################################################################################

################################################################################

## ##

## ORIENTAMENTO CONSAPEVOLE - UniBA ##

## ##

## Imparare dai dati: la Statistica come strumento della conoscenza ##

## ##

## ##

## Praticamente Statistica - Laboratorio di software open source ##

## ##

## C. Calculli - Dipartimento di Economia e Finanza ##

## ##

################################################################################

################################################################################

#---------------------------------------------------------------

# -> Entrare in R

# -> Uscire da R

# -> Ri-entrare in R

#---------------------------------------------------------------

# Cominciamo dall'elaborazione minima, ovvero

# dalla possibilità di usare R come calcolatrice

2+2

# Tutti gli oggetti di R possiedono un nome

# L'operatore di assegnazione è

# " <- " o " = "

# (si possono usare entrambi i simboli)

# Primi oggetti R

a

a=2+2

a

b=2\*a

# Si utilizza il simbolo "#" per scrivere commenti all'interno

# dei file script. Tutti i comandi che seguono il simbolo # sulla

# stessa riga non vengono eseguiti

# Tutti gli oggetti creati risiedono nell'area di lavoro (R workspace)

# Salvo l'area di lavoro dando un nome al file (estensione .RData)

save.image("myfile.RData")

# Quali sono gli oggetti memorizzati nell'area di lavoro?

ls()

# Dove stiamo lavorando? In quale cartella stiamo memorizzando l'area

# di lavoro?

getwd()

# Cambio cartella mediante il menù a tendina

# (Session --> Set Working Directory --> Choose Directory),

# da adesso in poi l'area di lavoro verrà memorizzata nella nuova cartella

# Quali comandi ho digitato fino ad ora?

history()

# Salvo la history (anche dal pulsante "Savw history into a file")

savehistory()

# Uscire: 2 diverse possibilità

# 1. chiudiamo dalla X rossa in alto a destra: salva area di lavoro? SI/NO

# 2. q()

# Ri-entrare: 2 diverse possibilità

# 1. doppio click sull'icona del file .RData - ripristina l'area di lavoro

# e la cartella

# 2. doppio click sull'icona di R - area di lavoro vuota

#---------------------------------------------------------------

# -> Come ottenere aiuto

#---------------------------------------------------------------

# Esistono diverse fonti di informazioni disponibili su R

# sia all'interno di R stesso che, naturalmente, sul Web

# Funzione help()

# Per informazioni su una funzione di R (ad esempio sum) basta digitare

help(sum)

# oppure

?sum

# Per caratteri speciali o parole riservate bisogna usare ""

# Ad esempio, per info sull'operatore "<" digita

help("<")

# oppure

?"<"

# Ad esempio per info sulla parola riservata "if" digita

help("if")

# oppure

?"if"

# Ad esempio per le operazioni numeriche

help(Arithmetic)

help(sqrt)

help(Trig)

help(Special)

# Funzione help.search()

# La funzione help() serve quando si conosce il nome della funzione

# o della parola riservata da utilizzare, invece la funzione

# help.search() serve ad effettuare ricerche in stile Google nella

# documentazione di R

# Ad esempio per info su grafici 3d

help.search("3d plots")

# oppure

??"3d plots"

# R trova i pacchetti che contengono funzioni per tracciare grafici 3d

??logarithm

# Si ottiene aiuto per interi pacchetti di funzioni digitando

# help(package=nome\_del\_pacchetto)

# Ad esempio ottengo aiuto per il pacchetto "graphics" digitando

help(package=graphics)

# Funzione example()

# Per esempi di uso delle funzioni o delle parole riservate usa

# la funzione example()

example(mean)

example("for")

#---------------------------------------------------------------

# Pacchetti (info su CRAN - Comprehensive R Archive Network):

# installa e carica (menù a tendina)

#---------------------------------------------------------------

#---------------------------------------------------------------

# Oggetti di R

#---------------------------------------------------------------

# Si può risalire al tipo di oggetto tramite il comando mode

# Cominciamo con qualche esempio semplice

# [1] mode: numeric (integer, real o complex)

#---------------------------------------------------------------

a=10

b=15.53

d=3.51+1.2i

# Di seguito otteniamo il tipo di oggetto

class(a)

class(b)

class(d)

