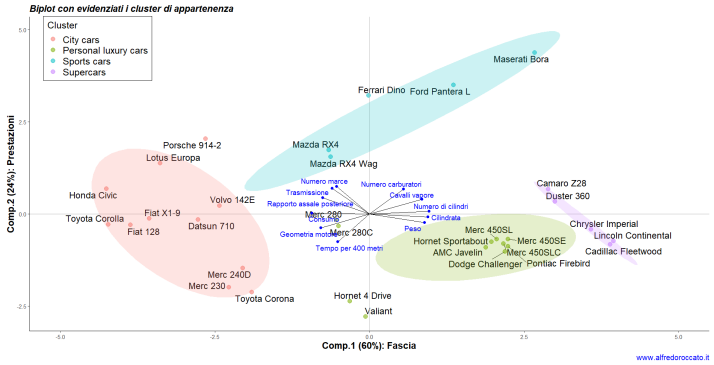
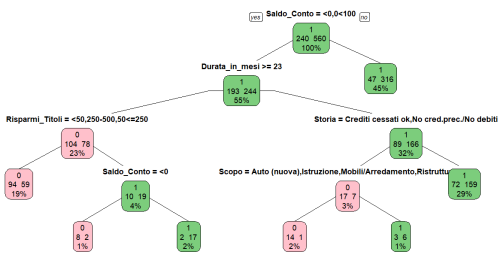
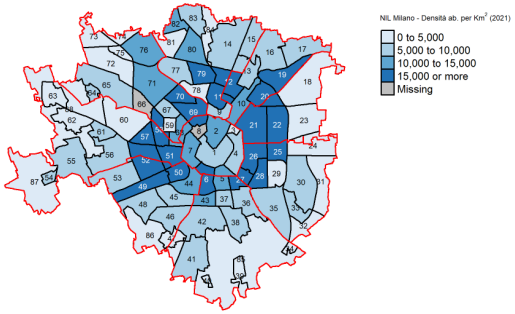
Introduzione al software







Questo libro è stato aggiornato per il software R (Versione 4.2.0 e superiori).

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Che cos'è R

R è un potente **linguaggio di programmazione** per il **calcolo statistico e l'analisi dei dati**.

Fu sviluppato nei primi anni '90 da Ros Ihaka e Robert Gentleman e rilasciato nel 2000.

R è mantenuto dal R Core Team.

Oggi R è diventato il software di riferimento per le analisi statistiche nel mondo accademico e, negli ultimi anni, lo sta diventando anche in quello aziendale.

Questo grazie a una serie di librerie di base associate alla distribuzione e alle librerie aggiuntive (18.785 package al 24 gennaio 2022) e alla disponibilità di interfacce per un facile utilizzo.

Perché si utilizza R

R è **popolare**: si colloca tra i posti alti dell'indice TIOBE (una misura della popolarità dei linguaggi di programmazione): https://tiobe.com/tiobe-index/

R è sviluppato con una licenza **open source**, per cui lo si può installare, utilizzare e distribuire, anche per scopi commerciali, **senza alcun costo**.

R funziona su diverse piattaforme (Windows, macOS e Linux).

R è interpretato: significa che il codice può essere eseguito non appena viene scritto. Ciò significa cicli di sviluppo più rapidi.

È completo: permette di gestire modelli statistici dai più semplici ai più complessi, realizzare rappresentazioni grafiche di alto livello, e molto altro ancora.

È all'avanguardia: difficilmente l'ultima frontiera dell'analisi statistica non ha un'implementazione in R. E se non ce l'ha oggi, molto probabilmente ce l'avrà in un breve tempo.

Ambiente operativo

Il sito ufficiale di R e CRAN

R ha un suo sito ufficiale: https://www.r-project.org;

e un suo archivio centrale, **CRAN** (Comprehensive R Archive Network), gestito da l'R Core Team dove si trova il software e la documentazione: https://cran.r-project.org/.

In questo sito si può:

- effettuare il download della distribuzione R (Windows, macOS e Linux);
- visualizzare e scaricare i package di interesse (organizzati secondo le "task views");
- reperire la documentazione (manuali) di base e avanzata.

Programma

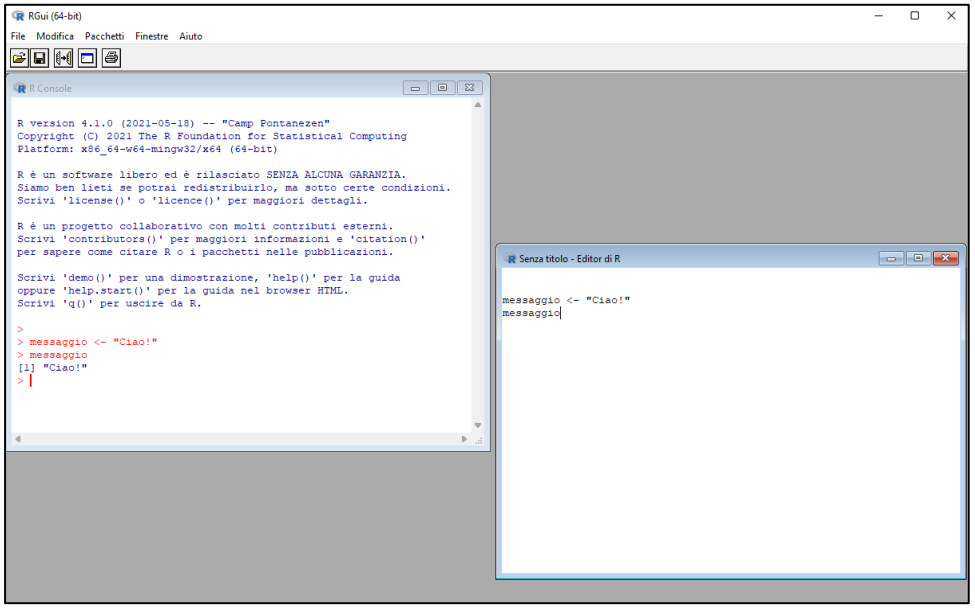
- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Ambiente operativo

La R GUI (Graphics User Interface)

È l'**interfaccia nativa di R**, a riga di comando. È composta da:

- La finestra *Console* dove vengono eseguite le istruzioni e le segnalazioni (messaggi ed errori)
- La finestra *Editor* che permette di creare un file contenente linee di codice R oppure di aprirne uno già esistente ed eseguirlo per intero o per parti.



RStudio

RStudio è una delle **interfacce di R più utilizzate**. È un'interfaccia di tipo IDE (Integrated Development Environment). È anch'essa open-source, scaricabile dal sito: https://www.rstudio.com/products/rstudio/download

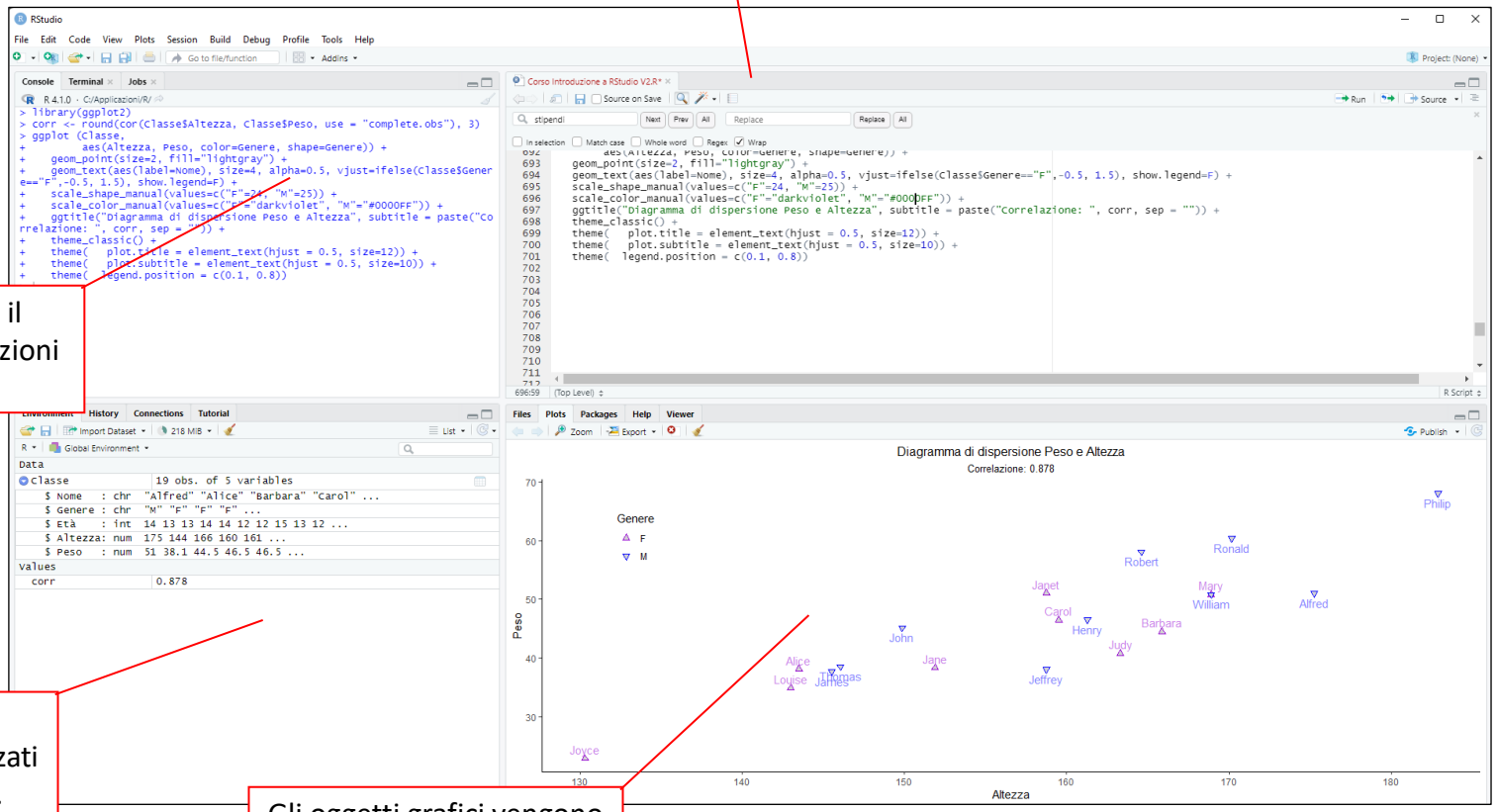
Questa interfaccia consente di:

- scrivere codice R usufruendo delle funzionalità più avanzate (colorazione della sintassi, completamento del codice, rientro intelligente);
- eseguire direttamente il codice;
- visualizzare dati e oggetti nel workspace;
- installare i package;
- consultare l'help online;
- …

Ambiente operativo

- **RStudio**

La finestra degli *Script* contiene il codice R da eseguire



La finestra *Console* contiene il codice eseguito e le segnalazioni (messaggi ed errori)

Gli oggetti presenti in memoria vengono visualizzati nella finestra *Environment*.

Gli oggetti grafici vengono rappresentati nella finestra *Plots*

Installazione dei package

Le capacità di R sono estese attraverso package (pacchetti) creati dagli utenti per particolari esigenze di analisi ed elaborazione.

Il grande numero di pacchetti disponibili per R (18.000+) e la facilità d'installazione e utilizzo, è stato un fattore importante nel guidare l'adozione diffusa del linguaggio R nell'ambito della Data Science.

Per l'installazione di un package bisogna eseguire queste istruzioni:

```
install.packages("nome del package")
library(nome del package)
```

Oppure utilizzando la **R GUI** scegliendo dal menu *Pacchetti* → *Installa pacchetti*..., e da **RStudio** scegliendo dal menu *Tools* → *Install Packages…*

Una breve lista dei package più utilizzati:
https://www.r-bloggers.com/2021/04/15-essential-packages-in-r-for-data-science/

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Importazione dati

- **Lettura file di testo, file con separatori, da URL, file Excel**

```
# File di testo (txt)
Iris <-   read.delim("C:/Temp/Iris_it.txt", header=TRUE, sep =",", dec =".")

# File con separatori (csv)
path <- "C:/Temp/Iris_it.csv"
Iris <- read.csv(path, header=TRUE, sep =",", dec =".")

# Lettura da URL
Class <- read.csv("https://www.ssc.wisc.edu/~hemken/Rworkshops/read/class.csv")

# File Excel (richiede il package readxl)
library(readxl)

path <- "C:/Temp/Iris_it.xlsx"
Iris2 <- read_xlsx(path, sheet= "Iris" , col_names = TRUE)
```

Esportazione dati

- **Scrittura file con separatori, file Excel**

```
# File di testo con separatori
path <- "C:/Temp/Iris_it_2.csv"
write.table(Iris, path, sep=";", dec=",", na="", row.names=F, col.names=T, quote=T)


# File Excel
install.packages("openxlsx")
library(openxlsx)

path <- "C:/Temp/Iris_it_2.xlsx"

# Scrive un foglio di lavoro nel file
write.xlsx(Iris, file = path, sheetName = "Iris_Copia")

# Scrive più fogli di lavoro nel file
fogli <- list("Iris_Copia1" = Iris, "Iris_Copia2" = Iris)
write.xlsx(fogli, file = path)
```

Importazione dati

Lettura tabelle di database (MySQL)

```
library(RMySQL)
db <- dbConnect(RMySQL::MySQL(), dbname = "tweater",
        host = "courses.csrrinzqubik.us-east-1.rds.amazonaws.com",
        port = 3306, user = "student", password = "datacamp")

dbListTables(db)
[1] "comments" "tweats"   "users"

dbGetQuery(db, "DESCRIBE comments;")
     Field       Type Null Key Default Extra
1       id bigint(20)  YES        <NA>
2 tweat_id bigint(20)  YES        <NA>
3  user_id     double  YES        <NA>
4  message       text  YES        <NA>

users <- dbGetQuery(db, "SELECT * FROM users;"); users
  id      name     login
1  1 elisabeth  elismith
2  2      mike     mikey
3  3      thea   teatime
4  4    thomas tomatotom
5  5    oliver olivander
6  6      kate  katebenn
7  7    anjali    lianja
```

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

- Visualizzazione della struttura e del contenuto di una struttura dati

```
# Visualizza il contenuto di un dataframe
Iris
```

```
      SepaloLung  SepaloLarg   PetaloLung  PetaloLarg     Specie
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
...          ...         ...          ...         ...     ...
150          5.9         3.0          5.1         1.8  virginica
```

```
# Visualizza il contenuto di un dataframe (le prime n righe)
head(Class, n=2)
```

```
     Name Sex Age Height Weight
1  Alfred   M  14   69.0  112.5
2   Alice   F  13   56.5   84.0
```

```
# Visualizza la struttura interna di un dataframe
str(Iris)
```

```
'data.frame':       150 obs. of  5 variables:
 $ SepaloLung: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ SepaloLarg: num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ PetaloLung: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ PetaloLarg: num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Specie    : chr  "setosa" "setosa" "setosa" "setosa" ...
```

Strutture

- **Strutture dati presenti in R**
  - Vettori
  - Matrici
  - Liste
  - Fattori
  - Data Frame

Vettori – 1/2

```
v1 <- c(1,2,3);                          # vettore numerico
v2 <- c( "1","2","3" )                   # vettore alfanumerico
v2[2] <- "b"                             # modifica contenuto di un elemento

str(v1); str(v2)
num [1:3] 1 2 3
chr [1:3] "1" "b" "3"

n = 10
v <- seq(from=1, to=n, by=1); v;     # creazione vettore numerico "da-a,con-passo-di"
 [1]  1  2  3  4  5  6  7  8  9 10

v <- rep(0,n); v;                        # vettore di dimensione n con gli stessi valori
 [1] 0 0 0 0 0 0 0 0 0 0

l <- length(v); l;                       # ritorna la dimensione di un vettore
 [1] 10
```

Vettori – 2/2

```
# trasformazione degli elementi da una classe all'altra

x <- 0:6; class(x)
[1] "integer"

x1 <- as.numeric(x); x1
[1] 0 1 2 3 4 5 6

x2 <- as.double(x); x2
[1] 0 1 2 3 4 5 6

x3 <- as.character(x); x3
[1] "0" "1" "2" "3" "4" "5" "6"

x4 <- as.logical(x); x4
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Matrici

```
# Matrice numerica 9 elementi, 3 righe e 3 colonne
m = matrix(1:9, nrow = 3, ncol = 3); m;
```

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

```
str(m)
```

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

```
 # Matrice alfanumerica 6 elementi, 2 righe e 3 colonne
m <- matrix(c("a","b","c","d","e","f"), nrow = 2, ncol = 3); m;
```

```
 [,1] [,2] [,3]
 [1,] "a"  "c"  "e"
 [2,] "b"  "d"  "f"
```

Liste

```
# Sono oggetti che contengono elementi di tipo differente come numeri, stringhe,
# vettori e altre liste.
lst <- list(1, "abc", 1.23, TRUE)
str(lst)
```

```
 $ : num 1
 $ : chr "abc"
 $ : num 1.23
 $ : logi TRUE
```

```
lst
```

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

[[2]]
[1] "abc"

[[3]]
[1] 1.23

[[4]]
[1] TRUE
```

Fattori

```
# Un fattore è una collezione ordinata di valori.
# I diversi valori vengono chiamati livelli.

città <- c("Milano", "Firenze", "Milano")
str(città)
```

```
chr [1:3] "Milano" "Firenze" "Milano"
```

```
città <- factor(città)
str(città)
```

```
Factor w/ 2 levels "Firenze","Milano": 2 1 2
```

