col.names
= NA,dec=",")   # la versione italian di excel usa la "," come
separatore di decimali

read.delim("dati-excel.txt",dec=",")              # per re-
importare un file generato da Excel
read.delim("dati-excel.txt",dec=",",row.names=1)   # row.names
e' utile per specificare dove si trovano le etichette delle
righe

?read.table    # per un help completo su come importare file di
testo
?write.table   # su come esportare file di testo

library(foreign) # input/output di dati da SAS, SPSS, Stata e
altri
library(xtable)  # output in formato TeX e HTML di dati e
tabelle
```

- ## Salvare l'output dei comandi di R

```
sink("R-out.txt")     # tutto l'output di R verra' registrato
nel file R-out.txt

load("dati1.rda")
dati
str(dati)
```

```
sink()                # il file viene chiuso
```

- **Salvare la lista dei comandi**

```
savehistory()
loadhistory()
```

- **Utilizzare script esterni**

```
source("script.R")  # in un semplice file di testo si possono
elencare dei comandi e chiedere ad R di eseguirli in sequenza
```

- **I pacchetti di R**

```
library()           # elenco delle librerie istallate
library(eda)        # carica il pacchetto "eda"
library(help=eda)   # sommario dei principali comandi del
pacchetto "eda"
```