# [2] mode: character

#---------------------------------------------------------------

e="Hello"

class(e)

# [3] mode: logical

#---------------------------------------------------------------

g=TRUE

h=F

class(g)

class(h)

#---------------------------------------------------------------

# Vettori

#---------------------------------------------------------------

# I vettori si definiscono mediante la funzione di concatenamento c

# Vettori di numeri

v1=c(5,2,3,1,9)

v1

class(v1)

# Vettori di caratteri

v2=c("a","b","c")

v2

class(v2)

# Vettori di \*oggetti\*

v3=c(a,b,d)

a

b

d

v3

v4=c(g,h)

v4

class(v3)

class(v4)

# Vettore vuoto

v6=c()

v6

# Tutti gli elementi di un vettore sono dello stesso tipo.

# Se si mettono insieme elementi di tipo diverso ...

v1

v2

new.1=c(v1,v2)

new.1

class(new.1)

v3

g

h

new.2=c(v1,v2,v3,g,h)

new.2

class(new.2)

# Il tipo di un vettore può essere modificato

as.character(v1)

as.character(v3)

as.character(v4)

as.numeric(v4)

as.numeric(v2)

# ... cosa succede?

# Gli elementi di un vettore possono essere selezionati

# mediante [posizione elemento]

# Ad esempio per selezionare il secondo elemento di v1

v1[2]

# Per selezionare elementi consecutivi, digita

# [posizione primo elemento : posizione ultimo elemento]

# Ad esempio per selezionare gli elementi dal secondo al quarto di v1

v1

v1[2:4]

# Per selezionare elementi in posizioni non consecutive, digita

# [c("posizione 1","posizione 2","posizione 3")]

# Ad esempio per selezionare gli elementi alle posizioni 3, 2 e 5 di v1

v1

v1[c(3,2,5)]

# Quanti sono gli elementi di v1?

length(v1)

# Semplici operazioni sui vettori

sum(v1)

mean(v1)

prod(v1)

cumsum(v1) # somma cumulata

v1-2

v1/2

# risultato intero della divisione %/%

v1

v1%/%3

# resto della divisione %%

v1

v1%%3

# Quando si effettua un'operazione con due vettori R assume

# che siano della stessa dimensione (length). Se non lo sono

# R ripete automaticamente gli elementi del vettore più corto

# sino a raggiungere la lunghezza del vettore più lungo

# Ad esempio

c(2,4,1)+c(5,1,2,0,0,4)

# corrisponde a

c(2,4,1,2,4,1)+c(5,1,2,0,0,4)

v1\*c(5,2)

# corrisponde a

v1\*c(5,2,5,2,5)

# [4] class: list

#---------------------------------------------------------------

# Serve a mettere insieme elementi di tipo diverso senza

# cambiare tipo. Contenitore di oggetti di tipo diverso

list.1=list(c("Hello","Friends","Hello"),c(30,50,0.5,0.1),c(TRUE,TRUE,FALSE))

list.1

class(list.1)

# Come selezionare gli elementi di una lista

# Soluzione 1: se gli elementi della lista non hanno un nome

# (come nel caso precedente) si può selezionarne la posizione

# mediante [[]]

list.1[[1]]

list.1[[1]][1]

list.1[[1]][2]

list.1[[1]][c(1,3)]

list.1[[2]]

list.1[[2]][4]

list.1[[2]][1:3]

list.1[[2]][c(1,4)]

# Soluzione 2: se gli elementi della lista hanno un nome

# si possono selezionare mediante $

list.2=list(welcome=c("Hello","Friends","Hello"),

mynumbers=c(30,50,0.5,0.1),

myidea=c(TRUE,TRUE,FALSE))