Dataframe

```
# A differenza dei vettori e matrici, i dataframe non hanno restizioni sul tipo
# dei dati delle colonne; In poche parole, un dataframe è una lista di vettori di
# uguale lunghezza (come un file csv, o un foglio di lavoro Excel)

nome <- c("Anna", "Sonia", "Marco")
età <- c(25,26,23)
città <- c("Milano", "Firenze", "Milano")

df <- data.frame(nome, età, città); df;
```

```
   nome età   città
1  Anna  25  Milano
2 Sonia  26 Firenze
3 Marco  23  Milano
```

```
str(df)
```

```
'data.frame':      3 obs. of  3 variables:
$ nome : chr  "Anna" "Sonia" "Marco"
$ età  : num  25 26 23
$ città: chr  "Milano" "Firenze" "Milano"
```

Strutture dati presenti in R

Symbol	Meaning	Example
%d	day as a number (0-31)	11324
%a	abbreviated weekday	Mon
%A	unabbreviated weekday	Monday
%m	month (00-12)	00-12
%b	abbreviated month	Jan
%B	unabbreviated month	January
%y	2-digit year	7
%Y	4-digit year	2007

```
# Colonne di tipo Data/Orario[1]

path <- "C:/Temp/Stipendi.csv"
Stipendi <- read.csv(path,header=T,sep =";",dec =".")
str(Stipendi)
```

```
'data.frame':       11 obs. of  6 variables:
 $ Data_di_nascita   : chr  "1970-05-07" "1970-09-15" ...
 $ Data_di_assunzione: chr  "1990-12-02" "1991-10-11" ...
 ...
```

```
Stipendi$Data_di_nascita <- as.Date(Stipendi$Data_di_nascita, "%Y-%m-%d" )
Stipendi$Data_di_assunzione <- as.Date(Stipendi$Data_di_assunzione, "%Y-%m-%d" )
str(Stipendi)
```

```
'data.frame':       11 obs. of  6 variables:
 $ Data_di_nascita   : Date, format: "1970-05-07" "1970-09-15" "1970-04-20" ...
 $ Data_di_assunzione: Date, format: "1990-12-02" "1991-10-11" "1990-06-02" ...
 ...
```

[1] Il giorno 0 parte dal 1 gennaio 1970

Strutture dati presenti in R

```
# Valori mancanti (NA = Not Available; NaN = Not a Number)

x <- NA;        x;        is.na(x);          is.nan(x);
[1] NA
[1] TRUE
[1] FALSE

y <- 0/0;       y;        is.na(y);          is.nan(y);

[1] NaN
[1] TRUE
[1] TRUE

z <- 1/0;       z;        is.na(z);          is.nan(z);

[1] Inf
[1] FALSE
[1] FALSE
```

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Trasformazione

Selezione righe – 1/3

```
Iris1 <- Iris[149:150,]                          # seleziona righe (da-a)

Iris1 <- Iris[149:nrow(Iris), ]                  # seleziona righe (da-fine)

Iris1 <- Iris[c(-1,-2:-147,-148), ]              # elimina righe (lista con segni "-")


lista <- c(-1,-2:-147,-148)                      # lista esterna
lista
```

```
  [1]   -1   -2   -3   -4   -5   -6   -7   -8   -9  -10  -11  -12  -13  -14  -15  -16  -17
  ...
[137] -137 -138 -139 -140 -141 -142 -143 -144 -145 -146 -147 -148
```

```
Iris1 <- Iris[lista, ]
Iris1
```

```
    SepaloLung SepaloLarg PetaloLung PetaloLarg    Specie
149        6.2        3.4        5.4        2.3 virginica
150        5.9        3.0        5.1        1.8 virginica
```

Trasformazione

Selezione righe – 2/3

```
Iris2 <- Iris[ which(Iris$SepaloLung >  5.5 &                    # con la funzione which
                     Iris$PetaloLarg <= 1.5 &
  (Iris$Specie == "setosa" | Iris$Specie == "virginica" )), ] # e operatore | (OR)
```

```
Iris2 <- subset(Iris, SepaloLung >  5.5 &                        # con la funzione subset
                      PetaloLarg <= 1.5 &
         (Specie %in% c( "setosa" , "virginica" )))              # e operatore %in% (IN)

Iris2
```

15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
19	5.7	3.8	1.7	0.3	setosa
120	6.0	2.2	5.0	1.5	virginica
134	6.3	2.8	5.1	1.5	virginica
135	6.1	2.6	5.6	1.4	virginica

Selezione righe – 3/3

```
library(dplyr)                                          # utilizzo funzione select()
Top3SL <- Iris %>%                                      # valori più alti (primi 3)
            select(c("Specie","SepaloLarg")) %>%
            top_n( 3, SepaloLarg); Top3SL
```

```
  Specie SepaloLarg
1 setosa        4.4
2 setosa        4.1
3 setosa        4.2
```

```
Bot3SL <- Iris  %>%                                      # valori più bassi (primi 4)
            select(c("Specie","SepaloLarg")) %>%
            top_n( -4, SepaloLarg); Bot3SL
```

```
  Specie SepaloLarg
1 versicolor        2.0
2 versicolor        2.2
3 versicolor        2.2
4 virginica         2.2
```

Trasformazione

Selezione colonne

```
Iris1 <- Iris[, 1:3]                                    # seleziona le prime 3
Iris1 <- Iris[, c(-4,-5)]                               # elimina le colonne 4 e 5
Iris1 <- Iris[, 1:(ncol(Iris)-2)]; Iris1                # elimina le ultime 2 colonne
```

```
    SepaloLung SepaloLarg PetaloLung
1          5.1        3.5        1.4
...
```

```
colonne <- c("PetaloLung", "PetaloLarg","Specie")   # seleziona colonne per nome
Iris2 <- Iris[,colonne]; Iris2
```

```
colonne <- names(Iris) %in% c("SepaloLung", "SepaloLarg")  # elimina colonne per nome
Iris2 <- Iris[,!colonne]; Iris2                            # con l'operatore ! (NOT)
```

```
    PetaloLung PetaloLarg     Specie
1          1.4        0.2     setosa
2          1.4        0.2     setosa
...
```

Trasformazione

Selezione colonne

```
library(dplyr)
Iris2 <- Iris %>% select(SepaloLung, SepaloLarg, Specie);  # seleziona per nome
Iris2                                                      # con funzione select()
```

```
    PetaloLung PetaloLarg     Specie
1          1.4        0.2     setosa
2          1.4        0.2     setosa
3          1.3        0.2     setosa
...
```

```
Iris2 <- Iris
Iris2$SepaloLung <- Iris2$SepaloLarg <- NULL
Iris2                                                  # elimina colonne con NULL
```

```
PetaloLung PetaloLarg     Specie
 1          1.4        0.2     setosa
 2          1.4        0.2     setosa
 3          1.3        0.2     setosa
 ...
```

Trasformazione

Rinominare colonne

```
Iris1 <- Iris;
names(Iris1)[5] <- "SPECIE"; colnames(Iris1)
```

```
[1] "SepaloLung"  "SepaloLarg"  "PetaloLung"  "PetaloLarg"  "SPECIE"
```

```
Iris1 <- Iris;
names(Iris1)[names(Iris1) == "Specie"] <- "Iris_Specie"; colnames(Iris1)
```

```
[1] "SepaloLung"  "SepaloLarg"  "PetaloLung"  "PetaloLarg"  "Iris_Specie"
```

```
Iris1 <- Iris;
names(Iris1) <- c( "Slu","Sla","Plu","Pla","Spe" ); colnames(Iris1)
```

```
[1] "Slu" "Sla" "Plu" "Pla" "Spe"
```

Trasformazione

Rinominare colonne

```
library(dplyr)
Iris1 <- Iris;
Iris1 <- rename(Iris,
c("PLung"="PetaloLung","PLarg"="PetaloLarg","SLung"="SepaloLung","SLarg"="SepaloLarg"))
colnames(Iris1)
```

```
[1] "SLung"  "SLarg"  "PLung"  "PLarg"  "Specie"
```

```
library(data.table)
Iris1 <- setnames(Iris, old =c("Specie"), new =c("Iris_Specie")); colnames(Iris1)
```

```
[1] "SepaloLung"  "SepaloLarg"  "PetaloLung"  "PetaloLarg"  "Iris_Specie"
```

Trasformazione

Ordinamento righe

```
Class2 <- Class[order( Class$Age, Class$Sex, Class$Weight,
                       decreasing=c(FALSE,FALSE,TRUE), method= "radix" ), ]

Class2
```

```
      Name Sex Age Height Weight
11   Joyce   F  11   51.3   50.5
18  Thomas   M  11   57.5   85.0
7     Jane   F  12   59.8   84.5
13  Louise   F  12   56.3   77.0
16  Robert   M  12   64.8  128.0
10    John   M  12   59.0   99.5
6    James   M  12   57.3   83.0
3  Barbara   F  13   65.3   98.0
2    Alice   F  13   56.5   84.0
9  Jeffrey   M  13   62.5   84.0
4    Carol   F  14   62.8  102.5
12    Judy   F  14   64.3   90.0
1   Alfred   M  14   69.0  112.5
5    Henry   M  14   63.5  102.5
8    Janet   F  15   62.5  112.5
14    Mary   F  15   66.5  112.0
17  Ronald   M  15   67.0  133.0
19 William   M  15   66.5  112.0
15  Philip   M  16   72.0  150.0
```

Trasformazione

Ordinamento colonne

```
Iris1 <- Iris[, c(5, 2, 1, 4, 3) ]; colnames(Iris1)
```

```
[1] "Specie"     "SepaloLarg" "SepaloLung" "PetaloLarg" "PetaloLung"
```

```
Iris1 <- Iris1[, c( "Specie","PetaloLung","PetaloLarg","SepaloLung","SepaloLarg" )]
colnames(Iris1)
```

```
[1] "Specie"     "PetaloLung" "PetaloLarg" "SepaloLung" "SepaloLarg"
```

```
library(dplyr)
Iris2 <- Iris %>% select(Specie, everything())
names(Iris2)
```

```
[1] "Specie"     "SepaloLung" "SepaloLarg" "PetaloLung" "PetaloLarg"
```

```
Iris2 <- Iris %>% select(-SepaloLung,-SepaloLarg , everything())
names(Iris2)
```

```
[1] "PetaloLung" "PetaloLarg" "Specie"     "SepaloLung" "SepaloLarg"
```

Trasformazione

Sostituzione di valori

```
# Su tutta la colonna

Class2 <- Class
Class2$Height <- Class2$Height * 2.54
Class2$Weight <- Class2$Weight * 0.453592
Class2
```

```
      Name Sex Age  Height   Weight
1   Alfred   M  14 175.260 51.02910
2    Alice   F  13 143.510 38.10173
3  Barbara   F  13 165.862 44.45202
```

```
# Per selezione dei valori
Class2$Name[Class2$Name == "Alfred"] <- "Alfredo"
Class2
```

```
      Name Sex Age  Height   Weight
1  Alfredo   M  14 175.260 51.02910
2    Alice   F  13 143.510 38.10173
3  Barbara   F  13 165.862 44.45202
```

Trasformazione

Conversione in valori mancanti

```
Class2 <- Class; Class2[2:3,3:5] <- -9999; Class2[4:5,1:2] <- "?";
head(Class2, n=5)
```

```
     Name Sex   Age  Height  Weight
1  Alfred   M    14    69.0   112.5
2   Alice   F -9999 -9999.0 -9999.0
3 Barbara   F -9999 -9999.0 -9999.0
4       ?   ?    14    62.8   102.5
5       ?   ?    14    63.5   102.5
```

```
# Converte valori a NA (Not Available)
library(dplyr)
Class2 <- Class2 %>%  mutate(Sex = na_if(Sex, "?"))
Class2 <- Class2 %>%  mutate(Age = na_if(Age, -9999))
Class2 <- Class2 %>%  mutate(Weight = na_if(Weight, -9999))
head(Class2, n=5)
```

```
     Name  Sex Age  Height  Weight
1  Alfred    M  14    69.0   112.5
2   Alice    F  NA -9999.0      NA
3 Barbara    F  NA -9999.0      NA
4       ? <NA>  14    62.8   102.5
5       ? <NA>  14    63.5   102.5
```

Trasformazione

Sostituzione dei valori mancanti

```
Class2 <- Class; Class2[2:3,3] <- NA;
head(Class2, n=5)
```

```
 Name Sex Age Height Weight
1  Alfred   M  14   69.0  112.5
2   Alice   F  NA   56.5   84.0
3 Barbara   F  NA   65.3   98.0
4   Carol   F  14   62.8  102.5
5   Henry   M  14   63.5  102.5
```

```
Class2$Age[is.na(Class2$Age)] <- mean(Class2$Age, na.rm = TRUE)
head(Class2, n=5)
```

```
 Name Sex      Age Height Weight
1  Alfred   M 14.00000   69.0  112.5
2   Alice   F 13.35294   56.5   84.0
3 Barbara   F 13.35294   65.3   98.0
4   Carol   F 14.00000   62.8  102.5
5   Henry   M 14.00000   63.5  102.5
```

```
# su tutte le colonne numeriche:
```

Trasformazione

Eliminazione dei valori estremi (outlier)

```
# Eliminazione manuale degli outlier

Class2 <- Class[-c(which(Class$Age < 13 | Class$Age > 14)), ]
```

```
      Name Sex Age Height Weight
1   Alfred   M  14   69.0  112.5
2    Alice   F  13   56.5   84.0
3  Barbara   F  13   65.3   98.0
4    Carol   F  14   62.8  102.5
5    Henry   M  14   63.5  102.5
9  Jeffrey   M  13   62.5   84.0
12   Judy    F  14   64.3   90.0
```

```
# Eliminazione con i percentili (sotto il 5% e sopra il 95%)

Class2 <- Class

Q <- quantile(Class2$Weight, probs=c(.05, .95), na.rm = FALSE)
Class2_outlier <- Class2[-which(Class2$Weight < Q[2] & Class2$Weight > Q[1]),]
Class2_outlier
```

```
     Name Sex Age Height Weight
11  Joyce   F  11   51.3   50.5
15 Philip   M  16   72.0  150.0
```

Trasformazione

Creazione variabili dummy – 1/2

```
# One-Hot Encoding
Class2 <- Class
Class2$F <- ifelse(Class$Sex == "F", 1, 0);
Class2$M <- ifelse(Class$Sex == "M", 1, 0); Class2
```

```
      Name Sex Age Height Weight F M
1   Alfred   M  14   69.0  112.5 0 1
2    Alice   F  13   56.5   84.0 1 0
3  Barbara   F  13   65.3   98.0 1 0
...
```

```
library(fastDummies)
Class2$Gruppo <- ifelse(Class2$Age < 13,"A",ifelse(Class2$Age < 15, B","C")); Class2
```

```
      Name Sex Age Height Weight Gruppo
1   Alfred   M  14   69.0  112.5      B
...
6    James   M  12   57.3   83.0      A
...
8    Janet   F  15   62.5  112.5      C
```

```
Class2 <- dummy_cols(Class2,select_columns="Gruppo",remove_selected_columns=F); Class2
```

```
      Name Sex Age Height Weight Gruppo Gruppo_A Gruppo_B Gruppo_C
1   Alfred   M  14   69.0  112.5      B        0        1        0
2    Alice   F  13   56.5   84.0      B        0        1        0
...
6    James   M  12   57.3   83.0      A        1        0        0
7     Jane   F  12   59.8   84.5      A        1        0        0
8    Janet   F  15   62.5  112.5      C        0        0        1
```

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Operatori aritmetici e logici

```
# Operatori di assegnazione
5 -> x
y <- 4
z = 4
```

```
[1] 5
[1] 4
[1] 4
```

```
# Operatori aritmetici: +, -, *, /
Classe <- Class
names(Classe) <- c("Nome","Genere","Età","Altezza_cm","Peso_kg")
Classe$Altezza_cm   <- Classe$Altezza_cm * 2.54
Classe$Peso_kg      <- Classe$Peso_kg * 0.453592
Classe
```

```
     Nome Genere Età Altezza_cm     Peso_kg
1   Alfred      M  14    175.260    51.02910
2    Alice      F  13    143.510    38.10173
3  Barbara      F  13    165.862    44.45202
...
```

Operatori aritmetici e logici

```
# Operatori Aritmetici: ^ (elev. potenza); sqrt (rad. quadrata)

Iris2 <- Iris
Iris2$RappPet  <- Iris2$PetaloLung / Iris2$PetaloLarg
Iris2$QuadPetL <- Iris2$PetaloLung ^ 2
Iris2$RadqPetL <- sqrt(Iris2$PetaloLung ); Iris2[1:5,5:8]
```

```
  Specie RappPet QuadPetL RadqPetL
1 setosa     7.0     1.96 1.183216
2 setosa     7.0     1.96 1.183216
3 setosa     6.5     1.69 1.140175
4 setosa     7.5     2.25 1.224745
5 setosa     7.0     1.96 1.183216
```

```
# Operatori Relazionali: == (uguale), != (diverso), < (minore), > (maggiore),
# <= (minore o uguale), >= (maggiore o uguale)

Iris2$Confronto1 <- Iris2$SepaloLung >= Iris2$PetaloLung * 3.08; head(Iris2,5)
```

```
 S epaloLung SepaloLarg PetaloLung PetaloLarg Specie RappPet QuadPetL RadqPetL Confronto1
1         5.1         3.5         1.4         0.2 setosa     7.0     1.96 1.183216       TRUE
2         4.9         3.0         1.4         0.2 setosa     7.0     1.96 1.183216       TRUE
3         4.7         3.2         1.3         0.2 setosa     6.5     1.69 1.140175       TRUE
4         4.6         3.1         1.5         0.2 setosa     7.5     2.25 1.224745      FALSE
5         5.0         3.6         1.4         0.2 setosa     7.0     1.96 1.183216       TRUE
```

