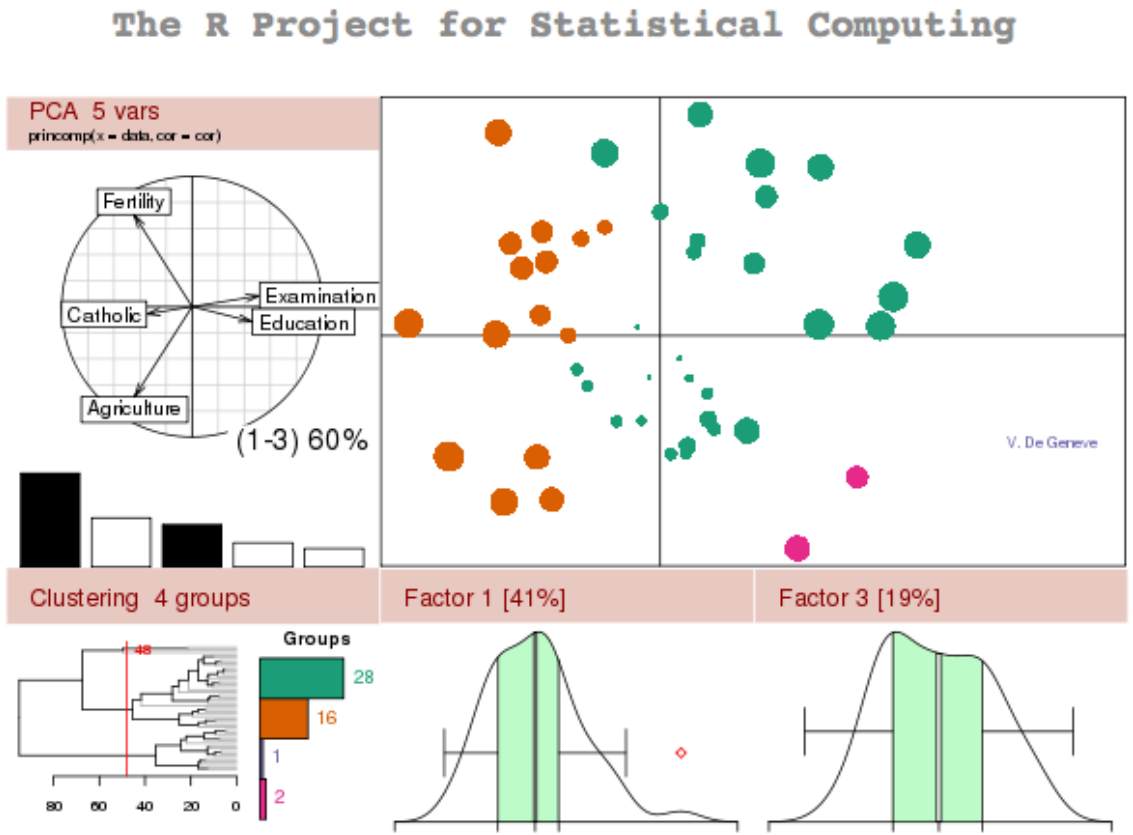
LABORATORIO 1 - R Intro

STATISTICA E LABORATORIO (CDL in INTERNET OF THINGS, BIG DATA, MACHINE LEARNING)

Anno Accademico 2023-2024

R (http://www.r-project.org)