# notate come in R è possibile scrivere comandi su più righe

list.2

list.2[[1]]

list.2[[1]][c(1,3)]

# gli ultimi due comandi equivalgono a

list.2$welcome

list.2$welcome[c(1,3)]

#---------------------------------------------------------------

# Facciamo pulizia!

#---------------------------------------------------------------

# Eliminazione di oggetti e pulizia dell'area di lavoro

# Per eliminare un oggetto dall'area di lavoro si usa la

# funzione rm(nome oggetto).

# Ad esempio, eliminate a digitando

rm(a)

# Verifica

ls()

# ATTENZIONE!!

# Qualunque oggetto eliminato o rinominato non può essere

# recuperato in alcun modo

a # a non c'è più

# consideriamo b

b

new=5+5

b=new

b

# ... ok, ma ... il vecchio b? sovrascritto!

# Per eliminare più oggetti dall'area di lavoro si può

# usare la funzione rm nel modo seguente:

rm(list=c("b","new"))

ls()

# Per eliminare \*tutti\* gli oggetti dall'area di lavoro:

rm(list=ls())

ls()

#---------------------------------------------------------------

# Vettori, matrici e array

#---------------------------------------------------------------

#---------------------------------------------------------------

# Vettori

#---------------------------------------------------------------

v1=c(5,78,3,18)

# Operazioni elementari

length(v1)

sum(v1)

prod(v1)

min(v1)

max(v1)

cumsum(v1) # somma cumulata

cumprod(v1) # prodotto cumulato

# Aggiungere elementi ad un vettore

v2=c(v1,10)

v3=c(v1,v2,v1)

v4=c(v1[1:3],v1[c(1,3,4)])

# Eliminare elementi di un vettore

# Ad esempio, eliminiamo il secondo elemento di v1

v1.r1=v1[-2]

v1.r1

# Ad esempio, eliminiamo il secondo e il quarto elemento di v1

v2.r2=v1[-c(2,4)]

v2.r2

# Cambiamo il valore del secondo elemento

v1[2]=0

v1

# Vettori di caratteri: la funzione paste()

# Serve a concatenare stringhe di testo (eventualmente

# specificando un simbolo separatore)

paste("Hello","World")

paste("Ro","ma",sep="")

paste("Roma","Milano","Torino",sep="-")

paste(c("Roma","Milano","Torino"),"Palermo",sep="-")

#---------------------------------------------------------------

# Funzioni utili a generare vettori

#---------------------------------------------------------------

# Alcune funzioni di R utili per generare particolari tipi di vettori

# L'operatore ":" genera sequenze di valori con elementi separati

# da un'unità

# Ad esempio, sequenza di valori interi da 5 a 8

v2=5:8

v2

# Ad esempio, sequenza di valori interi da 8 a 5

8:5

# La funzione seq() permette di generare sequenze di lunghezza

# specificata (length =) o con elementi che si differenziano per un valore

# specificato (by =)

seq(from=5, to=8, by=1)

seq(from=8, to=5,by=-1)

# 100 valori nell'intervallo [0,1]

seq(from=0, to=1, length=100)

# Valori nell'intervallo [0,1] che si differenziano di 0.2

seq(from=0, to=1, by=0.2)

# La funzione rep() permette di replicare lo stesso valore più volte

# nello stesso vettore

# Replica 2 10 volte

rep(x=2,times=10)

# Replica un insieme di valori

rep(x=c(3,1,6),times=4)

# Replica 4 volte ciascun valore di un insieme

rep(x=c(3,1,6),each=4)

#---------------------------------------------------------------

# Operatori logici:

# 1) negazione: !

# 2) unione: |

# 3) intersezione: &

#---------------------------------------------------------------

# Negazione

!TRUE

# Intersezione

TRUE & FALSE

TRUE & TRUE

# Unione

TRUE | FALSE

c(T,T,F,F)|c(T,F,T,F)