Operatori aritmetici e logici

```
# Operatori Logici: ! (NOT), & (AND), | (OR)
Iris2$Confronto2 <- Iris2$SepaloLung >= Iris2$PetaloLung * 3.08 &
                    Iris2$SepaloLarg >= Iris2$PetaloLarg * 14.58; Iris2
```

```
    SepaloLung SepaloLarg PetaloLung PetaloLarg     Specie Confronto1 Confronto2
1          5.1        3.5        1.4        0.2     setosa       TRUE       TRUE
2          4.9        3.0        1.4        0.2     setosa       TRUE       TRUE
3          4.7        3.2        1.3        0.2     setosa       TRUE       TRUE
4          4.6        3.1        1.5        0.2     setosa      FALSE      FALSE
5          5.0        3.6        1.4        0.2     setosa       TRUE       TRUE
6          5.4        3.9        1.7        0.4     setosa       TRUE      FALSE
```

```
# Operatori Logici: %in% (IN)
Classe2 <- subset(Classe, Età %in% c(11,12,13)); Classe2
```

```
     Nome Genere Età Height Weight Altezza_cm  Peso_kg
2   Alice      F  13   56.5   84.0    143.510  8.10173
3 Barbara      F  13   65.3   98.0    165.862  4.45202
6   James      M  12   57.3   83.0    145.542  7.64814
...
```

```
library(dplyr)
Classe2 <- Classe %>% filter(Classe$Età %in% c(11,12,13)); Classe2
```

```
1   Alice      F  13   56.5   84.0 143.510 38.10173
2 Barbara      F  13   65.3   98.0 165.862 44.45202
3   James      M  12   57.3   83.0 145.542 37.64814
...
```

Manipolazione delle stringhe

```
# Sostituzione di caratteri
testo <- c("12357e", "12575e", "197e18", "e18947")
testo2 <- gsub("e", "", testo); testo2
```

```
[1] "12357" "12575" "19718" "18947"
```

```
stringa <- c("aaa123XYZ")
stringa <- gsub("[^0-9]","",stringa); stringa
```

```
[1] "123"
```

```
# sostituzione di caratteri in una sottostringa
substr(testo2,2,3) <- "**"; testo2
```

```
[1] "1**57" "1**75" "1**18" "1**47"
```

Manipolazione delle stringhe

```
# estrazione di una sottostringa (primi 3 caratteri)
testo3 <- substr(testo,1,3); testo3
```

```
[1] "123" "125" "197" "e18"
```

```
# estrazione di una sottostringa (ultimi 2 caratteri)
nc <- nchar(testo)
testo4 <- substr(testo, max(nc)-1, nc); testo4
```

```
[1] "7e" "5e" "18" "47"
```

```
# estrazione di una sottostringa dalla sua posizione
testo <- "Leonardo da Vinci"
testo2 <- substr(testo, regexpr("da", testo), nchar(testo)); testo2
```

```
[1] "da Vinci"
```

```
# unione di stringhe
testo1 <- "Mario" ; testo2 <- "Rossi"
testo <- paste(testo2,testo1, sep=", ")
[1] "Rossi, Mario"

data <- paste("Oggi è", format(Sys.Date(), format="%d/%m/%Y"), sep=" "); data
[1] "Oggi è 07/07/2022"
```

Strutture di controllo – Condizionali

```
# Istruzione condizionale if (condizione) { espressione } < else {} >

# non funziona su vettori o colonne di dataframe!

# Strutture if/else concatenate
# if ( condizione1 )
#  { esegue istruzioni se soddisfa condizione1
#  } else { if ( condizione2 )
#           { esegue istruzioni se soddisfa condizione2
#           } else { if ( condizione3 )
#                    { esegue istruzioni se soddisfa condizione3
#                    } else { esegue se non soddisfa alcuna condizione precedente }
#                  }
#         }
```

```
x <- -5; y <- 7
xy <- if (x <= 0) { -y + x} else { y - x }; xy
```

```
[1] -12
```

Strutture di controllo – Condizionali

```
# Istruzione condizionale ifelse (si utilizza su vettori o colonne di dataframe)

Class2 <- Class
Class2$Fascia_età <- ifelse ( Class2$Age <= 12, "Fanciullezza", "Adolescenza" )
Class2
```

```
      Name Sex Age Height Weight   Fascia_età
1   Alfred   M  14   69.0  112.5  Adolescenza
2    Alice   F  13   56.5   84.0  Adolescenza
3  Barbara   F  13   65.3   98.0  Adolescenza
4    Carol   F  14   62.8  102.5  Adolescenza
5    Henry   M  14   63.5  102.5  Adolescenza
6    James   M  12   57.3   83.0 Fanciullezza
```

```
# Alternativa a ifelse

Class2 <- Class
Class2$Fascia_età[Class2$Age <= 12] <- "Fanciullezza"
Class2$Fascia_età[Class2$Age > 12] <- "Adolescenza"
```

Strutture di controllo – Cicli

```
# Sintassi: for (indice in vettore) { espressione }

# Esempio: sostituzione dei valori mancanti con la media

Class2 <- Class; Class2[2:3,3] <- NA;

for (i in 3:ncol(Class2) ){
  Class2[is.na(Class2[,i]), i] <- mean(Class2[,i], na.rm = TRUE)
}
head(Class2, n=5)
```

```
     Name  Sex      Age  Height   Weight
1  Alfred    M 14.00000    69.0 112.5000
2   Alice    F 13.35294 -9999.0 101.0882
3 Barbara    F 13.35294 -9999.0 101.0882
4       ? <NA> 14.00000    62.8 102.5000
5       ? <NA> 14.00000    63.5 102.5000
```

Strutture di controllo – Cicli

```
# Esempio: calcolo del fattoriale

num = 7
fattoriale = 1
# Controlla se il numero è negative, zero, o positivo
if (num < 0) {
  print( "Non esistono fattoriali per numeri negativi" )
} else if (num == 0) { print( "Il fattoriale di 0 è 1" )
} else { for(i in 1:num) { fattoriale = fattoriale * i }
         print(paste( "Il fattoriale di" , num , "è" ,fattoriale))
       }

[1] "Il fattoriale di 7 è 5040"



# ovviamente esiste anche la funzione factorial()!

factorial(7)
[1] 5040
```

Strutture di controllo – Cicli (2/2)

```
attuali  <- c(0,   1,   0,   1,   0,   1,   1,   1,   0,   1,   1,   0,   1  )
predetti <- c(0.3, 0.6, 0.2, 0.7, 0.4, 0.5, 0.4, 0.8, 0.4, 0.7, 0.2, 0.8, 0.4)

soglia <- 0.5
veri_pos=0; veri_neg=0; falsi_pos=0; falsi_neg=0

for (i in 1:length(predetti)) {
if       (predetti[i] >= soglia & attuali[i] == 1) { veri_pos  = veri_pos  + 1 }
else  if (predetti[i] <  soglia & attuali[i] == 0) { veri_neg  = veri_neg  + 1 }
else  if (predetti[i] <  soglia & attuali[i] == 1) { falsi_pos = falsi_pos + 1 }
else                                               { falsi_neg = falsi_neg + 1 }
};

cat("\n Soglia:  ", soglia,    "\n",
" Veri negativi: ", veri_neg,  "\n",
"Falsi positivi: ", falsi_pos, "\n",
"Falsi negativi: ", falsi_neg, "\n",
" Veri positivi: ", veri_pos,  sep="")
```

```
 Soglia: 0.5
 Veri negativi: 4
Falsi positivi: 3
Falsi negativi: 1
 Veri positivi: 5
```

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Funzioni (esempio)

```
# Sintassi: nome_funzione <- function (input1 <, input2, . . ., inputn>) {
#                                output <- risultato_elaborazione
#                                return (output) }
#           nome_funzione(input1 <,input2,...>);
```

```
imputazione <- function (dati, FUN) {
for (i in 3:ncol(dati) ) { dati[is.na(dati[,i]), i] <- FUN (dati[,i], na.rm = TRUE) }
return (dati) }
```

```
Class2 <- Class; Class2[2:5,3] <- NA; Class2[4:7,4:5] <- NA; Class2
```

```
      Name Sex Age Height Weight
1   Alfred   M  14   69.0  112.5
2    Alice   F  NA   56.5   84.0
3  Barbara   F  NA   65.3   98.0
4    Carol   F  NA     NA     NA
5    Henry   M  NA     NA     NA
```

```
Class3 <- imputazione(Class2, median); Class3
```

```
      Name Sex Age Height Weight
1   Alfred   M  14   69.0  112.5
2    Alice   F  13   56.5   84.0
3  Barbara   F  13   65.3   98.0
4    Carol   F  13   64.3   99.5
5    Henry   M  13   64.3   99.5
```

Funzioni (esempio)

```
fattoriale <- function (num)
{
fattoriale = 1
# Controlla se il numero è negativo, zero, o positivo
if (num < 0) { return(NaN)}
else if (num == 0) { return(1)}
else { for(i in 1:num) { fattoriale = fattoriale * i }
return(fattoriale)
}
                              }

f <- fattoriale(7); f
```

```
[1] 5040
```

Funzioni (esempio)

```
Conversione <- function (tipo, x) {
   if (tipo == "F->C") { y <- (x - 32) * 5 / 9;
   out <- paste("Fahrenheit:",sprintf(x,fmt="%#.1f")," -> Celsius:",sprintf(y, fmt="%#.1f")) }
   if (tipo == "L->K") { y <- x / 2.2046;
   out <- paste("Libbre:", sprintf(x, fmt="%#.3f"), " -> Kg:", sprintf(y, fmt="%#.3f")) }
   if (tipo == "P->C") { y <- x / 0.39370;
   out <- paste("Pollici:", sprintf(x, fmt="%#.2f"), " -> cm:", sprintf(y, fmt="%#.2f")) }
   return(out)
}

ris <- Conversione("F->C", 78.5 ); ris

[1] "Fahrenheit: 78.5  -> Celsius: 25.8"

ris <- Conversione("L->K", 1); ris

[1] "Libbre: 1.000  -> Kg: 0.454"

ris <- Conversione("P->C", 1); ris

[1] "Pollici: 1.00  -> cm: 2.54"
```

Funzioni (esempio)

```
campionatore <- function (data, num, pct) {

   if (missing(num) & missing(pct)) { message ( "Mancano num e pct." ) ; return() }
   if (!missing(pct)) { if (pct > 1) { message ( "pct deve essere tra 0 e 1." );
   return() }}

   if (!missing(pct)) { num = round(pct*nrow(data)) }
   set.seed(12345)
   ind <- sample(nrow(data), num, replace = F)
   out <- data[ind,]

   message ( "Sono state selezionate " , nrow(out), " righe." )
   return (out);
}

Train_dataset  <- campionatore(data=Iris, num=, pct=0.2);
Sono state selezionate 30 righe.

Train_dataset  <- campionatore(data=Iris, num=30, pct=);
Sono state selezionate 30 righe.

Test_dataset   <- campionatore(data=Iris, num=, pct=1-0.2);
Sono state selezionate 120 righe.
```

Funzioni – Utilizzo in lapply

```
# Sintassi: lapply(X, FUN, ...)
# lapply ritorna una lista della stessa lunghezza di X, come risultato
# dell'applicazione della funzione FUN a ogni elemento di x

lapply(Class[3:5], mean);
```

```
$Age
[1] 13.31579

$Height
[1] 62.33684
```

```
lapply(Class[c("Age", "Weight")], median);
```

```
$Age
[1] 13

$Height
[1] 62.8
```

```
# Esempio: applicazione della normalizzazione Z-score per alcune colonne
zscore <- function(x) { output <- (x - mean(x, na.rm = T)) / sd(x, na.rm = T );
                        return (output); }

col <- c("Age","Weight");
newcol <- paste0(col,"_Norm")

Class[newcol] <- lapply(Class[col], zscore); Class
```

```
      Name Sex Age Height Weight   Age_Norm Weight_Norm
1   Alfred   M  14   69.0  112.5  0.4583796  0.54771760
2    Alice   F  13   56.5   84.0 -0.2115598 -0.70371312
                        ...
18  Thomas   M  11   57.5   85.0 -1.5514388 -0.65980327
19 William   M  15   66.5  112.0  1.1283191  0.52576268
```

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Combinare e raggruppare dati

Concatenazione colonne

```
Classe1 <- Classe[,c("Nome","Genere","Età")]; Classe1
```

```
      Nome Genere Età
1   Alfred      M  14
2    Alice      F  13
3  Barbara      F  13
...
```

```
Classe2 <- Classe[,c(4,5)]; Classe2
```

```
   Altezza_cm  Peso_kg
1     175.260 51.02910
2     143.510 38.10173
3     165.862 44.45202
...
```

```
Classe_3 <- cbind(Classe1, Classe2); Classe3
```

```
      Nome Genere Età Altezza_cm  Peso_kg
1   Alfred      M  14    175.260 51.02910
2    Alice      F  13    143.510 38.10173
3  Barbara      F  13    165.862 44.45202
...
```

Combinare e raggruppare dati

Concatenazione righe

```
Classe1 <- Classe[1:10,]; Classe1
```

```
      Nome Genere Età Altezza_cm  Peso_kg
1   Alfred      M  14    175.260 51.02910
2    Alice      F  13    143.510 38.10173
...
10    John      M  12 149.860 45.13240
```

```
Classe2 <- Classe[11:nrow(Classe),]; Classe2
```

```
      Nome Genere Età Altezza_cm  Peso_kg
11   Joyce      F  11    130.302 22.90640
12    Judy      F  14    163.322 40.82328
...
```

```
Classe2 <- rbind(Classe1, Classe2); Classe2
```

```
      Nome Genere Età Altezza_cm  Peso_kg
1   Alfred      M  14    175.260 51.02910
2    Alice      F  13    143.510 38.10173
...
10    John      M  12    149.860 45.13240
11   Joyce      F  11    130.302 22.90640
12    Judy      F  14    163.322 40.82328
...
```

Combinare e raggruppare dati

Combinazione – 1/3

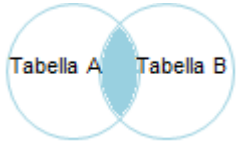
```
Prodotto <- c("Sedie", "Poltrone", "Divani"); Previsione <- c(1000, 2000, 5000)
Prev <- data.frame (Prodotto, Previsione);

Prodotto <- c("Divani","Sedie","Altro"); Vendite <- c(800,300,50)
Cons <- data.frame (Prodotto, Vendite);
```

Prev

	Prodotto	Previsione
1	Sedie	1000
2	Poltrone	2000
3	Divani	5000

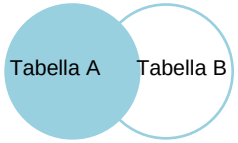
Cons

	Prodotto	Vendite
1	Divani	800
2	Sedie	300
3	Altro	50




```
Inner_join <- merge(Prev, Cons, by = "Prodotto" );
Inner_join
```

	Prodotto	Previsione	Vendite
1	Divani	5000	800
2	Sedie	1000	300




```
Left_join  <- merge(x=Prev, y=Cons, by = "Prodotto", all.x=TRUE);
Left_join
```

1	Divani	5000	800
2	Poltrone	2000	NA
3	Sedie	1000	300

Combinare e raggruppare dati

Combinazione – 2/3

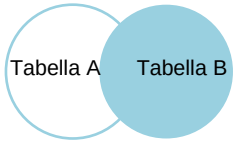
Prev

	Prodotto	Previsione
1	Sedie	1000
2	Poltrone	2000
3	Divani	5000

Cons

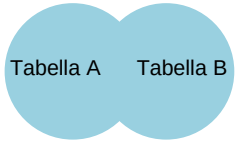
	Prodotto	Vendite
1	Divani	800
2	Sedie	300
3	Altro	50

```
Right_join <- merge(x=Prev, y=Cons, by = "Prodotto", all.y=TRUE);
Right_join
```



	Prodotto	Previsione	Vendite
1	Altro	NA	50
2	Divani	5000	800
3	Sedie	1000	300

```
Full_join  <- merge(x=Prev, y=Cons, by = "Prodotto", all=TRUE);
Full_join
```



	Prodotto	Previsione	Vendite
1	Altro	NA	50
2	Divani	5000	800
3	Poltrone	2000	NA
4	Sedie	1000	300

Combinazione – 3/3

```
library(plyr)

Inner_join <- join(Prev, Cons, by= "Prodotto", type = "inner" ); Inner_join
  Prodotto Previsione Vendite
1    Sedie       1000     300
2   Divani       5000     800

Left_join <- join(Prev, Cons, by= "Prodotto", type = "left" ); Left_join
  Prodotto Previsione Vendite
1    Sedie       1000     300
2 Poltrone       2000      NA
3   Divani       5000     800

Right_join <- join(Prev, Cons, by= "Prodotto", type = "right" ); Right_join
  Prodotto Previsione Vendite
1   Divani       5000     800
2    Sedie       1000     300
3    Altro         NA      50

Full_join <- join(Prev, Cons, by= "Prodotto", type = "full" ); Full_join
  Prodotto Previsione Vendite
1    Sedie       1000     300
2 Poltrone       2000      NA
3   Divani       5000     800
4    Altro         NA      50
```

Combinare e raggruppare dati

Transcodificazione (Lookup Table)

```
library(plyr)

lookup <- data.frame(Sex = c("F","M"), Sex_descr = c("Female","Male"))

Class2 <- Class
Class2 <- join(Class2, lookup, by = "Sex",  type="left")[-2]; Class2
```

```
      Name Age Height Weight Sex_descr
1   Alfred  14   69.0  112.5      Male
2    Alice  13   56.5   84.0    Female
              ...
9  Jeffrey  13   62.5   84.0      Male
10    John  12   59.0   99.5      Male
```