About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

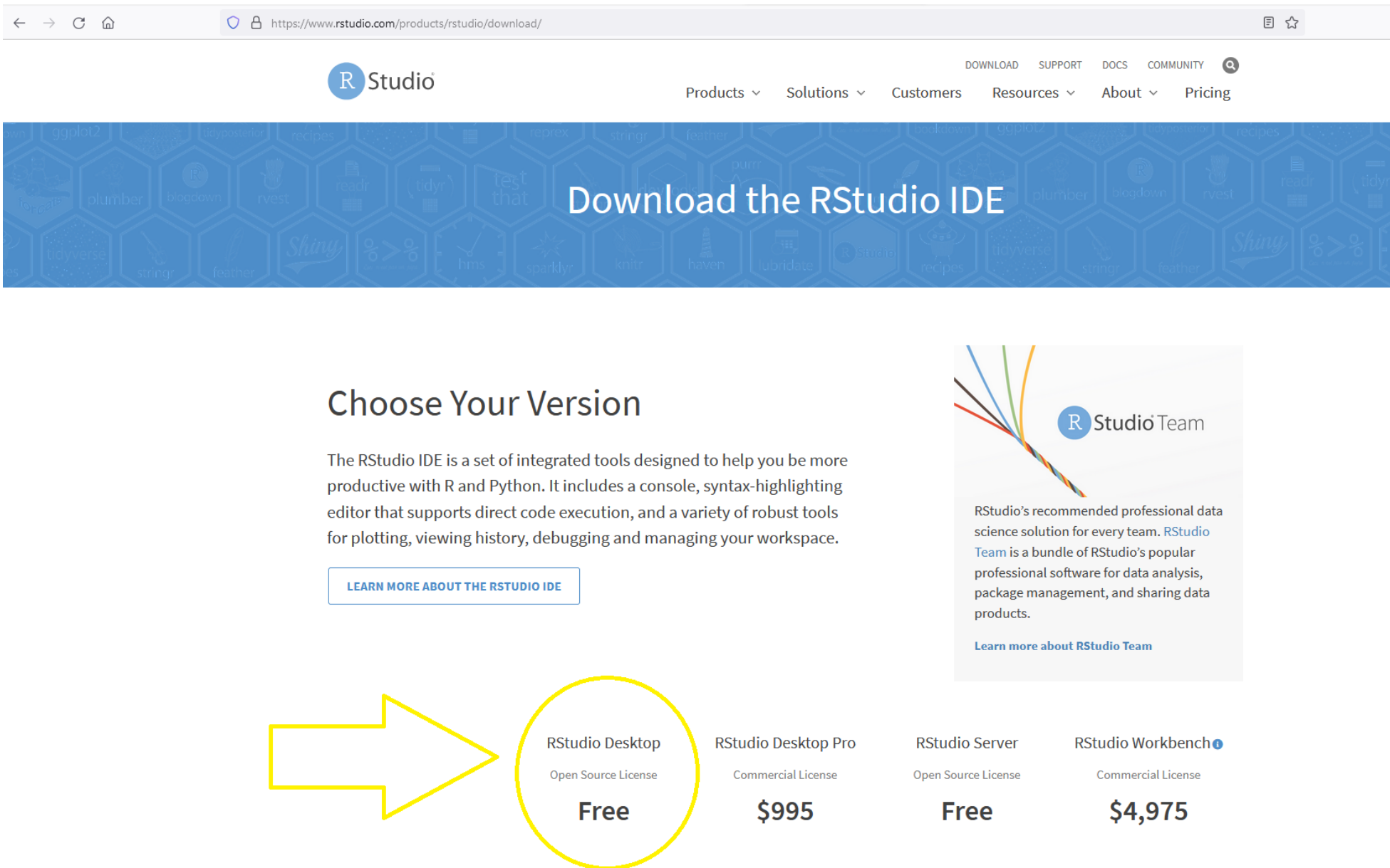
Misc
Bioconductor
Related Projects

Getting Started:

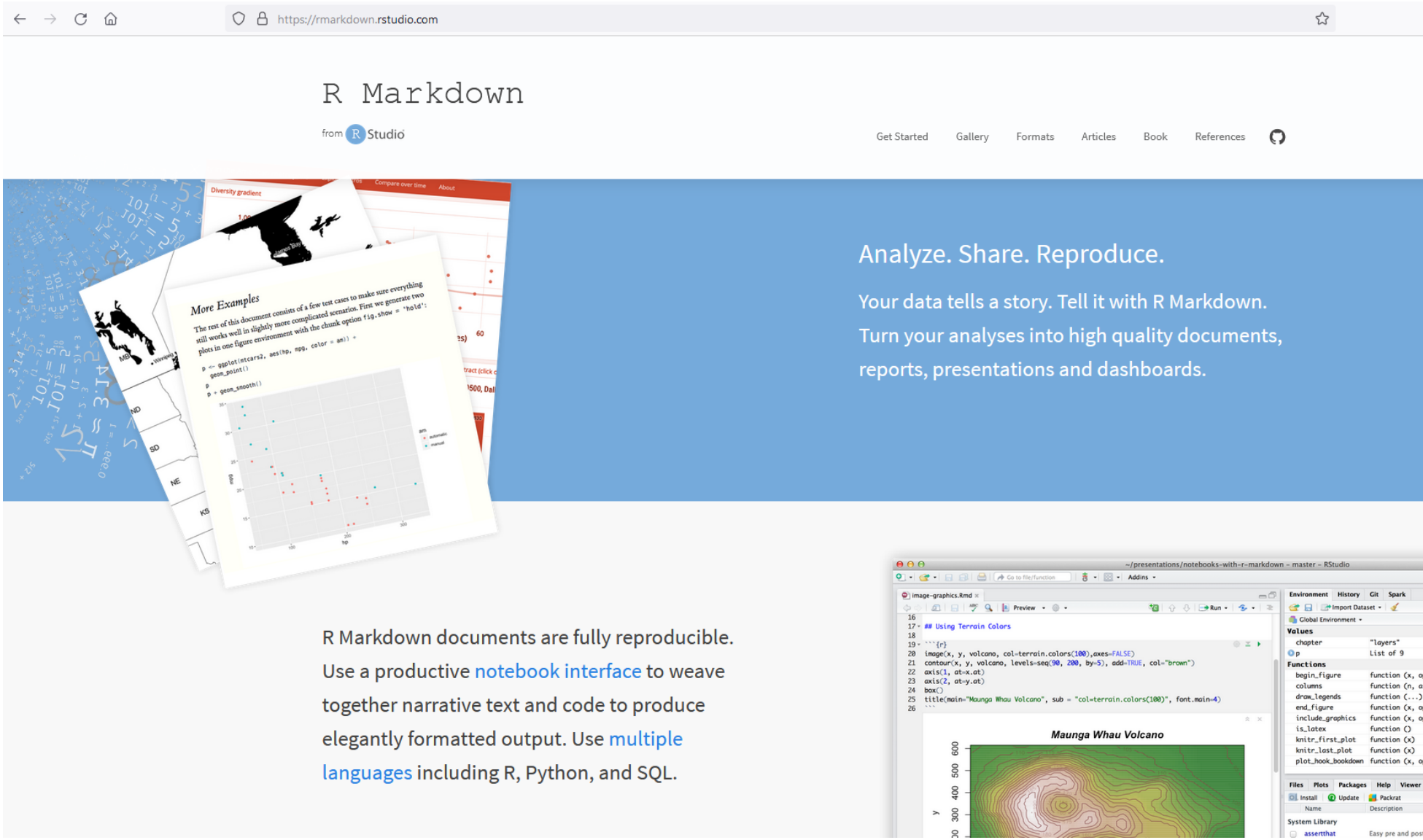
- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News:

R-studio (https://www.rstudio.com)



RMarkdown (https://rmarkdown.rstudio.com/)



RMarkdown

RMarkdown si basa sull'idea di integrare in uno stesso documento codice eseguibile e testo.

RMarkdown utilizza knitr per processare la parte di codice, creare l'output ed includerli in un file che viene poi trasformato in un documento (ad esempio html, pdf o word).