# Verifica del valore logico: operatori any, all e which

# Le funzioni any() e all() stabiliscono se almeno uno o tutti gli elementi

# di un vettore sono TRUE. La funzione which() restituisce la posizione degli

# elementi di un vettore che rispettano una condizione logica specificata

x=c(10,4,2,7,23)

# Valori maggiori di 8

x>8

# Valori uguali a 2

x==2

# Restituiscono una sequenza di TRUE e FALSE:

# TRUE quando la condizione è soddisfatta, FALSE in caso contrario

any(x>8)

any(x>88)

all(x>8)

all(x<8)

which(x>8)

x[which(x>8)]

# Alcuni esempi di utilizzo degli operatori logici

# Prendi gli elementi di x maggiori di 5 e minori di 20

x>5 & x<20

which(x>5 & x<20)

x[which(x>5 & x<20)]

# o analogamente

x[(x>5 & x<20)]

# ... vengono selezionati solo gli elementi corrispondenti a TRUE

# Verifica se ci sono elementi di x maggiori di 5 o minori di 20

x>5 | x<20

# Trova la posizione degli elementi di x uguali a 2

which(x==2)

# Prendi gli elementi di x diversi da 2

which(x!=2)

x[which(x!=2)]

#---------------------------------------------------------------

# NaN, NA e NULL

#---------------------------------------------------------------

0/4

4/0

0/0

# NaN significa \*Not a Number\*

# Nelle basi di dati statistici si incontrano spesso dati mancanti,

# codificati in R con NA (\*Not Available\*)

# NULL vuol dire che il valore non esiste

# Utilizzo di NA

y=c(8,NA,12,0,23)

# Uso dell'opzione na.rm (\*NA remove\*)

sum(y)

sum(y, na.rm=T)

mean(y)

mean(y,na.rm=T)

# Al contrario, i valori NULL sono considerati inesistenti

z=c(10,2,NULL,3)

z

length(z)

sum(z)

# Individuazione dei valori mancanti: funzione is.na()

is.na(y)

!is.na(y)

# Creazione di un vettore senza valori mancanti

y.new=y[!is.na(y)]

y.new

#---------------------------------------------------------------

# Matrici

#---------------------------------------------------------------

# Una matrice è un insieme di numeri caratterizzati da due attributi:

# il numero di riga ed il numero di colonna.

# Per creare una matrice si può utilizzare la funzione matrix()

# Matrice 6x4 con elementi uguali a 0

mat1=matrix(0,nrow=6,ncol=4)

mat1

# Trasformare un vettore in una matrice

z=seq(from=-10, to=10, length=9)

z

mat2=matrix(z,nrow=3,ncol=3)

mat2

# ATTENZIONE: gli elementi vengono assegnati per colonna!!!

# Se vogliamo che siano assegnati per riga, usiamo byrow=T

mat3=matrix(z,nrow=3,ncol=3,byrow=T)

mat3

# Si può ottenere una matrice anche mettendo insieme dei vettori con i

# comandi cbind() e rbind()

x1=seq(1,5,1)

x1

x2=sample(1:10,5) # campione casuale di dimensione 5

x2

x3=sample(20:50,5) # campione casuale di dimensione 5

x3

?sample

# L'oggetto seguente ha x1, x2 e x3 come colonne

matrice = cbind(x1,x2,x3)

matrice

# ... oppure posso fare in modo che x1, x2 ed x3 siano righe di una matrice

rbind(x1,x2,x3)

# Calcolo delle dimensioni di una matrice

dim(mat2)

nrow(mat2)

ncol(mat2)

# Selezione di elementi di una matrice

# Ad esempio, elemento in riga 2 e colonna 3 di mat2

mat2[2,3]

# Elementi nella seconda riga di mat2

mat2[2,] # tutti gli elementi

mat2[2,1:2] # prime due colonne

mat2[2,c(1,3)] # prima e terza colonna

# Elementi nella seconda colonna di mat2

mat2[,2] # tutti gli elementi

mat2[1:2,2] # prime due righe

mat2[c(1,3),2] # prima e terza riga

# Come denominare le righe e le colonne di una matrice

colnames(mat2)=c("First-Column","Second-Column","Third-Column")