Combinare e raggruppare dati

Classificazione di variabili numeriche in gruppi

```
library(dplyr)

Class2 <- Class
Class2$Age_Groups <- cut( Class2$Age,
                          breaks = c(0,   12,     14,   18),
                          labels = c(  "<=11","12-13",">13"), right = F); Class2
```

```
1   Alfred   M  14   69.0  112.5       >13
2    Alice   F  13   56.5   84.0     12-13
...
18  Thomas   M  11   57.5   85.0      0-11
19 William   M  15   66.5  112.0       >13
```

```
Class2 %>% group_by(Age_Groups) %>%
   summarise(n=n(), min_Age= min(Age), max_Age = max(Age), mean_Age=mean(Age))
   %>% ungroup
```

```
  Age_Groups     n min_Age max_Age mean_Age
  <fct>      <int>   <int>   <int>    <dbl>
1 0-11           2      11      11     11
2 12-13          8      12      13     12.4
3 >13            9      14      16     14.7
```

Combinare e raggruppare dati

Raggruppamento

```
library(dplyr)
Classe_grp <- summarise(Classe,
                Numero = n(),
                Media_peso = mean(Peso_kg),
                Mediana_Peso = median(Peso_kg))
```

```
  Numero Media_peso Mediana_Peso
1     19   45.37114      45.1324
```

```
# per gruppi
Classe_grp <- Classe %>%
                group_by (Genere) %>%
                  summarise(Numero = n(),
                            Media_Peso = mean(Peso_kg),
                            Mediana_Peso = median(Peso_kg))
```

```
  Genere Numero Media_Peso Mediana_Peso
1      F      9   40.87368     40.82328
2      M     10   49.41885     48.64774
```

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Statistiche di base

Statistiche descrittive

```
# su output

summary(Iris)
```

```
   SepaloLung      SepaloLarg      PetaloLung      PetaloLarg      Specie
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Length:150
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Class :character
Median :5.800   Median :3.000   Median :4.350   Median :1.300   Mode  :character
Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

```
# su file

stat <- data.frame()
stat[1,1] <- mean(Iris$PetaloLung); names(stat)[1]  <- "Media_PetLun"; stat
```

```
   Media_PetLun
1         3.758
```

Statistiche di base

Statistiche descrittive (su file) – 1/3

```
Iris2 <- Iris
Iris2[1:15, c( "PetaloLung" )]  <- NA            # si forzano 15 valori mancanti (per % Mancanti)

stat[1,1]  <- length(Iris2$PetaloLung[!is.na(Iris2$PetaloLung)]);names(stat)[1]  <- "Num. (validi)"
stat[1,2]  <- sum(Iris2$PetaloLung, na.rm=T);                    names(stat)[2]  <- "Somma"
stat[1,3]  <- sum(is.na(Iris2$PetaloLung))/nrow(Iris2)*100;      names(stat)[3]  <- "% Mancanti"
stat[1,4]  <- min(Iris2$PetaloLung, na.rm=T);                    names(stat)[4]  <- "Minimo"
stat[1,5]  <- quantile[1](Iris2$PetaloLung, probs=0.01, na.rm=T);  names(stat)[5]  <- "1° Pct.le"
stat[1,6]  <- quantile(Iris2$PetaloLung, probs=0.05, na.rm=T);   names(stat)[6]  <- "5° Pct.le"
stat[1,7]  <- quantile(Iris2$PetaloLung, probs=0.10, na.rm=T);   names(stat)[7]  <- "10° Pct.le"
stat[1,8]  <- quantile(Iris2$PetaloLung, probs=0.25, na.rm=T);   names(stat)[8]  <- "25° Pct.le"
stat[1,9]  <- mean(Iris2$PetaloLung, na.rm=T);                   names(stat)[9]  <- "Media"
stat[1,10] <- median(Iris2$PetaloLung, na.rm=T);                 names(stat)[10] <- "Mediana"
              mode <- function(x) { x2 <- x[!is.na(x)]; u <- unique(x2);
                     tab <- tabulate(match(x2, u)); u[tab == max(tab)] }
stat[1,11] <- mode(Iris2$PetaloLung);                            names(stat)[11] <- "Moda"
```

[1] Per il calcolo dei percentili R utilizza il metodo **R-7** (si può cambiare con l'argomento **type=**).

Statistiche di base

Statistiche descrittive (su file) – 2/3

```
stat[1,12] <- sd(Iris2$PetaloLung, na.rm=T);                    names(stat)[12]  <- "Dev. Std"
stat[1,13] <- quantile(Iris2$PetaloLung, probs=0.75, na.rm=T); names(stat)[13]  <- "75° Pct.le"
stat[1,14] <- quantile(Iris2$PetaloLung, probs=0.90, na.rm=T); names(stat)[14]  <- "90° Pct.le"
stat[1,15] <- quantile(Iris2$PetaloLung, probs=0.95, na.rm=T); names(stat)[15]  <- "95° Pct.le"
stat[1,16] <- quantile(Iris2$PetaloLung, probs=0.99, na.rm=T); names(stat)[16]  <- "99° Pct.le"
stat[1,17] <- IQR(Iris2$PetaloLung, na.rm=T);                   names(stat)[17]  <- "IQR"
stat[1,18] <- max(Iris2$PetaloLung, na.rm=T);                   names(stat)[18]  <- "Massimo"
stat
```

```
  Num. (validi) Somma % Mancanti Minimo 1° Pct.le 5° Pct.le 10° Pct.le 25° Pct.le    Media
1           135 542.4         10      1     1.234       1.3        1.4        1.9 4.017778
  Mediana Moda Dev. Std 75° Pct.le 90° Pct.le 95° Pct.le 99° Pct.le  IQR Massimo
1     4.5  1.5 1.668222       5.15        5.8        6.1        6.7 3.25     6.9
```

```
stat2 <- data.frame(t(stat)); stat2      # Traspone colonne/righe
```

```
                      X1
Num. (validi) 135.000000
Somma         542.400000
% Mancanti     10.000000
Minimo          1.000000
1° Pct.le       1.234000
5° Pct.le       1.300000
10° Pct.le      1.400000
25° Pct.le      1.900000
Media           4.017778
...
```

Statistiche descrittive (su file) – 33

```
library(tibble)
stat <- rownames_to_column(stat2, var="Statistica" )
names(stat2)[names(stat2) == "X1" ] <- "Valore"; stat2
```

```
       Statistica     Valore
1  Num.  (validi) 135.000000
2           Somma 542.400000
3      % Mancanti  10.000000
4          Minimo   1.000000
...
```

```
format(stat2, digits=3, decimal.mark=",", big.mark=".")
```

```
       Statistica Valore
1  Num.  (validi) 135,00
2           Somma 542,40
3      % Mancanti  10,00
4          Minimo   1,00
...
```

Approfondimento:
https://bookdown.org/mikemahoney218/LectureBook/basic-statistics-using-r.html

Statistiche di base

Frequenze

```
tabulate(Classe$Età)
```

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

```
freq <- tabulate(match(Classe$Età, c(11,12,13,14,15,16) ))
names(freq) <- c(11,12,13,14,15,16); freq
```

```
11 12 13 14 15 16
 2  5  3  4  4  1
```

```
freq <- table(Classe$Età); freq
```

```
11 12 13 14 15 16
 2  5  3  4  4  1
```

Tabelle incrociate – 1/3

```
table(Classe$Età, Classe$Genere)
```

```
    F M
11  1 1
12  2 3
13  2 1
14  2 2
15  2 2
16  0 1
```

```
tab <- table(Classe$Età, Classe$Genere)
names(dimnames(tab)) <- c( "Età", "Genere" ); tab
```

```
     Genere
Età   F M
  11  1 1
  12  2 3
  13  2 1
  14  2 2
  15  2 2
  16  0 1
```

Tabelle incrociate – 2/3

```
tab <- prop.table(table(Classe$Età, Classe$Genere),1);  # percentuali di riga
names(dimnames(tab)) <- c( "Età", "Genere" ); tab
```

```
  Genere
Età         F         M
11 0.5000000 0.5000000
12 0.4000000 0.6000000
13 0.6666667 0.3333333
14 0.5000000 0.5000000
15 0.5000000 0.5000000
16 0.0000000 1.0000000
```

```
options (digits=2) # limita i decimali
```

```
tab <- prop.table(table(Classe$Età, Classe$Genere),2); # percentuali di colonna
names(dimnames(tab)) <- c( "Età", "Genere" ); tab
```

```
  Genere
Età    F    M
11 0.11 0.10
12 0.22 0.30
13 0.22 0.10
14 0.22 0.20
15 0.22 0.20
16 0.00 0.10
```

Tabelle incrociate – 3/3

```
library(gmodels)
CrossTable(Classe$Età, Classe$Genere,
       digits=2, expected=F, prop.r=T, prop.c=T, prop.t=F, prop.chisq=F,
       resid=T, asresid=T, format=c("SPSS"), dnn = c("Età","Genere"))
```

```
 Cell Contents
|-------------------------|
|                   Count |
|             Row Percent |
|          Column Percent |
|                Residual |
|           Adj Std Resid |
|-------------------------|

==================================
         Genere
Età           F         M    Total
----------------------------------
11            1         1        2
          50.00%    50.00%   10.53%
          11.11%    10.00%
           0.05     -0.05
           0.08     -0.08
----------------------------------
12            2         3        5
          40.00%    60.00%   26.32%
          22.22%    30.00%
          -0.37      0.37
          -0.38      0.38
----------------------------------

...
```

Associazioni

```
tab <- table(Classe$Età, Classe$Genere)
chi <- chisq.test(tab, correct=F);        # test chi quadrato

chi$residuals; chi$stdres; chi$parameter; round(chi$p.value,4)
```

```
> chi$residuals;
                F            M
11  0.054073807 -0.051298918
12 -0.239394949  0.227109990
...
16 -0.688247202  0.652928625

> chi$stdres;
                F            M
11  0.078798162 -0.078798162
12 -0.384418753  0.384418753
...
16 -0.974679434  0.974679434

> chi$parameter;
df
 5

> round(chi$p.value,4)
[1] 0.9148
```

Associazioni – Diagramma a mosaico

```
library(readxl)
path <- "C:/Temp/Colore_Occhi_capelli.xlsx"
df1 <- read_xlsx(path, col_names = TRUE)

library(dplyr)
df2 <- data.frame(df1 %>% group_by(Occhi, Capelli) %>% dplyr::summarise(freq =
sum(Frequenza)))

df3 <- xtabs(data=df2, formula= freq ~ Occhi + Capelli )
chi <- chisq.test(df3, correct=F);

txt1 <- chi$observed;
txt2 <- round(chi$residuals,1);
txt <- matrix(paste0(txt1, "\n",txt2), nrow=nrow(txt2), ncol=ncol(txt2));

dimnames(txt) <- dimnames(txt2) # assegna nomi delle colonne e delle righe alla matrice txt

library(vcd)
vcd::mosaic( data=df3, ~ Occhi + Capelli, main="", pop=FALSE,
      gp = shading_hsv (h=c(0.55, 0.1),                                    # colore tonalità HSV
      df=chi$parameter, expected = chi$expected, residuals = chi$residuals),
      gp_varnames = gpar(cex=3, fontsize = 5),                             # variabili
      gp_labels = gpar(fontsize = 15),                                     # modalità
      main_gp = gpar(cex=5, fontface="bold", fontsize = 4))                # titolo
labeling_cells( text=txt, margin = 0, gp_text = gpar(fontsize = 18))(df3) # celle
```

Associazioni – Diagramma a mosaico

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Correlazione

```
corr <- cor(Classe$Peso_kg, Classe$Altezza_cm, method = "pearson"); corr
```

```
[1] 0.8777852
```

```
corr.test <- cor.test(Classe$Peso_kg, Classe$Altezza_cm, method = "pearson");
corr.test
```

```
        Pearson's product-moment correlation

data:  Classe$Peso_kg and Classe$Altezza_cm
t = 7.5549, df = 17, p-value = 7.887e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7044314 0.9523101
sample estimates:
      cor
0.8777852
```

Grafico a pag. 104

Regressione lineare

```
Inflazione  <- c(2,3.5,2.8,4.7,5.1,6.8,2.4,2.8)
Occupazione <- c(9,8.2,8.6,7  ,6.4,4.8,6.9,7.6)
Tassi <- data.frame(Occupazione, Inflazione)

mod <- lm(Inflazione ~ Occupazione, data=Tassi)                # costruzione
summary(mod)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.3709     1.8818   6.043 0.000929 ***
Occupazione  -1.0405     0.2536  -4.103 0.006335 **
Residual standard error: 0.9068 on 6 degrees of freedom
Multiple R-squared:  0.7373, Adjusted R-squared:  0.6935
```

```
Tassi$Valori_previsti <- predict(mod,data.frame(Occupazione)); # applicazione
Tassi
```

```
  Occupazione Inflazione Valori_previsti
1         9.0        2.0        2.006715
2         8.2        3.5        2.839087
3         8.6        2.8        2.422901
...
```

```
prev <-  predict(mod, data.frame(Occupazione=6)); prev          # singola previsione
```

```
       1
5.128111
```

Grafico a pag. 104

Regressione logistica – 1/4

```
library(readxl)
Studio <- as.data.frame(read_xlsx("C:/Temp/Studio_farmaco.xlsx", col_names = T));
Studio
```

```
 Dolore   Terapia Sesso Eta
1      No   Placebo     F  68
2      No Farmaco B     M  74
3      No   Placebo     F  67
...
```

```
Studio$Dolore  <- as.factor(Studio$Dolore)
Studio$Terapia <- as.factor(Studio$Terapia)
Studio$Sesso   <- as.factor(Studio$Sesso)

mod <- glm(data=Studio, relevel(Dolore, ref = "Sì") ~
                        relevel(Terapia, ref = "Placebo") +
                        relevel(Sesso,ref = "M")  + Eta, family=binomial())
summary(mod)
```

```
Estimate Std. Error z value Pr(>|z|)
(Intercept)                                15.86690    6.40509   2.477  0.01324 *
relevel(Terapia, ref = "Placebo")Farmaco A  3.17896    1.01348   3.137  0.00171 **
relevel(Terapia, ref = "Placebo")Farmaco B  3.72638    1.13377   3.287  0.00101 **
relevel(Sesso, ref = "M")F                  1.82353    0.79195   2.303  0.02130 *
Eta                                        -0.26496    0.09591  -2.763  0.00573 **
```

Regressione logistica – 2/4

```
Stats <- data.frame(coef(mod));
stats$OR <-format(exp(stats$coef.mod.),digits=3,scientific=F); stats
```

```
                                        coef.mod.          OR
(Intercept)                            15.8668986 7778690.495
relevel(Terapia, ref = "Placebo")Farmaco A  3.1789648      24.022
relevel(Terapia, ref = "Placebo")Farmaco B  3.7263795      41.528
relevel(Sesso, ref = "M")F                  1.8235267       6.194
Eta                                        -0.2649597       0.767
```

```
Studio$P_Dolore_No <- predict(mod, type="response", newdata=Studio)
Studio$P_Dolore_Sì <- 1 - Studio$P_Dolore_No

Studio$Prediction_Dolore <- ifelse(Studio$P_Dolore_No > 0.5, "No", "Sì");
Studio
```

```
   Dolore   Terapia Sesso Eta P_Dolore_Sì P_Dolore_No Prediction_Dolore
1      No   Placebo     F  68  0.58098940 0.419010602               Sì
2      No Farmaco B     M  74  0.50343541 0.496564594               Sì
3      No   Placebo     F  67  0.51546421 0.484535794               Sì
4      Sì   Placebo     M  66  0.83485629 0.165143710               Sì
5      No Farmaco B     F  67  0.02497707 0.975022934               No
6      No Farmaco B     F  77  0.26602090 0.733979101               No
...
```

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Alberi di Decisione – 1/3

```
library(readxl)
Credit <- as.data.frame(read_xlsx("C:/Temp/Credit_Scoring.xlsx", col_names = T)); Credit
```

```
  ID Solvenza        Saldo_Conto Durata_in_mesi                    Storia               Scopo Importo   ...
1  1        1                <0             6 Criticità/Crediti con altri     Televisore/HiFi     598
2  2        0             0<100            48        Crediti in essere ok     Televisore/HiFi    3043
3  3        1               N/D            12 Criticità/Crediti con altri           Istruzione    1072
4  4        1                <0            42        Crediti in essere ok   Mobili/Arredamento    4030
5  5        0                <0            24        Pagamenti in ritardo        Auto (nuova)    2490
...
```

```
Credit$Solvenza <- as.factor(Credit$Solvenza)

library(splitTools)    # partizionamento alternativo

indici <- partition(Credit$Solvenza, type="stratified", p=c(train = 0.8,valid = 0.2), seed=12345)
Train <- Credit[indici$train, -1]
Valid <- Credit[indici$valid, -1]

library(rpart)
Credit$Solvenza = relevel(Credit$Solvenza, ref = "0")

set.seed(12345)
mod  <-  rpart (data=Train, formula=Solvenza ~ . ,
                method="class",
                minsplit=10,
                minbucket=5,
                maxdepth=4,
               #xval=10,
                parms=list(split="gini"))
```

Alberi di Decisione – 2/3

```
probmtx <- predict(mod, type="prob", newdata=Valid)
prob <- data.frame(probmtx[,c(1,2)])