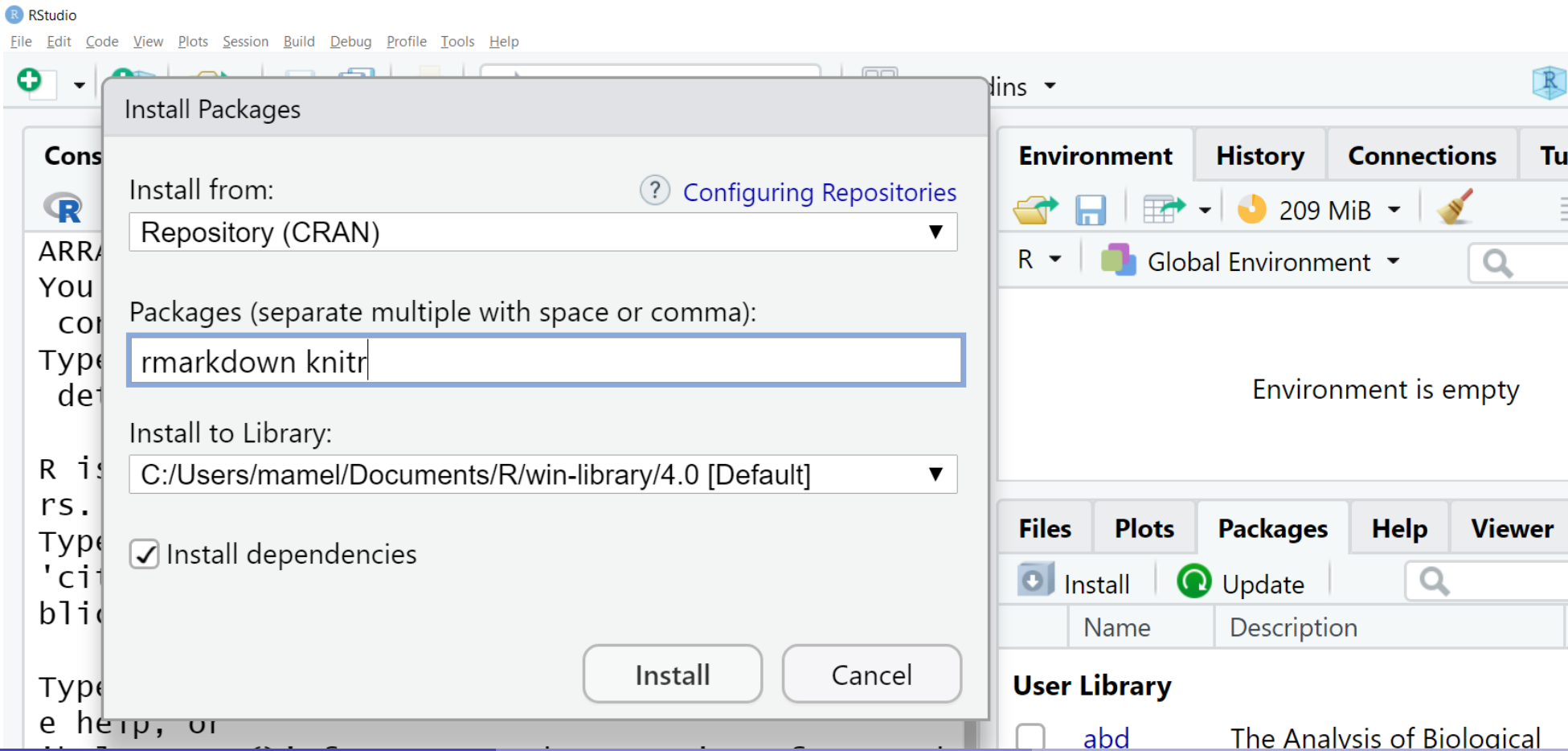
Il linguaggio Markdown è un linguaggio di formattazione per testi estremamente semplice.

Installare Rmarkdown

Per usare RMarkdown è necessario installare la corrispondente libreria.

install.packages("rmarkdown")

install.packages("knitr")



Pacchetti e librerie di R

```
# install.packages("package")  installa un pacchetto

# library("package")  carica un pacchetto

# update.packages()  aggiorna i pacchetti
```

Anatomia di RMarkdown

La prima parte di un documento RMarkdown è una sezione che descrive i metadati del documento: titolo, autore, data di creazione, abstract.

```
---
title: "Untitled"
author: "        "
date: "06/10/21"
output: html_document
---
```

Il resto del documento integra delle parti testuali, scritte in Markdown, con dei chunk di codice R.

Un chunk è un pezzo di codice

```
```{r}
1+2+3
2+3*4
3/2+1
2+(3*4)
(2 + 3) * 4
```
```

Utilizzare R

```
# q(), oppure usa il menu' grafico, per uscire da R.

# savehistory("history.r")

# loadhistory("history.r")

# source("command.r")

# save.image("myfile.Rdata")

# load("myfile.RData")

# R non utilizza specificazioni del percorso file del tipo
# C:\mydocuments\myfile.txt
# Questo perche' R utilizza "\" come carattere di "escape".
# Bisogna quindi usare C:/mydocuments/myfile.txt
# considerando il "back slash"
```

```
# getwd()
# stampa la directory di lavoro corrente

# setwd("c:/docs/mydirectory")
# cambia la directory di lavoro in mydirectory

# ls()
# stampa la lista di oggetti presenti nella directory
# di lavoro corrente

# rm()
# rimuove oggetti presenti nella directory
# di lavoro corrente

# rm(list = ls())
# rimuove tutti gli oggetti presenti nella directory di lavoro
```

R come calcolatrice

```
# + and - somma e sottrazione

# * and / moltiplicazione e divisione

# ^ esponente

# %% operatore modulo

# %\% divisione intera

# print() # stampa contenuto oggetti

# log() # logaritmo

# exp() # funzione esponenziale

# sqrt() # radice quadrata
```

```
# abs() # valore assoluto

# sin() # funzione seno

# cos() # funzione coseno

# tan() # funzione tangente

# asin() # funzione arcoseno

# factorial() # fattoriale

# choose() coefficiente binomiale

# sign() funzione segno (negativo, nullo o positivo)

# round() arrotondamento alla cifra decimale specificata.
```

```r
1+2+3
```

```
## [1] 6
```

```r
2+3*4
```

```
## [1] 14
```

```r
3/2+1
```

```
## [1] 2.5
```

```r
2+(3*4)
```

```
## [1] 14
```

```r
(2 + 3) * 4
```

```
## [1] 20
```

```r
4*3^3
```

```
## [1] 108
```

```r
27^(1/3)
```

```
## [1] 3
```

```r
2/0 # il risultato è infinito (positivo)
```

```
## [1] Inf
```

```r
0/0 # il risultato non è un numero, NaN (Not a Number)
```

```
## [1] NaN
```

```r
23%%3
```

```
## [1] 2
```

```
23%/%3
```

```
## [1] 7
```

```
sqrt(2)
```

```
## [1] 1.414214
```

```
sin(3.14159)
```

```
## [1] 2.65359e-06
```

```
sin(pi)
```

```
## [1] 1.224606e-16
```