# oppure

colnames(mat2)=paste(c("First","Second","Third"), "Column",sep="-")

rownames(mat2)=paste(c("First","Second","Third"), "Row",sep="-")

mat2

# Come modificare gli elementi di una matrice

mat2[1,1]=0

mat2[2,]=c(1,2,3)

mat2[,3]=-mat2[,3]

mat2

# Come eliminare righe o colonne di una matrice

# Elimina la seconda riga

mat2[-2,]

# Elimina la seconda e la prima riga

mat2[-c(2,1),]

# Elimina la seconda colonna

mat2[,-2]

# Per verificare se gli elementi di un vettore sono contenuti

# in un altro vettore si usa %in%

v1=c(5,2,4,6,1,8)

v2=c(4,9,2)

v1%in%v2

v1[v1%in%v2]

# ... o analogamente

v2%in%v1

v2[v2%in%v1]

# Quali elementi di mat2 sono in v1?

mat2%in%v1

# ... il risultato è un vettore: risistemiamolo!

matrix(mat2%in%v1,ncol=3)

# Operazioni algebriche con le matrici

# Prodotto matriciale: %\*%

mat2%\*%mat2

# Moltiplicazione di una matrice per un numero

2\*mat2

# Somma di matrici

mat3=matrix(1:9,nrow=3,ncol=3)

mat2 + mat3

# Trasposta di una matrice

t(mat2)

# Selezione degli elementi sulla diagonale di una matrice quadrata

diag(mat2)

#---------------------------------------------------------------

# Le funzioni apply(), lapply() e sapply()

#---------------------------------------------------------------

# La funzione apply() permette di applicare una funzione specificata a

# tutte le righe o a tutte le colonne di una matrice

# Somma per riga degli elementi di mat2

sum(mat2[1,])

sum(mat2[2,])

sum(mat2[3,])

# oppure analogamente

apply(X=mat2,MARGIN=1,FUN=sum)

# Nota MARGIN=1: la funzione specificata (sum) è applicata alle righe

# MARGIN=2: la funzione specificata (sum) è applicata alle colonne

sum(mat2[,1])

sum(mat2[,2])

sum(mat2[,3])

# oppure analogamente

apply(X=mat2,MARGIN=2,FUN=sum)

# La funzione lapply() (list apply) funziona in modo simile ad apply()

# applicando la funzione specificata a tutti gli elementi di una lista

# e restituendo un'altra lista

list.1=list(c(1,2,3),1:10,c(3,4))

lapply(list.1,sum)

# che corrisponde a

sum(list.1[[1]])

sum(list.1[[2]])

sum(list.1[[3]])

# Nota:

class(lapply(list.1,sum))

# In qualche caso la lista restituita da lapply() può prendere

# la forma più semplice di un vettore o di una matrice

# Questo è ciò che fa la funzione sapply()

sapply(list.1,sum)

class(sapply(list.1,sum))

#---------------------------------------------------------------

# Array

#---------------------------------------------------------------

# Un array è una : matrice con più di due dimensioni

# Esempio

array(1:24,dim=c(4,3,2))

#---------------------------------------------------------------

# Data frames

#---------------------------------------------------------------

# Un data frame è una matrice che può avere colonne di diverso tipo (class)

cont=c("Asia","Africa","America","Europa","Oceania")

pop.rur=c(58.8,60.0,46.0,27,28.4)

confl=c(F,T,T,F,F)

dat.1=data.frame(cont,pop.rur,confl)

dat.1

colnames(dat.1)=c("cont","pop.rur","confl")

rownames(dat.1)=cont

dat.1

dat.1=dat.1[,-1]

dat.1

class(dat.1)

dat.1$pop.rur

dat.1$confl

# Differenze tra un data frame e una matrice

# Costruiamo una matrice affiancando le stesse colonne che compongono il

# data frame mediante il comando cbind()

mat.dat.1=cbind(cont,pop.rur,confl)

mat.dat.1

colnames(mat.dat.1)=c("cont","pop.rur","confl")

rownames(mat.dat.1)=cont

mat.dat.1=mat.dat.1[,-c(1)]

mat.dat.1

typeof(mat.dat.1)