Valid$P_Solvenza_0 <- prob[,c(1)]
Valid$P_Solvenza_1 <- prob[,c(2)]
Valid$Prediction_Solvenza <- ifelse(Valid$P_Solvenza_1 > 0.5, "1", "0"); Valid
```

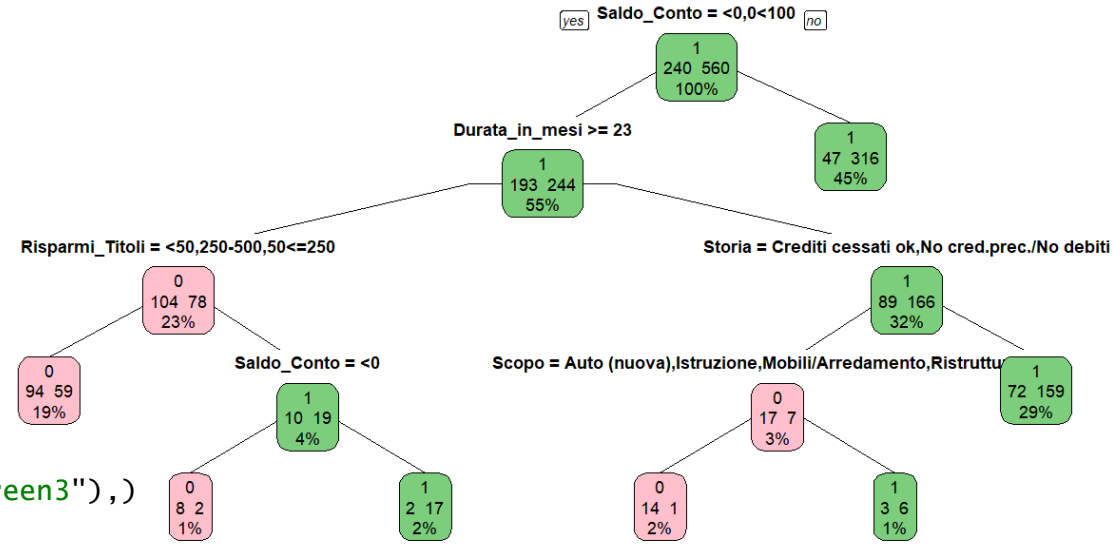
```
   Solvenza ... P_Solvenza_0 P_Solvenza_1 Prediction_Solvenza
6         1        0.1294766   0.87052342                  1
8         1        0.6143791   0.38562092                  0
9         1        0.1294766   0.87052342                  1
...
```

```
cm <- table(Valid$Solvenza, Valid$Prediction_Solvenza );   # matrice confusione alternativa

accuracy <- sum(cm[1], cm[4]) / sum(cm[1:4])
precision <- cm[4] / sum(cm[4], cm[2])
sensitivity <- cm[4] / sum(cm[4], cm[3])
fscore <- (2 * (sensitivity * precision))/(sensitivity + precision)
specificity <- cm[1] / sum(cm[1], cm[2])

s1 <- paste("Accuracy   :" , format(accuracy, digits=2))
s2 <- paste("Specificity:" , format(specificity, digits=2))
s3 <- paste("Sensitivity:" , format(sensitivity, digits=2))
s4 <- paste("Precision  :" , format(precision, digits=2))
s5 <- paste("F score    :" , format(fscore, digits=2))
st <- paste("",s1,s2,s3,s4,s5, sep="\n");
```

Alberi di Decisione – 3/3

```
cm; cat(st)

     0   1
  0  33  27
  1  20 120

Accuracy   : 0.76
Specificity: 0.62
Sensitivity: 0.82
Precision  : 0.86
F score    : 0.84
```

```
library(rpart.plot)
prp(mod,
    type=1,
    varlen=0,
    extra=101,
    faclen=0,
    tweak=1.5,
    box.palette = c("pink", "palegreen3"),)
```



Random Forest – 1/3

```
library(readxl)
Credit <- as.data.frame(read_xlsx("C:/Temp/Credit_Scoring.xlsx", col_names = T)); Credit
```

```
   ID Solvenza        Saldo_Conto Durata_in_mesi                    Storia              Scopo Importo   ...
1   1        1                <0              6 Criticità/Crediti con altri     Televisore/HiFi     598
2   2        0            0<100             48        Crediti in essere ok     Televisore/HiFi    3043
3   3        1              N/D             12 Criticità/Crediti con altri           Istruzione    1072
4   4        1                <0             42        Crediti in essere ok   Mobili/Arredamento    4030
5   5        0                <0             24        Pagamenti in ritardo        Auto (nuova)    2490
...
```

```
library(caret)
set.seed(12345)
df.index <- createDataPartition(Credit$Solvenza, p = 0.8, list = FALSE)
Train <- Credit[ df.index,]; Test  <- Credit[-df.index,]

library(randomForest)
Train$Solvenza <- as.factor(Train$Solvenza)
set.seed(12345)
rf <- randomForest(Solvenza ~ ., data=Train, ntree = 500, nodesize = 5, proximity=TRUE,
importance=TRUE);

pred <- predict(rf, newdata=Test, type="prob")
Test$P_Solvenza_1 <- pred[,2]; Test$P_Solvenza_0 <- pred[,1];
Test$Solvenza     <- as.factor(Test$Solvenza)
Test$I_Solvenza   <- as.factor(ifelse(pred[,2] > 0.5, 1, 0))
```

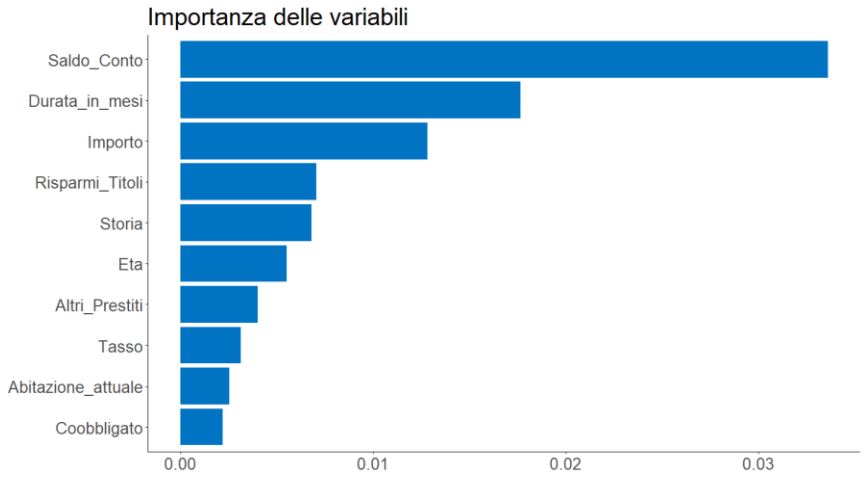
Random Forest – 2/3

```
library(readxl)
confusionMatrix(reference= Test$Solvenza, data= Test$I_Solvenza, positive = "1",
mode = "everything", dnn = c("Reference","Prediction"));
```

```
Confusion Matrix and Statistics

          Prediction
Reference   0   1
        0  29  17
        1  26 128

               Accuracy : 0.785
                 95% CI : (0.72153, 0.83982)
    No Information Rate : 0.725
    P-Value [Acc > NIR] : 0.031959

                  Kappa : 0.43197

 Mcnemar's Test P-Value : 0.222469

            Sensitivity : 0.88276
            Specificity : 0.52727
         Pos Pred Value : 0.83117
         Neg Pred Value : 0.63043
              Precision : 0.83117
                 Recall : 0.88276
                     F1 : 0.85619
             Prevalence : 0.72500
         Detection Rate : 0.64000
   Detection Prevalence : 0.77000
      Balanced Accuracy : 0.70502

       'Positive' Class : 1
```

Random Forest – 3/3

```
library(readxl)
imp <- data.frame(importance(rf, scale=FALSE, type=1));

library(dplyr)
imp10 <- imp %>% mutate(names=rownames(imp)) %>% arrange(desc(MeanDecreaseAccuracy)) %>% top_n(10,
 MeanDecreaseAccuracy)

library(ggplot2)
ggplot(data=imp10, aes(x = reorder(names, MeanDecreaseAccuracy), y = MeanDecreaseAccuracy)) +
   geom_col(fill="#0073C2FF") + coord_flip() +
   labs(title = "Importanza delle variabili", x= "", y= "") +
   theme_classic() + theme(text = element_text(size=20))
```



PCA e Cluster Analysis

```
# 1974 Motor Trend US magazine
library(readxl)
path <- "C:/Temp/mtcars.xlsx"
mtcars <- read_xlsx(path, sheet= "Foglio1", col_names = TRUE); names(mtcars)
```

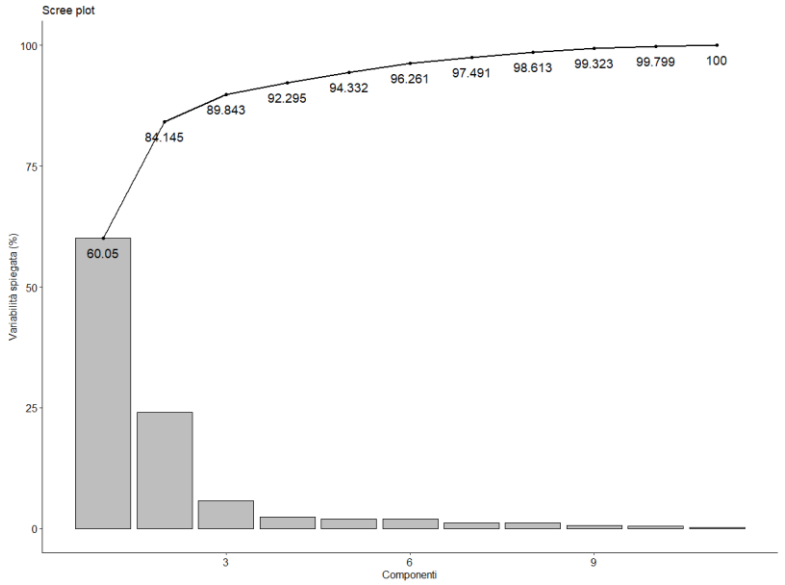
```
 [1] "Modello"
 [2] "Consumo (Km/litro)"
 [3] "Numero di cilindri"
 [4] "Cilindrata (cm cubici)"
 [5] "Cavalli vapore"
 [6] "Rapporto assale posteriore"
 [7] "Peso (kg)"
 [8] "Tempo per 400 metri (secondi)"
 [9] "Geometria motore (0=V, 1=in linea)"
[10] "Trasmissione (0=automatica, 1=manuale)"
[11] "Numero marce"
[12] "Numero carburatori"
```

```
mtcars_n <- mtcars[,c(2:12)]

library(FactoMineR)
pca <- PCA(mtcars_n, scale.unit=T, ncp=3, graph=FALSE)

pca$eig
```

```
        eigenvalue   percentage of variance   cumulative percentage of variance
comp 1  6.60552290              60.0502082                            60.05021
comp 2  2.65043467              24.0948607                            84.14507
comp 3  0.62682143               5.6983766                            89.84345
```

PCA e Cluster Analysis

```
library(ggplot2)

eig <- data.frame(pca$eig)
eig$comp <- 1:nrow(eig)

plot1 <- ggplot(eig) +
  geom_bar(aes(x=comp,y=percentage.of.variance), stat="identity", fill="grey",color="black") +
  geom_line (aes(x=comp,y=cumulative.percentage.of.variance),stat="identity",color="black",size=1) +
  geom_text (aes(x=comp,y=cumulative.percentage.of.variance,
   label=round(cumulative.percentage.of.variance,3), vjust=+2), size=5) +
  geom_point(aes(x=comp,y=cumulative.percentage.of.variance)) +
  labs(title= "Scree plot", x="Componenti", y="Variabilità spiegata (%)") +
  theme_classic() +
  theme (
    axis.text=element_text(size=12,color="black"),
    axis.title.x=element_text(size=12),
    axis.title.y=element_text(size=12))
plot1
```



PCA e Cluster Analysis

```
Loadings <- as.data.frame(pca$var$cor)
Loadings <- cbind(labels = rownames(Loadings), Loadings)
rownames(Loadings) <- NULL
Loadings
```

	labels	Dim.1	Dim.2	Dim.3
1	Consumo (Km/litro)	**-0.9304648**	0.02827583	-0.17852713
2	Numero di cilindri	**0.9613365**	0.07174912	-0.13826071
3	Cilindrata (cm cubici)	**0.9461766**	-0.07976734	-0.04943612
4	Cavalli vapore	0.8482052	0.40548724	0.11099991
5	Rapporto assale posteriore	-0.7564093	0.44673552	0.12767235
6	Peso (kg)	0.8894807	-0.23245959	0.26993130
7	Tempo per 400 metri (secondi)	-0.5150002	**-0.75458654**	0.31946162
8	Geometria motore (0=V, 1=in linea)	-0.7881009	-0.37760694	0.33864741
9	Trasmissione (0=automatica, 1=manuale)	-0.6044798	**0.69881831**	-0.16358658
10	Numero marce	-0.5324294	**0.75238520**	0.22936769
11	Numero carburatori	0.5498655	**0.67355477**	0.41928215

PCA e Cluster Analysis

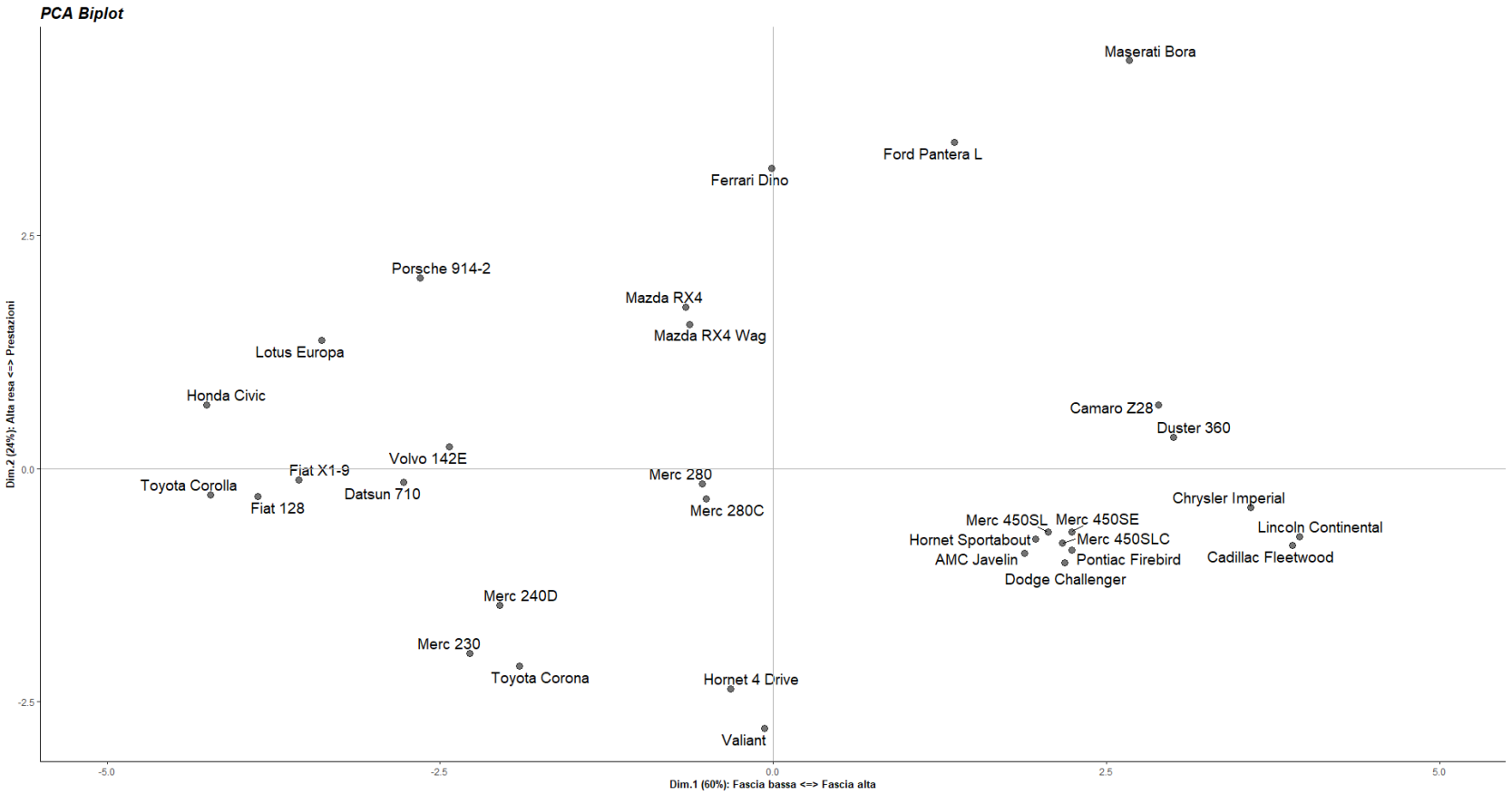
```
coord <- data.frame(mtcars[,1], pca$ind$coord);
coord
```

```
              Modello       Dim.1      Dim.2      Dim.3
1           Mazda RX4 -0.65735453  1.7352524 -0.5998012
2       Mazda RX4 Wag -0.62915675  1.5499369 -0.3809360
3          Datsun 710 -2.77114826 -0.1484201 -0.2383746
 ...
```