Operatori logici

```
# < minore

# <= minore o uguale

# > maggiore

# >= maggiore o uguale

# == uguale

# != diverso

# & operatore and

# | operatore or

# xor disgiunzione esclusiva
```

```r
1 == 1
```

```
## [1] TRUE
```

```r
1 == 2
```

```
## [1] FALSE
```

```r
1 != 2
```

```
## [1] TRUE
```

```r
1 <= 2 & 1 <= 3
```

```
## [1] TRUE
```

```r
1 == 1 | 1 == 2
```

```
## [1] TRUE
```

```r
1 > 1 | 1 > 2 & 3 == 3
```

```
## [1] FALSE
```

```r
1 > 1 & 1 > 2 & 1 > 3
```

```
## [1] FALSE
```

```r
xor(TRUE, TRUE)
```

```
## [1] FALSE
```

```r
xor(TRUE, FALSE)
```

```
## [1] TRUE
```

L'help di R

```
?lm
```

```
## starting httpd help server ... done
```

```
help(lm)
# ??lm ricerca ogni funzione collegata a lm

apropos("mean")
```

```
##   [1] ".colMeans"      ".rowMeans"      "colMeans"       "kmeans"
##   [5] "mean"           "mean.Date"      "mean.default"   "mean.difft
##   [9] "mean.POSIXct"   "mean.POSIXlt"   "rowMeans"       "weighted.me
```

```
# quando il nome della funzione
# non è noto in modo preciso
```

Assegnamento

```
1 + 2 # il risultato viene solo stampato sullo schermo
```

```
## [1] 3
```

```
a <- 1+2 # il risultato viene salvato nell'oggetto a
typeof(a)
```

```
## [1] "double"
```

```
class(a)
```

```
## [1] "numeric"
```

```
is(a)
```

```
## [1] "numeric" "vector"
```

```r
str(a)
```

```
##  num 3
```

```r
x <- sqrt(2)    #
x    # per stampare il contenuto di x
```

```
## [1] 1.414214
```

```r
typeof(x)
```

```
## [1] "double"
```

```r
class(x)
```

```
## [1] "numeric"
```

```r
is(x)
```

```
## [1] "numeric" "vector"
```

```
str(x)
```

```
##  num 1.41
```

```
b <-"hello"
typeof(b)
```

```
## [1] "character"
```

```
class(b)
```

```
## [1] "character"
```

```
is(b)
```

```
## [1] "character"              "vector"                 "data.frameRowLab
## [4] "SuperClassMethod"
```

```
str(b)
```

```
##  chr "hello"
```

```
x^3
```

```
## [1] 2.828427
```

```
y <- x^3
y
```

```
## [1] 2.828427
```

```
x <- pi # un nuovo assegnamento cancella quello precedente
x
```

```
## [1] 3.141593
```

```
is(x)
```

```
## [1] "numeric" "vector"
```

```
# b <-"hello"
# typeof(b)
# class(b)
# is(b)
```

Vettori

```
Vector1 <- c(1,2,3,4,5,6,7,8,9,10) # vettore numerico
Vector1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
x <- c(2,3,5,7)
x
```

```
## [1] 2 3 5 7
```

```
x <- c(x,11)
x
```

```
## [1]  2  3  5  7 11
```

```
Vector2 <- c("a","b","c","d") # vettore di caratteri
Vector2
```

```
## [1] "a" "b" "c" "d"
```

```r
Vector3 <- c("1","2","3","4")  # vettore di caratteri
#(i numeri vengono interpretati come caratteri)
Vector3
```

```
## [1] "1" "2" "3" "4"
```

```r
x = c(TRUE, FALSE, TRUE, FALSE)
y = !x
x
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

```r
y
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```r
x & y
```

```
## [1] FALSE FALSE FALSE FALSE
```

```r
x | y
```

```
## [1] TRUE TRUE TRUE TRUE
```

```r
Vector4 <- c(Vector2 , Vector3 , Vector2 , Vector2 , Vector2)
Vector4
```

```
##  [1] "a" "b" "c" "d" "1" "2" "3" "4" "a" "b" "c" "d" "a" "b" "c"
## [20] "d"
```

```r
xx <- 1:10
xx
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
5:-5
```

```
##  [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

```r
seq(from=0,to=10) # si possono omettere i nomi degli argomenti
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

```r
# e il passo  della sequenza,
# se non è indicato, corrisponde a 1 (valore di default)
```

```r
seq(0,10)
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

```r
seq(0,10,by=2) # per definire il passo della sequenza
```

```
## [1]  0  2  4  6  8 10
```

```r
seq(0,10,length.out=25) # per definire la lunghezza della sequenza
```

```
##  [1]  0.0000000  0.4166667  0.8333333  1.2500000  1.6666667  2.08
##  [7]  2.5000000  2.9166667  3.3333333  3.7500000  4.1666667  4.58
## [13]  5.0000000  5.4166667  5.8333333  6.2500000  6.6666667  7.08
## [19]  7.5000000  7.9166667  8.3333333  8.7500000  9.1666667  9.58
## [25] 10.0000000
```

```r
rep(0,time=10) # ripete l'elemento 10 volte
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```r
rep("Hello",3) # ripete l'elemento 3 volte
```

```
## [1] "Hello" "Hello" "Hello"
```

```r
rep(Vector1,2) # il vettore viene ripetuto 2 volte
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10  1  2  3  4  5  6  7  8  9 10
```

```r
rep(Vector2,each=2)
```

```
## [1] "a" "a" "b" "b" "c" "c" "d" "d"
```

```r
# ogni elemento del vettore
# viene ripetuto 2 volte
```