# Gli elementi della matrice sono ricondotti tutti

# allo stesso tipo (character)

#---------------------------------------------------------------

# Come scrivere una funzione

#---------------------------------------------------------------

# Come in tutti i linguaggi di programmazione, anche in R potete

# scrivere le vostre funzioni o modificare le funzioni esistenti

class(help)

class(sum)

# Una funzione è un insieme di istruzioni o comandi che prende

# degli "input", effettua determinati calcoli e restituisce

# degli "output", secondo la seguente struttura generale

# nome.funzione = function(input){

# out=calcoli

# return(out)}

# Ad esempio, una funzione semplice, senza input

hello = function(){

print("HELLO WORLD")

}

# Per eseguire la funzione ...

hello()

# Ad esempio, funzione per calcolare l'area di un cerchio

area.cerchio = function(raggio){

out=(raggio^2)\*pi

return(out)

}

# Calcoliamo l'area di un cerchio con raggio uguale a 3

area.cerchio(raggio=3)

# Ad esempio, funzione per contare il numero di numeri

# dispari in un vettore

# N.B. utilizziamo l'operatore modulo: se n è dispari,

# il resto della divisione per due è maggiore di zero

conta.dispari=function(x){

mod=x%%2

conta=sum(mod)

return(conta)

}

# Prova!

conta.dispari(x=c(1,2,3,7,9))

conta.dispari(x=c(1,2,5,10))

# Adesso prova a scrivere una funzione per contare i numeri pari

# Ricorda: se n è pari, il resto della divisione per due è zero

conta.pari=function(x){

mod=x%%2

conta=sum(mod==0)

return(conta)

}

conta.pari(x=c(1,2,5,10,12,34,25))

# Come utilizzare le funzioni di R: argomenti, ordine e default

# Consideriamo una funzione semplice:

ff=function(x,y,z){

s=(x-y)/z

return(s)

}

# Argomenti: x, y and z

# Ordine: 1° x, 2° y, 3° z

ff(x=6,y=3,z=2) # ok

# equivale a

ff(z=2,x=6,y=3)

# equivale a

ff(6,3,2)

# ma non equivale a

ff(2,6,3)

# Quando si usa una funzione (scritta da se o interna ad R) bisogna

# ricordare che:

# \*: gli argomenti vanno scritti nello stesso ordine della definizione

# \*: si può omettere il nome degli argomenti solo se si è sicuri della

# loro posizione (vedi ff(6,3,2))

# \*: se si usano i nomi degli argomenti, la posizione non è rilevante

# (vedi ff(x=6,y=3,z=2) e ff(z=2,x=6,y=3))

# Come si può modificare una funzione interna ad R?

# Il codice di una funzione può essere visualizzato digitandone il nome

# ... vantaggio dell'open source!

# Per la nostra funzione

ff

# Per qualunque funzione interna ad R, ad esempio

help

# Si può copiare e incollare questo codice e modificarlo per creare

# una nuova funzione

# Strumenti iterativi: for e if

# for serve a replicare uno o più comandi per un numero di volte specificato

# da un indice

# Ad esempio

for(i in 1:10){

print("Hello!")}

# Calcolo la media di tutte le righe di una matrice

a=matrix(1:9,nrow=3)

a

media.righe=rep(0,nrow(a)) # inizializzazione vettore di 0

media.righe

for(i in 1:nrow(a)){

media.righe[i]=mean(a[i,])} # ciascun elemento del vettore media.righe è

# la media dei valori delle righe di a

media.righe

# if serve ad eseguire un comando se è soddisfatta una certa condizione

# Ad esempio, dato un vettore di 7 numeri, se il numero è dispari

# moltiplicalo per 2, se il numero è pari moltiplicalo per 1

x=c(1,2,3,4,5,6,7)

x.new=rep(0,length(x)) # inizializzazione vettore di 0

x.new

for(i in 1:length(x)){

if((x[i]%%2)==0) # verifica se l'i-esimo elemento del vettore è pari

x.new[i]=x[i]

else

x.new[i]=2\*x[i]