```
library(ggrepel)

plot2 <- ggplot(coord,aes(x=Dim.1, y=Dim.2)) +
  geom_text_repel(aes(label = Modello), size=5, max.overlaps = 20) +
  theme_classic() +
  geom_hline(yintercept = 0, color = "gray70") +
  geom_vline(xintercept = 0, color = "gray70") +
  geom_point( alpha = 0.55, size = 3) +
  xlab("Dim.1 (60%): Fascia") +
  ylab("Dim.2 (24%): Prestazioni") +
  xlim(-5, 5) +
  ggtitle("PCA Biplot") +
  theme(
    plot.title = element_text(color="black", size=15, face="bold.italic"),
    axis.title.x = element_text(color="black", size=12, face="bold"),
    axis.title.y = element_text(color="black", size=12, face="bold")
  )
```

PCA e Cluster Analysis

```
plot2
```



PCA e Cluster Analysis

```
set.seed(12345)

kmeans <- kmeans (pca$ind$coord,
                  centers=4,
                  nstart=1,
                  iter.max=100,
                  algorithm="Hartigan-Wong" )


final <- cbind(mtcars, pca$ind$coord, kmeans$cluster); final
```

	Modello	Efficienza	N. Cilindri	...	Dim.1	Dim.2	Dim.3	kmeans.cluster
1	Mazda RX4	3.8	6	...	-0.65735453	1.7352524	-0.5998012	2
2	Mazda RX4 Wag	3.8	6	...	-0.62915675	1.5499369	-0.3809360	2
3	Datsun 710	4.1	4	...	-2.77114826	-0.1484201	-0.2383746	4
4	Hornet 4 Drive	3.9	6	...	-0.32044373	-2.3631053	-0.1425821	1
5	Hornet Sportabout	3.4	8	...	1.97048540	-0.7524820	-1.1370930	1
6	Valiant	3.3	6	...	-0.06304522	-2.7858979	0.1572161	1
7	Duster 360	2.6	8	...	3.00044911	0.3365345	-0.3632185	3

PCA e Cluster Analysis

```
library(dplyr)
cluster_means <- final %>%  group_by (kmeans.cluster) %>%
  summarise(Efficien = mean(`Consumo (Km/litro)`),
            N_Cilin  = mean(`Numero di cilindri`),
            Cilindr  = mean(`Cilindrata (cm cubici)`),
            CV       = mean(`Cavalli vapore`),
            Rapp_A_P = mean(`Rapporto assale posteriore`),
            Peso     = mean(`Peso (kg)`) ,
            Tempo400m= mean(`Tempo per 400 metri (secondi)`) ,
            Mot_L_V  = mean(`Geometria motore (0=V, 1=in linea)`) ,
            Tras_M_A = mean(`Trasmissione (0=automatica, 1=manuale)`) ,
            N_Marce  = mean(`Numero marce`) ,
            N_Carb   = mean(`Numero carburatori`));

cluster_means <- data.frame(cluster_means); format(round(cluster_means,1), nsmall=1)
```

	kmeans.cluster	Efficien	N_Cilin	Cilindr	CV	Rapp_A_P	Peso	Acc_sec	Mot_L_V	Tras_M_A	N_Marce	N_Carb
1	1	3.2	7.3	4510.4	150.1	3.2	1623.5	29.0	0.4	0.0	3.2	2.5
2	2	3.4	6.8	3661.0	198.8	3.8	1361.0	25.1	0.0	1.0	4.6	5.2
3	3	2.3	8.0	6823.4	228.0	3.2	2125.2	27.2	0.0	0.0	3.0	4.0
4	4	4.8	4.0	1722.9	82.6	4.1	1036.8	30.8	0.9	0.7	4.1	1.5

```
final$kmeans.cluster[final$kmeans.cluster == "1"] <- "Personal luxury cars"
final$kmeans.cluster[final$kmeans.cluster == "2"] <- "Sports cars"
final$kmeans.cluster[final$kmeans.cluster == "3"] <- "Supercars"
final$kmeans.cluster[final$kmeans.cluster == "4"] <- "City cars"
```

PCA e Cluster Analysis

```
plot3 <- ggplot(final,aes(x=Dim.1, y=Dim.2)) +
  geom_text_repel(aes(label = Modello), max.overlaps = 20) +
  theme_classic() +

  geom_point(aes(color = kmeans.cluster), alpha = 0.55, size = 3) +
  geom_hline(yintercept = 0, color = "gray70") +
  geom_vline(xintercept = 0, color = "gray70") +

  stat_ellipse( aes(fill = kmeans.cluster),geom="polygon",level=0.75,alpha=0.2,show.legend=F) +

  xlab("Comp.1 (60%): Fascia") +
  ylab("Comp.2 (24%): Prestazioni") +
  guides(colour=guide_legend(title="Cluster")) +

  ggtitle("Biplot con evidenziati i cluster di appartenenza") +
  theme(
    plot.title = element_text(color="black", size=15, face="bold.italic"),
    axis.title.x = element_text(color="black", size=15, face="bold"),
    axis.title.y = element_text(color="black", size=15, face="bold") ,

    legend.position = c(.10, .90),
    legend.title=element_text(size=15),
    legend.text=element_text(size=15) ,
    legend.box.background = element_rect(colour = "black") ,
    legend.box.margin = margin(t = 1, l = 1)
  )
```

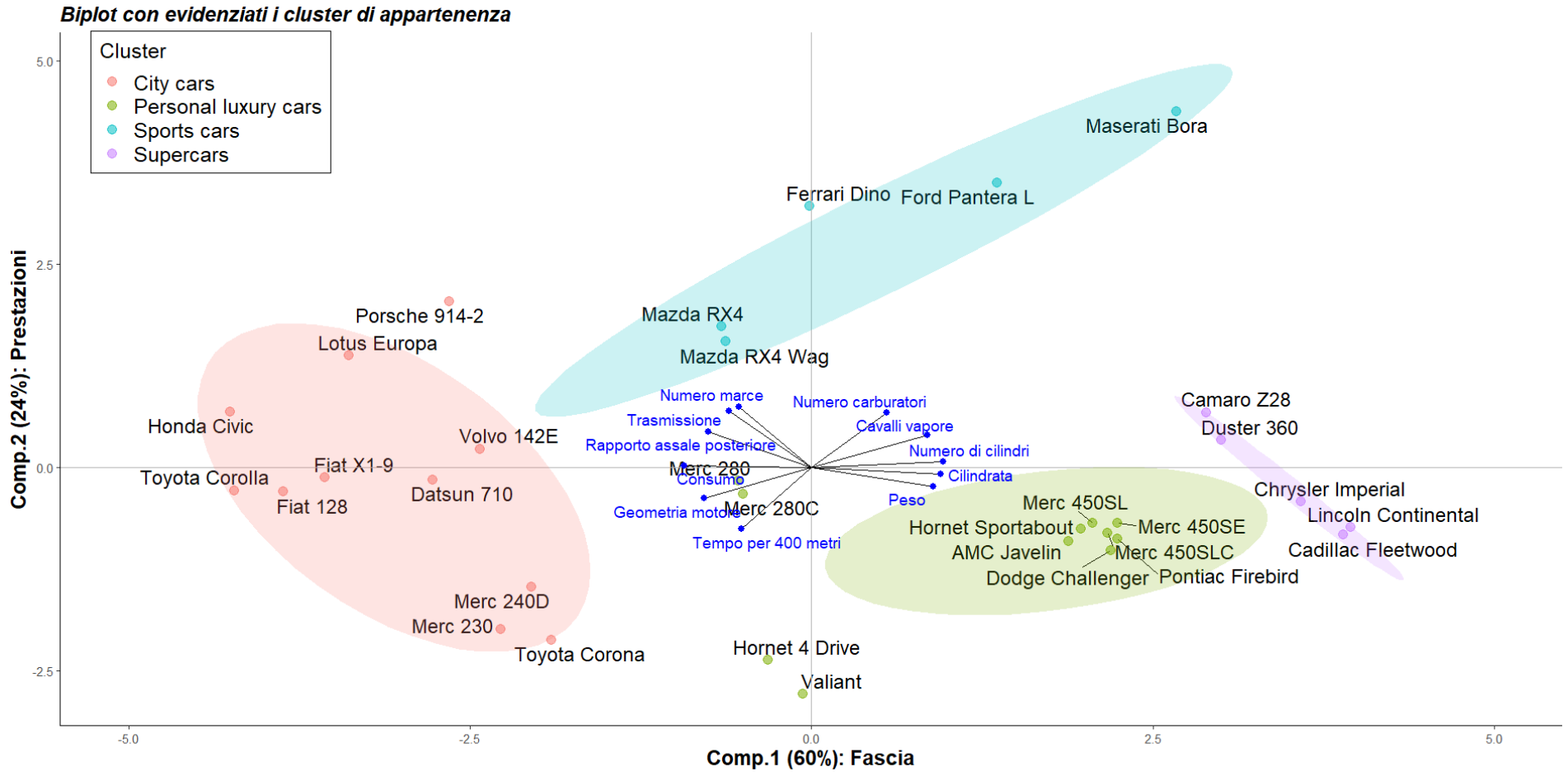
PCA e Cluster Analysis

```
Loadings$labels_short <- sub("\\s*\\(.*", "", Loadings$labels); Loadings$labels_short
```

```
 [1] "Consumo"
 [2] "Numero di cilindri"
 [3] "Cilindrata"
 [4] "Cavalli vapore"
 [5] "Rapporto assale posteriore"
 [6] "Peso"
 [7] "Tempo per 400 metri"
 [8] "Geometria motore"
 [9] "Trasmissione"
[10] "Numero marce"
[11] "Numero carburatori"
```

```
plot4 <- plot3 +
  geom_segment(data=Loadings, aes(xend=Dim.1, yend=Dim.2), x=0,y=0,
    arrow=arrow(type='closed', length=unit(0.02,"picas"))) +
  geom_text_repel(data=Loadings,
    aes(x=Dim.1, y=Dim.2, label=labels_short), size = 4, vjust=0.5, color="blue") +
  geom_point(data=Loadings, aes(x=Dim.1, y=Dim.2), color="blue" ,size=2) +
  labs(caption= "www.alfredoroccato.it" ) + theme(plot.caption = element_text(size=12,
  color="blue"))

plot4
```

PCA e Cluster Analysis



www.alfredoroccato.it

Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

La funzione ggplot

Un ggplot è composto da alcune componenti base:

- **Data** è il dataframe contenente i dati
- **Aesthetics** viene usata per indicare le variabili x e y. Si usa anche per definire il **colore**, la **dimensione** o la **forma** dei punti, l'altezza delle barre, ecc...
- **Scales** scale degli assi (unità di misura e altro)
- **Geometry** definisce il tipo di grafico (**linea, dispersione, barre**, **box plot**, ...)
- **Themes** imposta gli elementi visivi generali (sfondo, griglie, assi, colori e i caratteri)

```
library(ggplot2)
ggplot ([data=] dataframe,
        [mapping=] Aes ( [x=]x variable, [y=]y variable ) ) +
         geom_ () +
         scale_ () [ + ]
       [ geom_line () + ]
       [ geom_bar () + ]
       [ geom_text () + ]
         Theme () [ + ]
```

La funzione ggplot

Tipologie di grafici definite dalle diverse geometrie che possono essere soprapposte semplicemente usando il simbolo +. La prima geometria è sempre il primo strato, ogni geometria aggiunta viene posta sopra:

geom_line()	grafico a linee
geom_point()	grafico a dispersione
geom_histogram()	istogramma
geom_bar ()	diagramma a barre (raggruppate, impilate)
geom_bar()+ coord_polar	diagramma a torta
geom_*boxplot()*	diagramma a scatola e baffi (box plot)
geom_area()	aerogramma
facet_wrap()	sfaccettature (grafici multipli, uno per ogni sottoinsieme dei dati)

La funzione ggplot

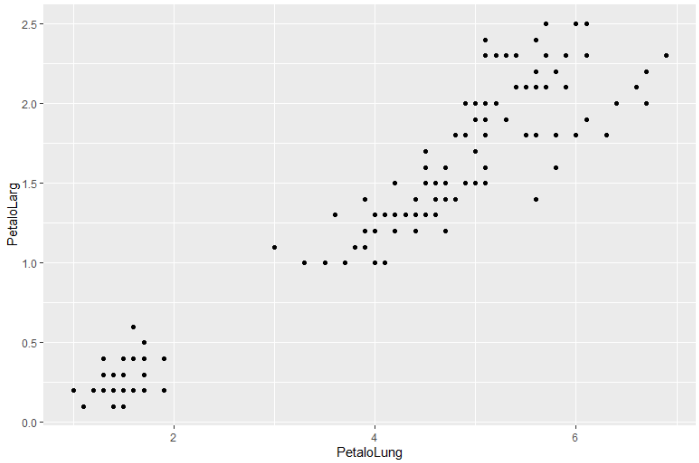
L'estetica permette di modificare l'aspetto dei grafici con colori, forme, dimensioni, ecc.:

color=	colore delle linee, del contorno dei punti
fill=	colore per i riempimenti
size=	dimensione del punto o spessore della linea
linetype=	tipo tratteggio della linea
alpha=	trasparenza del riempimento
labels=	testo sugli assi
shape=	forma dei punti
position=	posizione delle barre (raggruppate, impilate, impilate 100%)
scale=	intervallo dei dati

Grafico dispersione (scatter plot) – 1/3

```
library(ggplot2)
ggplot (Iris, aes(PetaloLung, PetaloLarg)) +
  geom_point()
```

```
g <- ggplot (Iris,
       aes(PetaloLung, PetaloLarg,
         color=Specie,
         shape=Specie)) +
       geom_point(alpha=0.5, size=2.5) +
scale_color_manual(values=c("setosa"="darkviolet",
                            "versicolor"="#0000FF"
                            "virginica"="red")) +
scale_shape_manual(values=c("setosa"=16,
                            "versicolor"=15,
                            "virginica"=17))
g
```



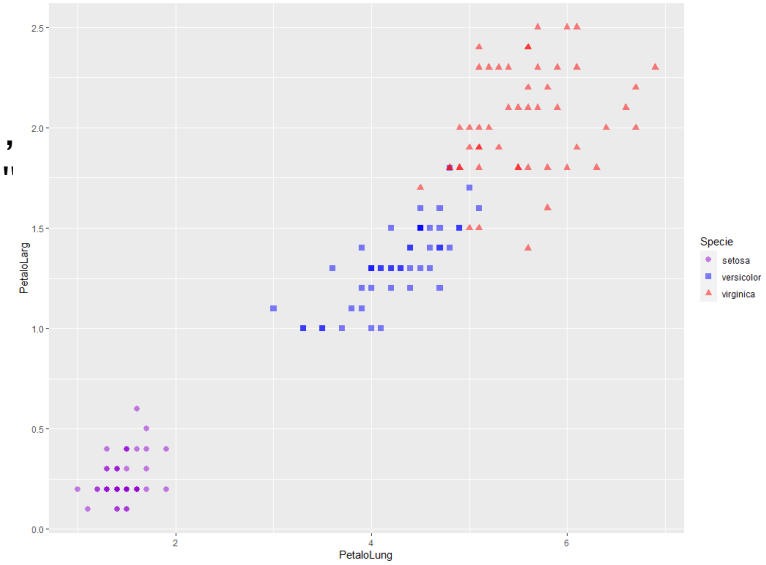
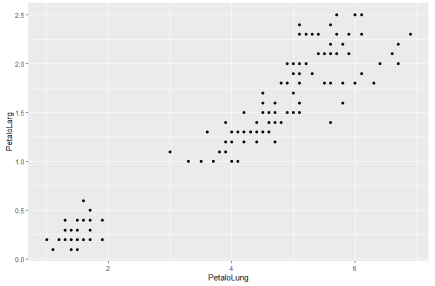


Grafico di dispersione (scatter plot) – 2/3