Operazioni con vettori

```
# vengono applicate elemento per elemento

# sum() somma gli elementi del vettore

# prod() prodotto degli elementi del vettore

# min() minimo degli elementi del vettore

# max() massimo degli elementi del vettore

# mean() media aritmetica

# median() mediana

# range() campo di variazione (minimo e massimo)

# var() varianza (divisione per n-1)
```

```r
# sd() deviazione standard

# cov() covarianza (due argomenti, ad esempio cov(x,y))

# cor() coefficiente di correlazione
# (due argomenti, ad esempio cor(x,y))

# sort() ordinamento degli elementi
# (come default decreasing = FALSE)

# order() indice degli elementi ordinati con ordinamento crescente

# length() lunghezza del vettore

# summary() fornisce opportune sintesi statistiche

# which() fornisce l'indice dell'elemento compatibile
# con una affermazione logica
```

```r
# which.min() fornisce l'indice del minimo

# which.max() fornisce l'indice del massimo

# unique() fornisce un vettore senza elementi ripetuti

# round() arrotonda i valori fino alla
# cifra decimale indicata (il valore di default è 0)
```

```r
x <- 0:10
x+1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11
```

```r
y <- -5:5
abs(y)
```

```
##  [1] 5 4 3 2 1 0 1 2 3 4 5
```

```r
x+y
```

```
##  [1] -5 -3 -1  1  3  5  7  9 11 13 15
```

```r
x*y
```

```
##  [1]  0 -4 -6 -6 -4  0  6 14 24 36 50
```

```r
y <- 0:8
```

```
x+y
```

```
## Warning in x + y: la lunghezza più lunga dell'oggetto non è un mu
## lunghezza più corta dell'oggetto
```

```
##  [1]  0  2  4  6  8 10 12 14 16  9 11
```

```
x
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

```
x > 5
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

```
x <- 3:26
max(x)
```

```
## [1] 26
```

```r
min(x)
```

```
## [1] 3
```

```r
sum(x)
```

```
## [1] 348
```

```r
prod(x)
```

```
## [1] 2.016457e+26
```

```r
x <- c(32,18,25:21,40,17)
x
```

```
## [1] 32 18 25 24 23 22 21 40 17
```

```r
sort(x)  # ordine crescente
```

```
## [1] 17 18 21 22 23 24 25 32 40
```

```r
order(x) # posizione degli elementi in ordine crescente
```

```
## [1] 9 2 7 6 5 4 3 1 8
```

```r
Vector1+Vector1
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```r
Vector1/Vector1
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```r
log(Vector1)
```

```
##  [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595
##  [8] 2.0794415 2.1972246 2.3025851
```

```r
round(log(Vector1))
```

```
##  [1] 0 1 1 1 2 2 2 2 2 2
```

```r
round(log(Vector1),digit = 3)
```

```
##  [1] 0.000 0.693 1.099 1.386 1.609 1.792 1.946 2.079 2.197 2.303
```

```r
which(Vector1>=5) # indice degli elementi >=5
```

```
## [1]  5  6  7  8  9 10
```

Selezione di elementi di un vettore

```
Vector6 <- c ("The","Starlab","Fellow","is","a Fool")
Vector6[3]
```

```
## [1] "Fellow"
```

```
Vector6[2:4]
```

```
## [1] "Starlab" "Fellow"  "is"
```

```
Vector6[c(1 ,3 ,4)]
```

```
## [1] "The"    "Fellow" "is"
```

```
Vector6[-2] # tutti gli elementi ad esclusione del secondo
```

```
## [1] "The"    "Fellow" "is"     "a Fool"
```

```
Vector6[5] <- "great"
```

```
Vector6
```

```
## [1] "The"      "Starlab" "Fellow"  "is"      "great"
```

```
xx <- 100:1
xx[7]
```

```
## [1] 94
```

```
xx[c(2,3,5,7,11)]
```

```
## [1] 99 98 96 94 90
```

```
xx[85:91]
```

```
## [1] 16 15 14 13 12 11 10
```

```
xx[91:85]
```

```
## [1] 10 11 12 13 14 15 16
```

```r
xx[c(1:5,8:10)]
```

```
## [1] 100  99  98  97  96  93  92  91
```

```r
xx[c(1,1,1,1,2,2,2,2)]
```

```
## [1] 100 100 100 100  99  99  99  99
```

```r
yy <- xx[c(1,2,4,8,16,32,64)]
yy
```

```
## [1] 100  99  97  93  85  69  37
```

```r
x <- c(32,18,25:21,40,17)
x
```

```
## [1] 32 18 25 24 23 22 21 40 17
```

```r
sort(x)
```

```
## [1] 17 18 21 22 23 24 25 32 40
```

```r
x[order(x)]
```

```
## [1] 17 18 21 22 23 24 25 32 40
```

```r
x <- c(1,2,4,8,16,32)
x
```

```
## [1]  1  2  4  8 16 32
```

```r
x[-4]
```

```
## [1]  1  2  4 16 32
```

```r
x[-c(3,4)]
```

```
## [1]  1  2 16 32
```

```r
x <- -8:7
x<0
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE
```

```
x[x<0]
```

```
## [1] -8 -7 -6 -5 -4 -3 -2 -1
```

```
x[x<0&x<(-2)]
```

```
## [1] -8 -7 -6 -5 -4 -3
```

```
x[x<0|x>5]
```

```
##  [1] -8 -7 -6 -5 -4 -3 -2 -1  6  7
```

```
x[x!=6]
```

```
##  [1] -8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  7
```

```
x[x==6]
```

```
## [1] 6
```

Altre funzioni

```
z<-c(2,3,4,3,NA,NA,6,6,10,11,2,NA,4,3)
max(z) # questa funzione non si può utilizzare in presenza di NA
```

```
## [1] NA
```

```
na.omit(z) # fornisce il vettore senza NA
```

```
##  [1]  2  3  4  3  6  6 10 11  2  4  3
## attr(,"na.action")
## [1]  5  6 12
## attr(,"class")
## [1] "omit"
```

```
max(na.omit(z))
```

```
## [1] 11
```

```r
max(z,na.rm=TRUE) # l'argomento na.rm=TRUE
```

```
## [1] 11
```

```r
# permette la rimozione degli NA

is.na(z)
```

```
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE
```

```r
z.noNA <- subset(z,is.na(z)==FALSE )
z.noNA
```

```
##  [1]  2  3  4  3  6  6 10 11  2  4  3
```

```r
X <- 1:70
Multiple7 <- subset(X,X%%7==0) # operatore modulo
Multiple7
```

```
##  [1]  7 14 21 28 35 42 49 56 63 70
```