}

x.new

#---------------------------------------------------------------

# Lettura di dati da file e prime elaborazioni

#---------------------------------------------------------------

# Carichiamo il file microdati che contiene i risultati dell'indagine

# CATI(Computer Assisted Telephone Interviewing):

# il rilevatore legge le domande del questionario da terminale

# senza facoltà di apportare modifiche né ai quesiti né al questionario

# stesso

mm=read.csv("microdati.csv",header=T, sep=";")

# oppure attraverso la procedura guidata dal pannello Environment

# mm=microdati

class(mm)

# Caratteristiche del data frame creato

dim(mm) # numero unità statistiche e numero variabili

names(mm) # nome variabili

head(mm) # prime 6 osservazioni

# Tipologia dei dati

str(mm)

# E' necessario riqualificare le variabili giudizio.med, ore.set, gua.des

# e anno.nasc per stabilire l'ordine delle categorie

mm$giudizio.med = ordered(mm$giudizio.med,

levels=c("sufficiente", "buono", "distinto", "ottimo"))

mm$ore.set = ordered(mm$ore.set,

levels=c(" 0-10", " 10-15", " 15-20", " 20-25", " 25-30", " 30-35",

" 35-40", " 40-45", " 45-50", "50-Inf"))

mm$gua.des = ordered(mm$gua.des,

levels=c("-500", "501-750", "751-1000", "1001-1250", "1251-1500",

"1501-2000", "2001-2500", "2501-3000", "3001-4000", "4000-"))

mm$anno.nasc = ordered(mm$anno.nasc,

levels=c("-1978", "1979-1985", "1986", "1987", "1988", "1989-1990"))

str(mm)

# Calcolo medio e mediana del voto di maturità

mean(mm$voto.mat)

median(mm$voto.mat)

hist(mm$voto.mat)

abline(v=mean(mm$voto.mat), col="red")

abline(v=median(mm$voto.mat), col="blue")

# Varianza e deviazione standard

var(mm$voto.mat)

sd(mm$voto.mat)

# Funzione summary()

# Per le variabili quantitative restituisce minimo, massimo, media, quartili

# e numero di valori mancanti. Per le variabili qualitative restituisce

# le frequenze delle categorie

summary(mm)

# Comando table()- Distibuzioni di frequenza

# Serve a calcolare le frequenze assolute delle categorie per una o più

# variabili qualitative

table(mm$genere)

t1 = table(mm$maturita)

t1

t2 = table(mm$genere, mm$maturita) # Tabella di contingenza

# o tabella a doppia entrata

t2

# Tabella con frequenze relative (al totale)

?prop.table

prop.table(t1)\*100 # equivale a t1/8384

prop.table(t2)\*100 # equivale a t2/8384

round(prop.table(t1)\*100,2)

round(prop.table(t2)\*100,2)

# Tabella con frequenze condizionate (per riga (1) o per colonna (2))

# Distribuzioni condizionate per riga

prop.table(t2, 1) # le righe sommano ad 1

sum(prop.table(t2, 1)[1,]) # verifica

sum(prop.table(t2, 1)[2,]) # verifica

# Distribuzioni condizionate per colonna

prop.table(t2, 2) # le colonne sommano ad 1

sum(prop.table(t2, 2)[,1]) # verifica

sum(prop.table(t2, 2)[,2]) # verifica

# Per le variabili quantitative bisogna effettuare una divisione in classi

# funzione cut per la definizione delle classi

# Ad esempio per il voto di maturità

table(cut(mm$voto.mat,breaks=c(60,70,80,90,100)))