```
g <- g +
  xlim(0, 7) + ylim(0, 3) +
  ggtitle("Titolo") +
  labs(x="Lunghezza petalo", y="Larghezza petalo",
  shape="Legenda", col="Legenda") +
  theme(
   plot.title       = element_text(color="black",
                                   size=14,
                                   face="bold.italic",
                                   hjust =0.5),
   axis.title.x     = element_text(color="black",
                                   size=12,
                                   face="plain"),
   axis.title.y     = element_text(color="black",
                                   size=12,
                                   face="bold") ,
   panel.background = element_rect(fill = "white",
                                   linetype = "solid",
                                   color = "gray"),
   panel.grid.major = element_line(size = 0.5,
                                   linetype = 2,
                                   color = "gray"),
   panel.grid.minor = element_line(size = 0.25,
                                   linetype = 2,
                                   color = "gray"))
```

Grafico di dispersione (scatter plot) – 3/3



g

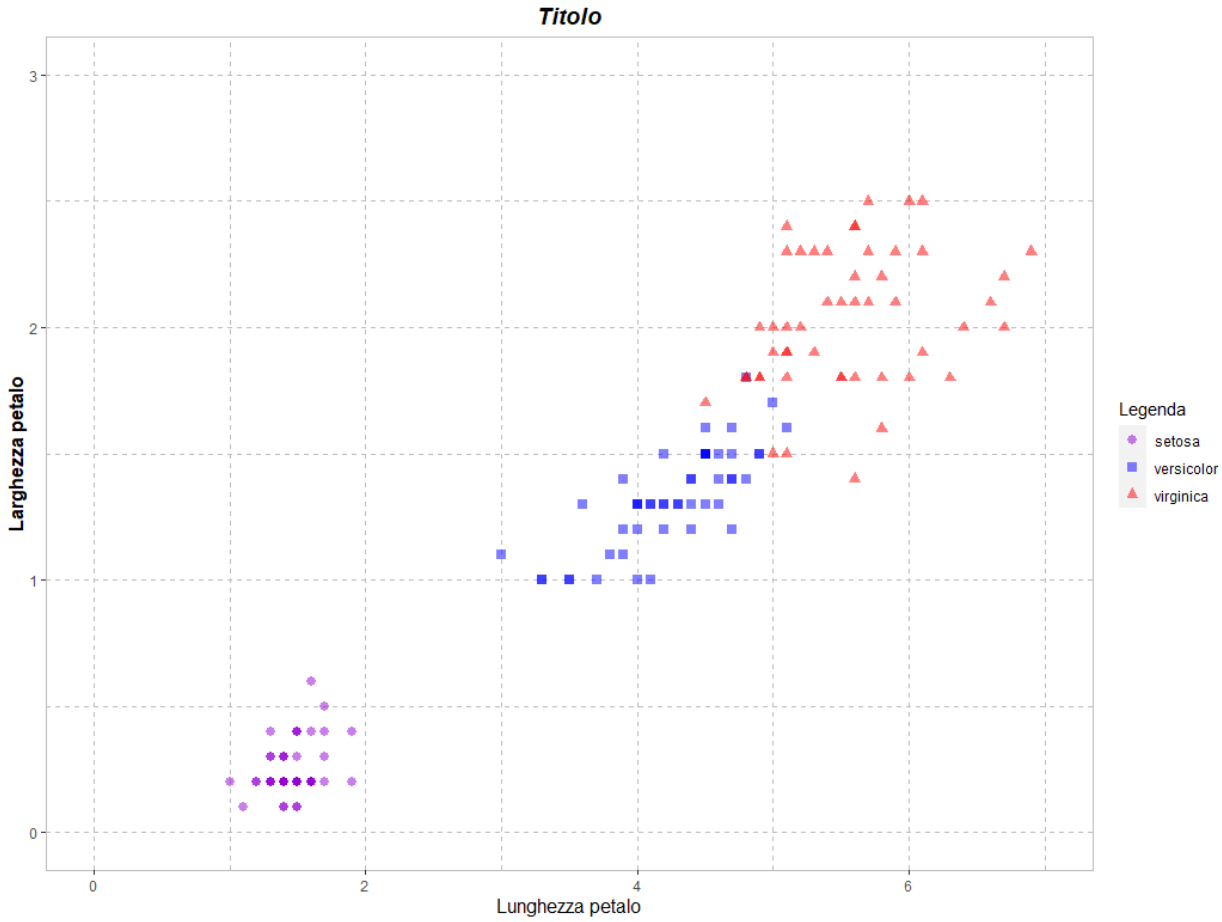


Grafico di dispersione (con etichette, colori, simboli e sottotitolo)

```
corr <- round(cor(Classe$Altezza_cm, Classe$Peso_kg, use = "complete.obs"), 3)

ggplot (Classe,
    aes(Altezza_cm, Peso_kg, color=Genere, shape=Genere)) +
geom_point(size=2, fill="lightgray") +
geom_text(aes(label=Nome),
size=6, alpha=0.5,
vjust=ifelse(Classe$Genere=="F",-0.5, 1.5),
   show.legend=F) +
scale_shape_manual(
   values=c("F"=24, "M"=25)) +
scale_color_manual(
   values=c("F"="darkviolet", "M"="#0000FF")) +
ggtitle("Scatter plot Peso e Altezza",
subtitle = paste("Correlazione: ",
   corr, sep = "")) +
theme_classic() +
theme(plot.title =
   element_text(hjust = 0.5, size=12)) +
theme(plot.subtitle =
   element_text(hjust = 0.5, size=10)) +
theme(legend.position = c(0.1, 0.8))
```

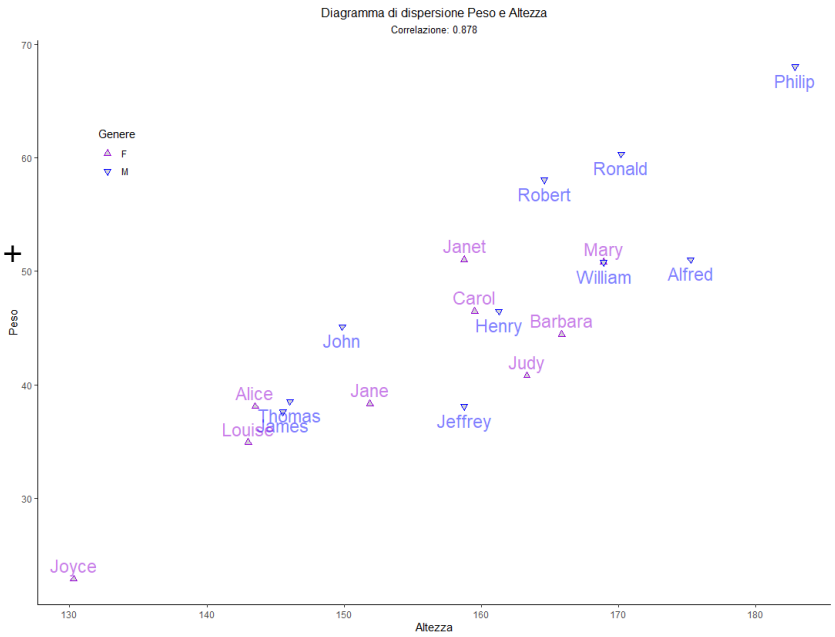


Grafico di dispersione (con retta interpolante)

```
mod <- lm(Classe$Peso_kg ~ Classe$Altezza_cm);
r <- cor(Classe$Peso_kg, Classe$Altezza_cm)

ggplot(data = Classe, aes(x = Altezza_cm, y =  Peso_kg)) +
  geom_point(color = "blue", size=4 ) +
  geom_smooth(method = "lm", formula=y~x, se = F, color= "black" ) +
  labs(title = "Scatterplot of Peso per Altezza",
       x = "Altezza (in cm)", y = "Peso (in kg)" ) +
  annotate("text", x = 135, y = 60, col = "black", size = 6, hjust=0,
       label=paste("Pearson r=",  signif(r))) +
  annotate("text", x = 135, y = 55, col = "black", size = 6, hjust=0,
       label = paste("y = ",      signif(coef(mod)[1],2) , "+",
                                  signif(coef(mod)[2],2), "x",
                     ", R^2= ",   signif(summary(mod)$r.squared,2)))
```

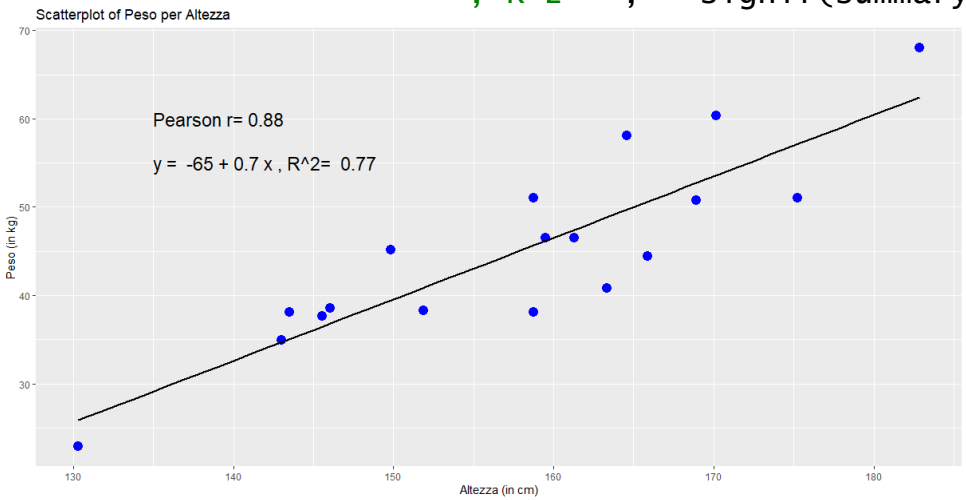


Grafico di dispersione a strati

```
ggplot (Iris,
    aes(PetaloLung, SepaloLung, color=Specie, shape=Specie)) + geom_point() +
geom_point(size=3) +
ggtitle("Titolo") +
scale_color_grey() + theme_classic() + facet_wrap(~Specie)
```

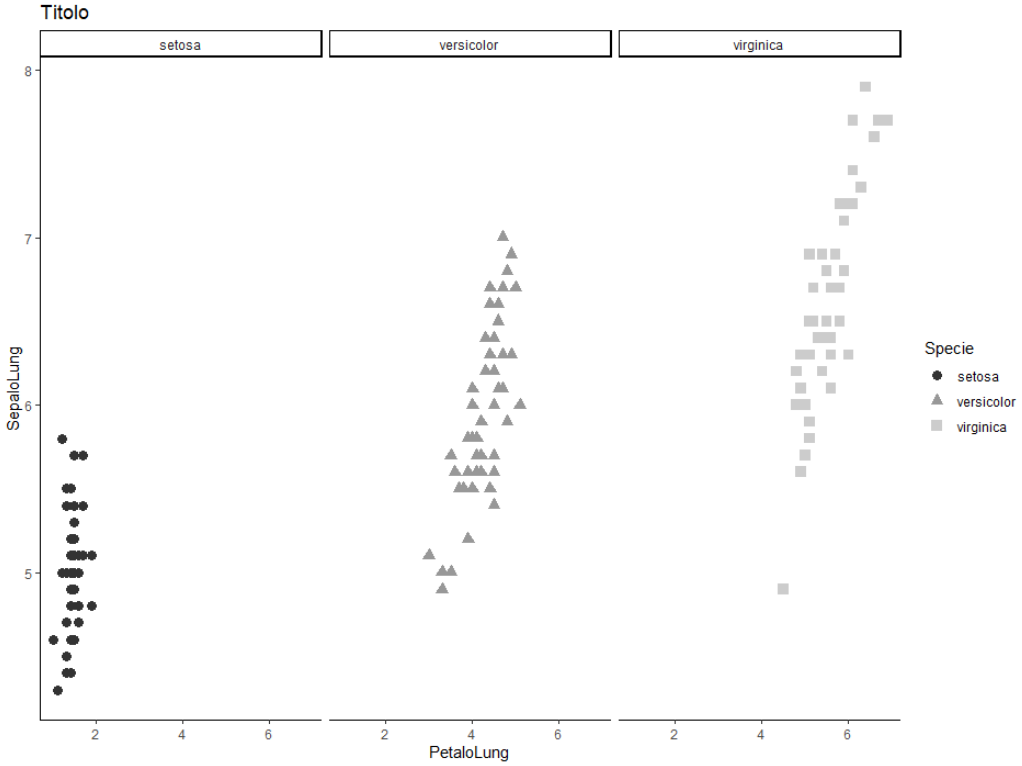
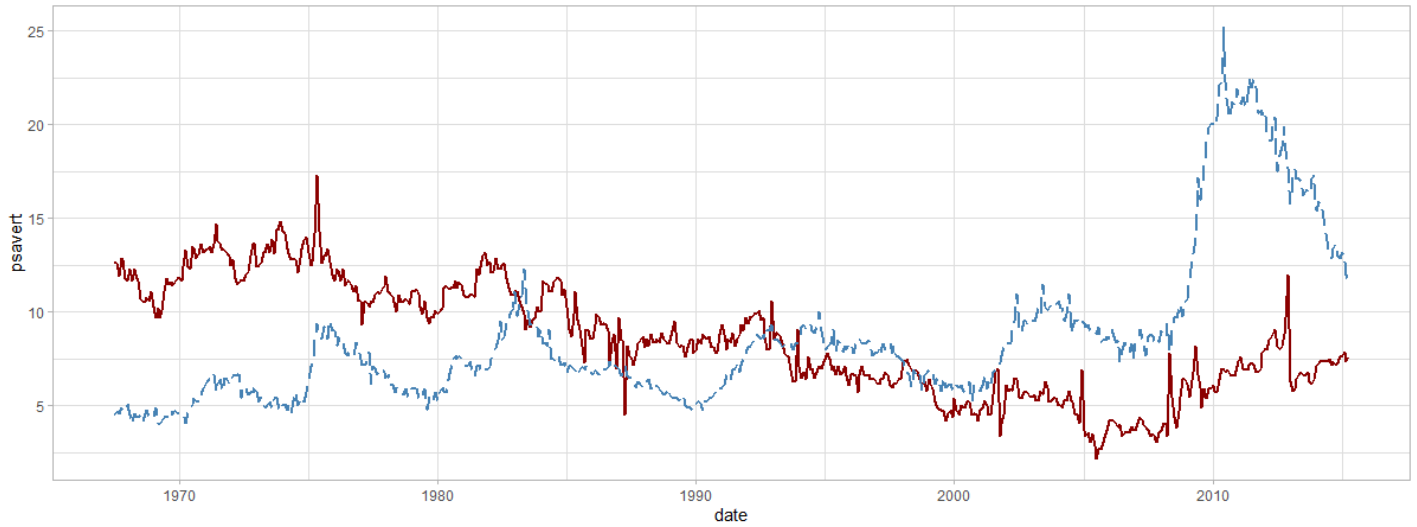


Grafico a linee

```
library(ggplot2)
data(economics)

ggplot (data=economics, aes(x=date)) +
  geom_line(aes(y = psavert), color[1] = "darkred"  , size=1) +
  geom_line(aes(y = uempmed), color  = "steelblue", size=1, linetype[2] = "longdash") +
  theme_light()
```



[1] https://rpubs.com/kylewbrown/r-colors
[2] 0 = blank, 1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash, 6 = twodash

Grafico a scatola e baffi (box and whiskers plot)

```
ggplot (Classe,
    aes(Genere, Peso_kg)) +
geom_boxplot(position="dodge", outlier.colour = "red", outlier.shape = 16) +
stat_boxplot(geom = "errorbar", width = 0.15)
```

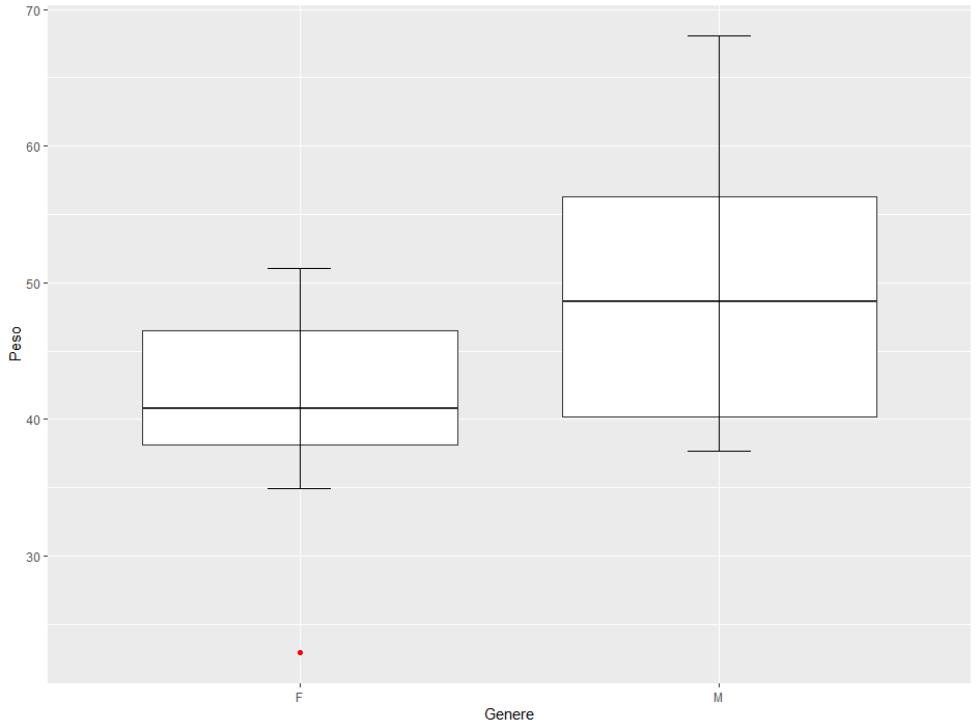
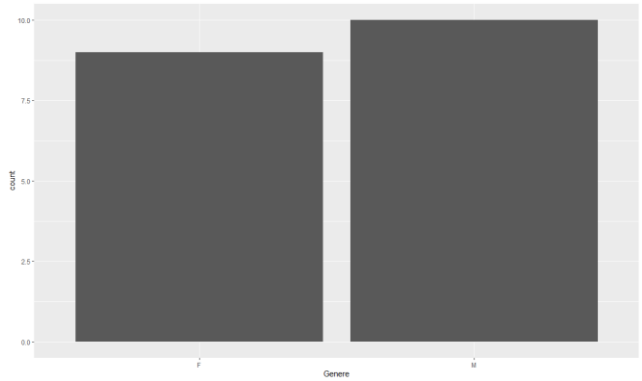


Grafico a barre