Fattori

```
treat <- factor(c("a","b","b","c","a","b"))
treat
```

```
## [1] a b b c a b
## Levels: a b c
```

```
levels(treat) # i livelli del fattore treat
```

```
## [1] "a" "b" "c"
```

```
resp <- c(10,3,7,6,4,5)
resp[treat=="a"]
```

```
## [1] 10  4
```

```
# le osservazioni riferite ai
# soggetti con trattamento "a"
```

```r
sum(resp[treat=="b"])
```

```
## [1] 15
```

```r
# la somma delle osservazioni
# riferite ai soggetti con trattamento "b"

treat1 <- ordered(c("a","b","b","c","a","b"), levels=c("c","b","a"))
# si ha un ordinamento diverso rispetto a quello
# naturale (alfabetico)

treat1
```

```
## [1] a b b c a b
## Levels: c < b < a
```

```r
levels(treat1)
```

```
## [1] "c" "b" "a"
```

```r
x <- c(1:12,25:38,-3:0,13:24)
x1 <- cut(x,c(-5,5,25,40),labels=c("B","M","A"))
# classi (-5,5],(5,25],(25,40] che corrispondono ai livelli B, M, A
x1
```

```
##  [1] B B B B B M M M M M M M M A A A A A A A A A A A A A B B B B
## [39] M M M M
## Levels: B M A
```

```r
levels(x1)
```

```
## [1] "B" "M" "A"
```

```r
f_x1 <- table(x1) # frequenze assolute per ogni livello (classe)
f_x1
```

```
## x1
##  B  M  A
##  9 20 13
```

```r
x<-1:20
y<-factor(rep(0:1,10))
y
```

```
##  [1] 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
## Levels: 0 1
```

```r
tapply(x,y,sum)
```

```
##   0   1
## 100 110
```

```r
# la funzione sum viene applicata ai dati x
# classificati sulla base degli associati livelli
# osservati del fattore y
```

Array e matrici

```
Matrix1 <- matrix(data=1,nrow=3,ncol=3)
# tutti gli elementi pari a 1

Matrix1
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    1    1    1
## [3,]    1    1    1
```

```
dim(Matrix1) # la dimensione della matrice
```

```
## [1] 3 3
```

```
Vector8 <- 1:12
Vector8
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```r
Matrix3 <- matrix(data=Vector8,nrow=4)  # come default byrow=FALSE
Matrix3
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```r
Matrix4 <- matrix(data=Vector8,nrow=4,byrow=TRUE)
# matrice popolata per righe
Matrix4
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```r
Vector9 <- 1:10
Vector9
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
Vector10 <- Vector9^2
Vector10
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```r
Matrix5 <- rbind(Vector9,Vector10)
Matrix5
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## Vector9     1    2    3    4    5    6    7    8    9    10
## Vector10    1    4    9   16   25   36   49   64   81   100
```

```
Matrix6 <- cbind(Vector9,Vector10,Vector9)
Matrix6
```

```
##       Vector9 Vector10 Vector9
##  [1,]       1        1       1
##  [2,]       2        4       2
##  [3,]       3        9       3
##  [4,]       4       16       4
##  [5,]       5       25       5
##  [6,]       6       36       6
##  [7,]       7       49       7
##  [8,]       8       64       8
##  [9,]       9       81       9
## [10,]      10      100      10
```

```
colnames(Matrix6)
```

```
## [1] "Vector9"  "Vector10" "Vector9"
```

```r
rownames(Matrix6) # le righe non hanno nome
```

```
## NULL
```

```r
colnames(Matrix6) <- c("A","B","C")
rownames(Matrix6) <- c("a","b","c","d","e","f","g","h","i","j")
Matrix6
```

```
##     A   B  C
## a   1   1  1
## b   2   4  2
## c   3   9  3
## d   4  16  4
## e   5  25  5
## f   6  36  6
## g   7  49  7
## h   8  64  8
## i   9  81  9
## j  10 100 10
```

```r
Matrix7 <- diag(5) # crea una matrice identica
Matrix7
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

```r
Vector11 <- c(1,2,3,4,5)
Matrix8 <- diag(Vector11) # matrice con Vector11 come
# diagonale principale
```

```
Matrix8
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    2    0    0    0
## [3,]    0    0    3    0    0
## [4,]    0    0    0    4    0
## [5,]    0    0    0    0    5
```

```
diag(Matrix7) # estrae la diagonale della matrice
```

```
## [1] 1 1 1 1 1
```

Operatori per il calcolo matriciale

```
# + - * / operazioni standard scalari o elemento per elemento

# %*% moltiplicazione tra matrici

# t() calcolo della matrice trasposta

# solve() calcolo della matrice inversa

# det() calcolo del determinante

# chol() decomposizione di Cholesky

# eigen() calcolo di autovalori e autovettori

# crossprod() prodotto incrociato

# \%x\% prodotto di Kronecker
```

Selezione di elementi di una matrice

```
Matrix9 <- matrix(1:9,3)
Matrix9
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
Matrix9[1,1] # primo elemento della prima riga
```

```
## [1] 1
```

```
Matrix9[2,3] # terzo elemento della seconda riga
```

```
## [1] 8
```

```
Matrix9[,1] # prima colonna
```

```
## [1] 1 2 3
```

```r
Matrix9[2,] # seconda riga
```

```
## [1] 2 5 8
```

```r
Matrix9[1:2, ] # prima e seconda riga
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
```

```r
Matrix9[Matrix9[,2]>4,]
```

```
##      [,1] [,2] [,3]
## [1,]    2    5    8
## [2,]    3    6    9
```

```r
# tutte le righe che
# come secondo elemento hanno un numero maggiore di 4
```

```r
x <- matrix(1:16,ncol=4)
x
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```r
apply(x,1,sum) # si applica sum alle righe (margine=1)
```

```
## [1] 28 32 36 40
```

```r
y<-apply(x,2,prod) # si applica prod alle colonne (margine=2)
y
```

```
## [1]    24  1680 11880 43680
```

```r
x<-1:5
y<-1:5
outer(x,y)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    2    4    6    8   10
## [3,]    3    6    9   12   15
## [4,]    4    8   12   16   20
## [5,]    5   10   15   20   25
```

```r
# matrice con elementi dati dal prodotto
# (funzione di default)  di tutte le combinazioni
# di elementi di x e y
```

```
outer(x,y,"+")
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    3    4    5    6
## [2,]    3    4    5    6    7
## [3,]    4    5    6    7    8
## [4,]    5    6    7    8    9
## [5,]    6    7    8    9   10
```