# classi chiuse a dx di default (ciascuna classe contiene

# l'estremo superiore e non quello inferiore)

# Equivale a

# mm$voto.mat[mm$voto.mat>60 & mm$voto.mat<=70]

# mm$voto.mat[mm$voto.mat>70 & mm$voto.mat<=80]

# ...

# Se voglio includere il 60 aggiungo l'opzione include.lowest=T

# table(cut(mm$voto.mat,breaks=c(60,70,80,90,100),include.lowest = T))

# Possiamo anche applicare ad una tabella a doppia entrata

t3 = table(mm$genere, cut(mm$voto.mat,breaks=c(60,70,80,90,100)))

t3

# opp right=F se le classi sono aperte a dx

t4 = table(mm$genere, cut(mm$voto.mat,breaks=c(60,70,80,90,100),right=F))

t4

prop.table(t3, 1)\*100

prop.table(t3, 2)\*100

# Grafici a barre e a torta (una sola variabile qualitativa)

barplot(table(mm$maturita), col="blue")

bb = barplot(table(mm$giudizio.med), col=3:6,

legend = T, main="Giudizio medio maturità")

text(bb, 0, round(table(mm$giudizio.med), 1), cex=1, pos=3)

my\_colors=c("lightgreen","yellow","salmon","lightblue")

pie(table(mm$giudizio.med), col=my\_colors)

legend(-1.2, 1.1, c("sufficiente","buono","distinto","ottimo"),

cex = 0.5, fill = my\_colors, horiz=TRUE, bty="n")

# Grafici a segmenti (una sola variabile quantitativa con pochi valori)

plot(table(mm$anno.lau))

# Istogramma (una sola variabile quantitativa continua o con molti valori)

hist(mm$voto.mat)

hist(mm$voto.mat, breaks=c(60,70,80,90,100))

# Grafici a barre affiancate o sovrapposte (due variabili qualitative)

barplot(table(mm$maturita,mm$genere),col=2:6)

barplot(table(mm$maturita,mm$genere), beside=T, col=2:6)

my\_colors = c("red","blue","green","yellow","pink")

barplot(table(mm$maturita,mm$genere), beside=T,

ylim=c(0,2000),col=my\_colors)

legend("topright", legend = unique(mm$maturita),

fill = my\_colors, box.lty = 0, cex = 0.5)

barplot(table(mm$genere,mm$maturita),col=c("red","blue"),

ylim=c(0,3000))

legend("top", legend = unique(mm$genere),

fill = c("red","blue"), box.lty = 0, cex = 0.5)

barplot(table(mm$genere,mm$maturita),beside=T,col=c("red","blue"))

legend("top", legend = unique(mm$genere),

fill = c("red","blue"), box.lty = 0, cex = 0.5)

# Diagramma cartesiano o grafico XY (due variabili quantitative)

plot(mm$voto.mat, mm$min.acc)

plot(mm$voto.mat, mm$min.acc, ylim=c(0,2000))

# Boxplot (una variabile quantitativa e una qualitativa)

boxplot(mm$voto.mat~mm$giudizio.med)

boxplot(mm$voto.mat~mm$maturita)

boxplot(mm$min.acc~mm$genere)

boxplot(mm$min.acc~mm$genere, ylim=c(0,2000))

# Grafico XY condizionato (due variabili quantitative e una qualitativa)

# Richiamo un dataset gia' presente in R

data(iris)

class(iris)

head(iris)

summary(iris)

str(iris)

plot(iris$Sepal.Length,iris$Petal.Width,

xlab="Sepal Length",

ylab="Petal Width",

main="IRIS",

pch=16,

col=c(2,4,3)[iris$Species])

# Aggiungo una legenda al grafico

legend("bottomright",c("setosa","versicolor","virginica"),pch=20,col=c(2,4,3))

# Matrice di scatterplot condizionati (più di due variabili quantitative e una

# qualitativa)

pairs(iris[1:4], pch=16, col=c("red","green","blue")[iris$Species])