```
ggplot (Classe, aes(x=Genere)) +
   geom_bar ()
```



```
ggplot (Classe, aes(x=Genere, fill=Genere)) +
   geom_bar () +
   theme(text = element_text(size=30)) +
   scale_y_continuous(limits=c(0,12),
     n.breaks=7) +
   scale_fill_manual("Genere",
   values = c("M"="#00A9FF","F"="#ED68ED"))
```

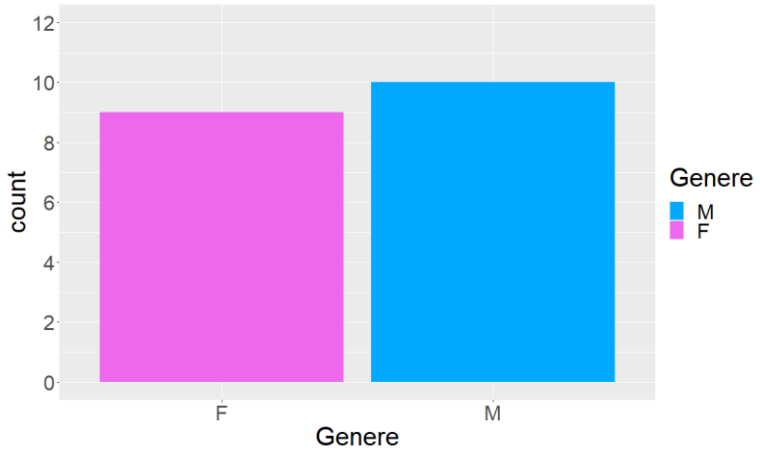
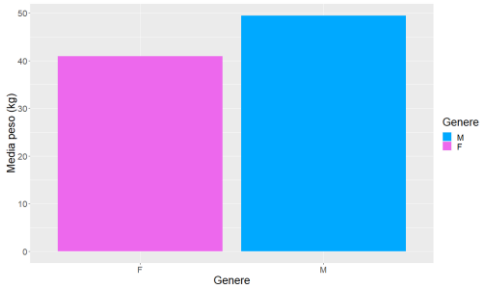
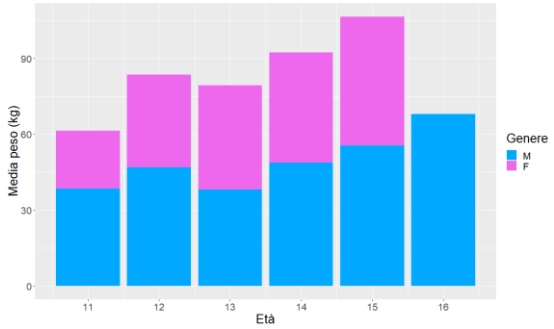


Grafico a barre

```
ggplot (Classe, aes(x=Genere, y=Peso_kg,
          fill=Genere)) +
  geom_bar (stat = "summary", fun = "mean") +
  theme(text = element_text(size=20)) +
  scale_y_continuous(name="Media peso (kg)") +
  scale_fill_manual("Genere",
  values = c("M" = "#00A9FF", "F" = "#ED68ED"))
```




```
ggplot (Classe, aes(x=Età, y=Peso_kg,
        group=Genere, fill=Genere)) +
  geom_bar (stat = "summary", fun = "mean") +
  theme(text = element_text(size=20)) +
  scale_x_continuous (n.breaks=6) +
  scale_y_continuous(name="Media peso (kg)") +
  scale_fill_manual("Genere",
  values = c("M" = "#00A9FF", "F" = "#ED68ED"))
```




```
ggplot (Classe, aes(x=Età, y=Peso_kg,
        group=Genere, fill=Genere)) +
  geom_bar (position="dodge", stat = "summary",
      fun = "mean", color="black") +
  theme_light() + theme(text = element_text(size=30)) +
  scale_x_continuous (n.breaks=6) +
  scale_y_continuous(name="Media peso (kg)",n.breaks=8) +
  scale_fill_manual("Genere",
  values = c("M" = "#00A9FF", "F" = "#ED68ED"))
```

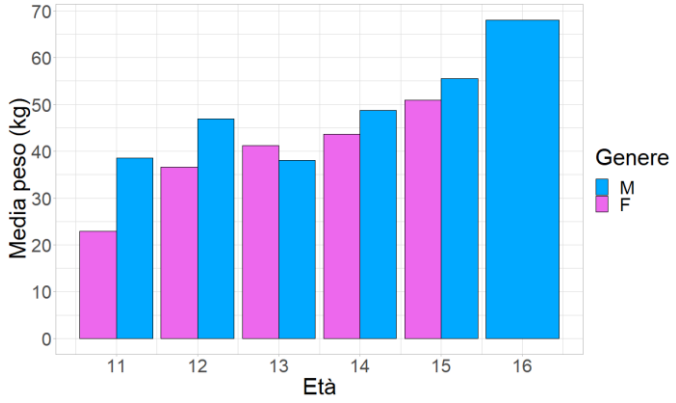


Grafico a barre

```
library(dplyr)
st <- Stipendi %>% group_by (Divisione, Genere) %>%
      summarise( Numero=n(), Somma = sum(Stipendio),
                 Media=mean(Stipendio))

ggplot (st,
        aes(x=Divisione, y=Media, fill=Genere)) +
   geom_bar(position = "dodge", stat="identity") +
   geom_text(aes(label=
     scales::comma(Media, big.mark = ".",
                decimal.mark=",", accuracy=0.1)),
     position=position_dodge(width=1),
     vjust=-0.5, size=4) +
   geom_text(aes(label=Numero),
     position=position_dodge(width=1),
     vjust=2, size=4) +
   scale_color_grey() + theme_classic() +
   ggtitle("Titolo") +
   theme(plot.title= element_text(hjust =0.5)) +
   theme(text = element_text(size=15)) +
   theme(legend.position = c(0.2, 0.9)) +
   scale_y_continuous(name="Media Stipendi",
      limits=c(0,100000)) +
   scale_fill_manual("Genere",
     values = c("M" = "cyan", "F" = "#FF66FF"))
```

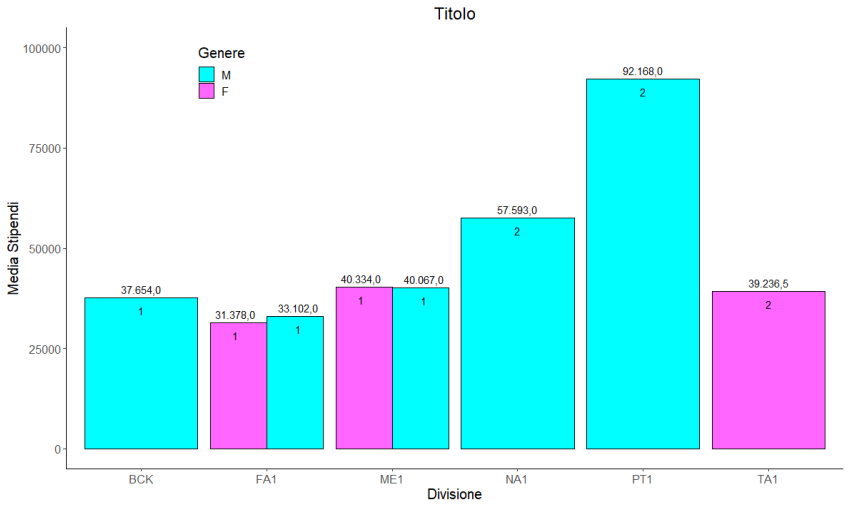


Diagramma a torta

```
library(dplyr)
st <- Stipendi %>% group_by (Genere, Divisione) %>%
   summarise( Numero=n(), Somma = sum(Stipendio),
              Media=mean(Stipendio)) %>% mutate (Pct=Somma/sum(Somma)*100)

ggplot(data = st, aes(x = "",
y = Pct , fill = Divisione)) +
geom_bar(stat = "identity",
   position = position_fill()) +
geom_text(aes(label =
   round(Pct,1)),
   position =
    position_fill(vjust = 0.5)) +
coord_polar(theta = "y") +
theme_void() +
   facet_wrap(~ Genere)  +
   theme(axis.title.x =
element_blank(),
         axis.title.y =
element_blank()) +
   theme(legend.position="bottom") +
   guides(fill=guide_legend(nrow=1, byrow=T))
```

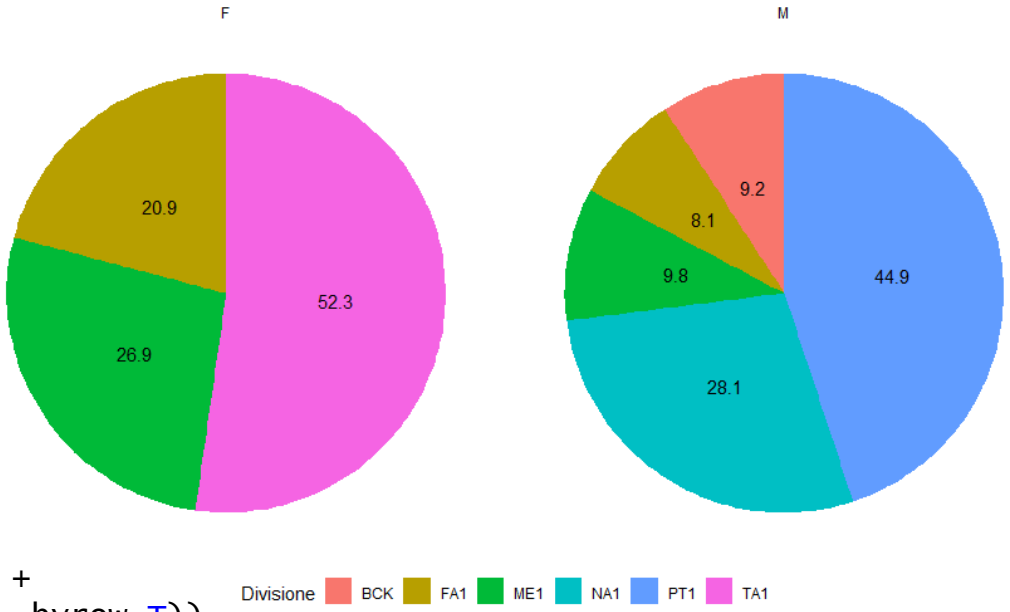


Grafico superficie 3D

```
w1 <- seq(-2,2, length=40)
w2 <- w1
z <- function (w1, w2) { w1*exp(-w1^2-w2^2) }
f <- outer(w1, w2, z)

persp (w1,w2,f,
       theta=-40,     # Ruota su Z
       ltheta=120,
       phi=25,        # Ruota su X
       d = 0.5,
       expand=0.5,
       col= "lightgray" ,
       shade=0.25,
       border = '#555555' ,
       axes=T,
       box=T,
       xlab = NULL,
       ylab = NULL,
       zlab = "f(w1,w2)" )
```

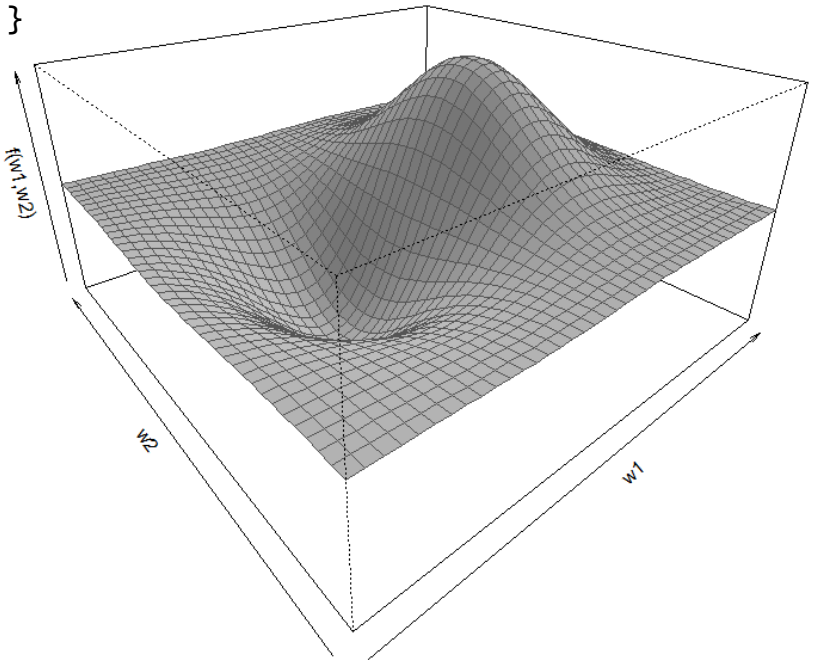
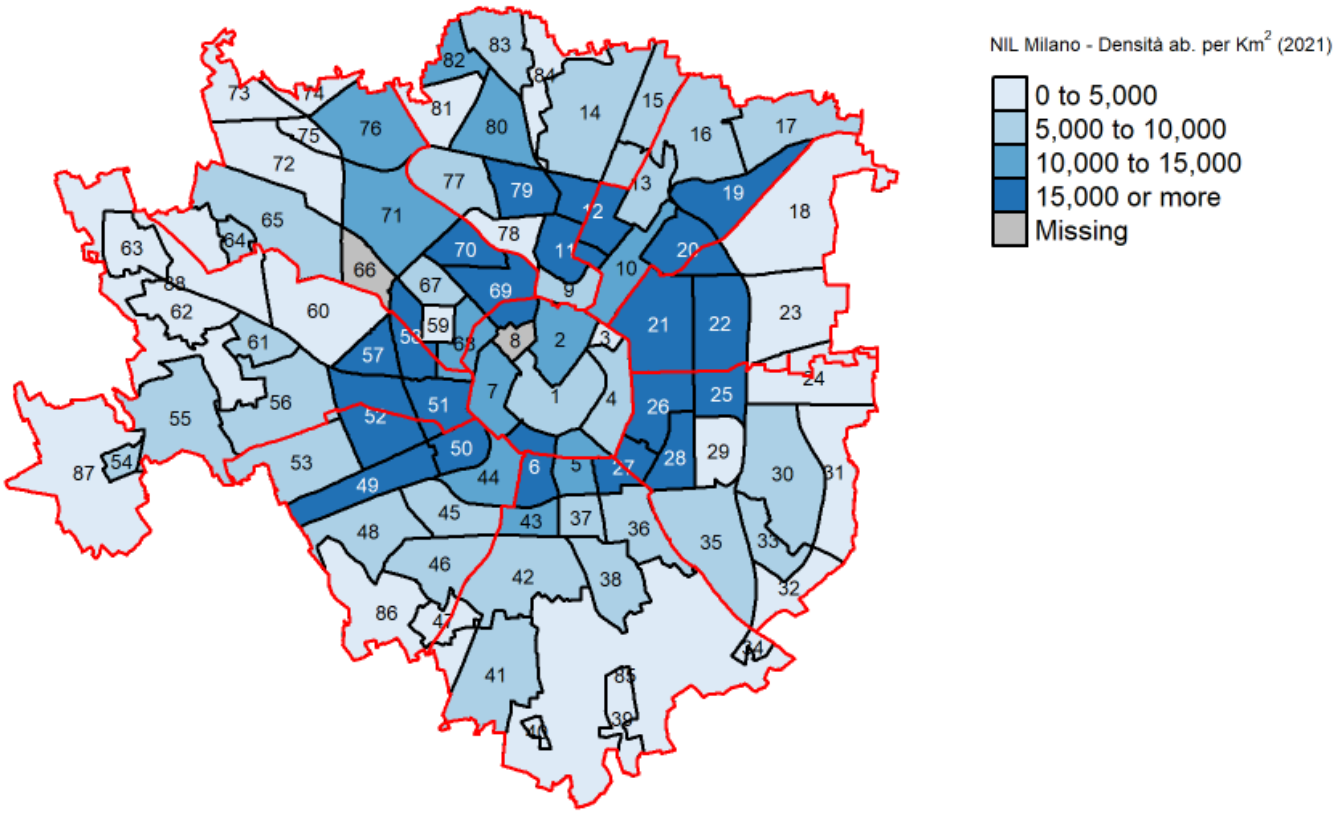


Grafico Mappa tematica

```
library(tmap)
library(sf)

st1 <- st_read("C:/Temp/Geo/Municipi.shp")

st2 <- st_read("C:/Temp/Geo/NIL_WM.shp")

st2 <- st2[,c("ID_NIL","Shape_Area","geometry")]

path <- "C:/Temp/Milano_NIL_Residenti_2021.csv"
Abitanti <- read.csv(path, header=TRUE, sep =";", dec =".");
names(Abitanti)[1] <- "ID_NIL";
Abitanti$ID_NIL <- gsub("[^0-9]","",Abitanti$ID_NIL);

st2 <- merge(x=st2, y=Abitanti, by = "ID_NIL", all.x=TRUE);
st2$Densità_Ab_km2 <- st2$Residenti/st2$Shape_Area*1000000

tm <- tm_shape(st2) +
        tm_layout(frame = FALSE) +
        tm_borders("black", lwd=1.5) +
        tm_text("ID_NIL", size=0.5) +
        tm_polygons("Densità_Ab_km2" , alpha=1, palette="Blues",
         style="fixed", breaks = c(0, 5000, 10000, 15000, Inf),
         title=expression("NIL Milano - Densità ab. per Km"^2* " (2021)")) +
        tm_legend(legend.outside=TRUE) +
        tm_shape(st1) +
        tm_borders("red",lwd=1.5)
```

Grafico Mappa tematica

```
tmap_save(tm,"C:/Temp/Mappa_NIL_Milano_densità_ab.png", width=1920, height=1080, asp=0)
```



Programma

- Introduzione
- Ambiente operativo
- Importazione/esportazione dati
- Strutture
- Trasformazione
- Operatori aritmetici e logici, strutture di controllo
- Funzioni
- Combinare e raggruppare dati
- Statistiche di base
- Modelli lineari
- Machine Learning
- Rappresentazioni grafiche
- Appendice

Appendice

Riferimenti

Sito web https://www.r-project.org/
Libreria moduli aggiuntivi https://cran.r-project.org/

Tutorial

https://www.tutorialspoint.com/r/r_quick_guide.htm
https://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html

Corsi online

https://www.learnbyexample.org/r-introduction/

Bibliografia

https://www.amazon.it/dp/1491910399?tag=guru99it-21&geniuslink=true

https://www.amazon.in/Action-2ed-MANNING-Robert-Kabacoff/dp/9351198073?geniuslink=true&linkCode=sl1&tag=datasciencetu-21&linkId=7996f4d2e0d850acbed44cd8933dccf0&language=en_IN&ref_=as_li_ss_tl

L'esperienza è il miglior maestro

Contatti
alfredo.roccato(at)fastwebnet.it
www.alfredoroccato.it