```
# matrice con elementi dati dalla somma di tutte le
# combinazioni di elementi di x e y
```

Liste

```
Lst <- list(name="Fred", wife="Mary", no.children=3,
            child.ages=c(4,7,9))
Lst
```

```
## $name
## [1] "Fred"
##
## $wife
## [1] "Mary"
##
## $no.children
## [1] 3
##
## $child.ages
## [1] 4 7 9
```

```r
Lst[[1]] # il primo oggetto della lista
```

```
## [1] "Fred"
```

```r
Lst[[4]] # il quarto oggetto della lista
```

```
## [1] 4 7 9
```

```r
Lst$child.ages # l'oggetto chiamato child.ages
```

```
## [1] 4 7 9
```

```r
Lst$child.ages[2] # il secondo elemento dell'oggetto child.ages
```

```
## [1] 7
```

Data frame

```
data(airquality) # viene caricato il data set
dim(airquality)
```

```
## [1] 153   6
```

```
# dimensione del data frame:
# 153 osservazioni su 6 variabili

# help(airquality)  #  descrizione del data frame
names(airquality)  # nomi delle variabili
```

```
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

```r
# fornisce le prime 6 righe del data frame, in alternativa
# si può usare airquality[1:6,]
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```r
str(airquality)  # struttura del data frame airquality
```

```
## 'data.frame':    153 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
airquality$Ozone
```

```
##   [1]  41  36  12  18  NA  28  23  19   8  NA   7  16  11  14  18
##  [19]  30  11   1  11   4  32  NA  NA  NA  23  45 115  37  NA  NA
##  [37]  NA  29  NA  71  39  NA  NA  23  NA  NA  21  37  20  12  13
##  [55]  NA  NA  NA  NA  NA  NA  NA 135  49  32  NA  64  40  77  97
##  [73]  10  27  NA   7  48  35  61  79  63  16  NA  NA  80 108  20
##  [91]  64  59  39   9  16  78  35  66 122  89 110  NA  NA  44  28
## [109]  59  23  31  44  21   9  NA  45 168  73  NA  76 118  84  85
## [127]  91  47  32  20  23  21  24  44  21  28   9  13  46  18  13
## [145]  23  36   7  14  30  NA  14  18  20
```

```
airquality$Ozone[1:5]
```

```
## [1] 41 36 12 18 NA
```

```
airquality$Ozone + airquality$Wind/airquality$Temp
```

```
##   [1]  41.110448  36.111111  12.170270  18.185484        NA  28.
##   [7]  23.132308  19.233898   8.329508        NA  7.093243  16.
##  [13]  11.139394  14.160294  18.227586  14.179688  34.181818   6.
##  [19]  30.169118  11.156452   1.164407  11.227397   4.159016  32.
##  [25]        NA        NA        NA  23.179104  45.183951 115.
##  [31]  37.097368        NA        NA        NA        NA
##  [37]        NA  29.118293        NA  71.153333  39.132184
##  [43]        NA  23.097561        NA        NA  21.193506  37.
##  [49]  20.141538  12.157534  13.135526        NA        NA
##  [55]        NA        NA        NA        NA        NA
##  [61]        NA 135.048810  49.108235  32.113580        NA  64.
##  [67]  40.131325  77.057955  97.068478  97.061957  85.083146
##  [73]  10.195890  27.183951        NA  7.178750  48.085185  35.
##  [79]  61.075000  79.058621  63.135294  16.093243        NA
##  [85]  80.100000 108.094118  20.104878  52.139535  82.084091  50.
##  [91]  64.089157  59.113580  39.085185   9.170370  16.090244  78.
##  [97]  35.087059  66.052874 122.044944  89.114444 110.088889
## [103]        NA  44.133721  28.140244  65.121250        NA  22.
## [109]  59.079747  23.097368  31.139744  44.132051  21.201299   9.
## [115]        NA
```

```
with(airquality, Ozone + Wind/Temp)    # stesso risultato
```

```
##   [1]  41.110448  36.111111  12.170270  18.185484         NA  28.
##   [7]  23.132308  19.233898   8.329508         NA   7.093243  16.
##  [13]  11.139394  14.160294  18.227586  14.179688  34.181818   6.
##  [19]  30.169118  11.156452   1.164407  11.227397   4.159016  32.
##  [25]         NA         NA         NA  23.179104  45.183951 115.
##  [31]  37.097368         NA         NA         NA         NA
##  [37]         NA  29.118293         NA  71.153333  39.132184
##  [43]         NA  23.097561         NA         NA  21.193506  37.
##  [49]  20.141538  12.157534  13.135526         NA         NA
##  [55]         NA         NA         NA         NA         NA
##  [61]         NA 135.048810  49.108235  32.113580         NA  64.
##  [67]  40.131325  77.057955  97.068478  97.061957  85.083146
##  [73]  10.195890  27.183951         NA   7.178750  48.085185  35.
##  [79]  61.075000  79.058621  63.135294  16.093243         NA
##  [85]  80.100000 108.094118  20.104878  52.139535  82.084091  50.
##  [91]  64.089157  59.113580  39.085185   9.170370  16.090244  78.
##  [97]  35.087059  66.052874 122.044944  89.114444 110.088889
## [103]         NA  44.133721  28.140244  65.121250         NA  22.
## [109]  59.079747  23.097368  31.139744  44.132051  21.201299   9.
## [115]         NA
```

```r
# si estrae un data set che contiene le misurazioni
# di maggio con Solar.R >= 150
subset(airquality, airquality$Month == 5 &
         airquality$Solar.R >= 150)
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 4     18     313 11.5   62     5   4
## 7     23     299  8.6   65     5   7
## 10    NA     194  8.6   69     5  10
## 12    16     256  9.7   69     5  12
## 13    11     290  9.2   66     5  13
## 14    14     274 10.9   68     5  14
## 16    14     334 11.5   64     5  16
## 17    34     307 12.0   66     5  17
## 19    30     322 11.5   68     5  19
## 22    11     320 16.6   73     5  22
## 26    NA     266 14.9   58     5  26
## 29    45     252 14.9   81     5  29
## 30   115     223  5.7   79     5  30
## 31    37     279  7.4   76     5  31
```

```r
x<-1:5
y<-factor(c("a","b","a","a","b"))
z<-matrix(rep(7,15),nrow=5,byrow=F)
es.df<-data.frame(z,uno=x,due=y)
es.df
```

```
##   X1 X2 X3 uno due
## 1  7  7  7   1   a
## 2  7  7  7   2   b
## 3  7  7  7   3   a
## 4  7  7  7   4   a
## 5  7  7  7   5   b
```

```r
z<-matrix(rep(7,15),nrow=5,byrow=F)
z
```

```
##      [,1] [,2] [,3]
## [1,]    7    7    7
## [2,]    7    7    7
## [3,]    7    7    7
## [4,]    7    7    7
## [5,]    7    7    7
```

```r
class(z)
```

```
## [1] "matrix" "array"
```

```r
z_df<- as.data.frame(z) # matrice trasformata in un data frame
```

```
z_df
```

```
##   V1 V2 V3
## 1  7  7  7
## 2  7  7  7
## 3  7  7  7
## 4  7  7  7
## 5  7  7  7
```

```
class(z_df)
```

```
## [1] "data.frame"
```

Caricare e salvare data set

```
# dat <- read.table("file.txt", sep=" ", header=T)
# legge il file di testo con sep=" "
# (il default per read.table) come separatore

# dat <- read.table("file.csv", sep=",", header=T)
# legge il file di testo con sep="," (il default per read.csv)
# come separatore
```

```r
airq <- airquality[!is.na(airquality$Ozone),]

# write.table(airq,file="airq.txt",sep=" ",row.names=F)
# per salvare il data set con lo spazio come
# separatore, row.names=F assicura che i nomi delle
# righe non vengono riportati in airq

# write.table(airq,file="airq.csv",sep=",",row.names=F)
# per salvare il data set con la virgola come separatore

# airq1<-read.table("airq.txt",header=T,sep=" ")
# per caricare il nuovo data frame

# airq2<-read.table("airq.csv",header=T,sep=",")
# per caricare il nuovo data frame
```

```
attach(airquality)
Ozone[1:3]
```

```
## [1] 41 36 12
```

```
detach(airquality)
#Ozone[1:3]
```

Ulteriori funzioni utili

```
# is.numeric()

# is.vector()

# is.factor()

# is.matrix()

# is.data.frame()

# is.character()

# as.numeric() trasforma vettori e matrici
# di altre classi nella classe numeric

# as.character() trasforma vettori e matrici
# di altre classi/di altro tipo in character
```

```
# as.integer() trasforma vettori e
# matrici di altro tipo in integer

# as.factor() trasforma vettori e matrici in fattori

# as.matrix() trasforma un vettore o un data frame
# in una matrice

# as.vector() trasforma una matrice in un vettore

# as.data.frame() trasforma vettori e matrici
# in data frame

# as.list() trasforma vettori e matrici in liste
```

Comandi grafici

```
# plot(x,y) se x e y sono vettori,
# produce lo scatterplot di y rispetto a x

# plot(x) se x e' una serie temporale,
# produce il grafico della serie;
# se x e' un vettore numerico, produce il grafico
# dei valori di x rispetto agli indici corrispondenti
```

```r
# funzione grafica di alto livello
curve(x^3-3*x,-2,2) # opzioni from=-2 e to=2
```

```r
curve(x^3-3*x,-2,2) # opzioni from=-2 e to=2
curve(x^2-2,add=T,col="red")
```

```r
# il colore selezionato è il rosso
# funzione grafica di basso livello, poiché con l'opzione add=T
# si aggiunge una curva al grafico esistente
```

```r
# lo stesso risultato si ottiene con
x <- seq(-2,2,0.01)
plot(x,x^3-3*x,type='l')
# opzione type='l' per
# rappresentare una linea continua invece dei punti
lines(x,x^2-2,col="red") # il colore selezionato è il rosso
```

Funzioni

```
# Indice di massa corporea sulla base di peso (chilogrammi)
# e altezza (metri)
bmi<-function(weight,height)
{
  x<- weight/height^2
  return(x)
}

bmi(75,1.7)
```

```
## [1] 25.95156
```

```
x<-c(60,65,75,80,90)
y<-c(1.5,1.6,1.7,1.8,1.9)
bmi(x,y)
```

```
## [1] 26.66667 25.39062 25.95156 24.69136 24.93075
```

Funzioni

```
bmi(1.7,y)
```

```
## [1] 0.7555556 0.6640625 0.5882353 0.5246914 0.4709141
```

```
# Indice di massa corporea con altezza in metri o centimetri
bmi1<-function(weight,height,cm=F)
{
  if(cm==T) height<-height/100
  weight/height ^2
}

bmi1(75,1.7)
```

```
## [1] 25.95156
```

```
bmi1(75,170,cm=T)
```

```
## [1] 25.95156
```

```r
x<-seq(50,100,by=0.1)
y<-seq(1.5,1.95, by=0.01)
index<-outer(x,y,bmi)
# rappresentazione grafica tridimensionale
persp(x,y,index,xlab="weight",ylab="height",zlab="bmi",
ticktype ="detailed",col="blue", border="blue")
```

```r
# rotazione della figura
persp(x,y,index,xlab="weight",ylab="height",zlab="bmi",theta=-90,
      phi=30, ticktype ="detailed",border="blue",col="blue")
```

```
contour(x,y,index,xlab="weight",ylab="height") # linee di livello
```

```r
contour(x,y,index,xlab="weight",ylab="height",col=2:11,levels=20:30)
# linee di livello da 20 a 30 con colori diversi
# to add a grid
abline(h=seq(1.5,1.9,by=0.05),lty=2,col="grey")
# funzione grafica di secondo livello
abline(v=seq(50,100,by=5),lty=2,col="grey")
```

```r
# i primi n termini di una serie geometrica
# con valore iniziale a0 e ragione r
geom<-function(n,r,a0)
{
  ser <- numeric(n)
  ser[1] <- a0

  for (i in 2:n)
    ser[i] <- ser[i-1]*r
  return(ser)
}

geom(10,0.5,1)
```

```
##  [1] 1.000000000 0.500000000 0.250000000 0.125000000 0.062500000
##  [7] 0.015625000 0.007812500 0.003906250 0.001953125
```