

01-PRactice

Ottavia M. Epifania, Ph.D

Lezione di Dottorato @Università Cattolica del Sacro Cuore (MI)

13 Giugno 2024

Table of contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics
- 4 Esport graphs
- 5 R for statistical computing
- 6 Classical hypothesis testing in R
- 7 Generalized Linear Models (GLMs)

Table of Contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics
- 4 Esport graphs
- 5 R for statistical computing
- 6 Classical hypothesis testing in R
- 7 Generalized Linear Models (GLMs)

Import csv

```
data = read.csv("data/benessere.csv",  
               header = TRUE,  
               sep = ",", dec = ".")  
head(data)
```

	benessere	stipendio	genere
1	5	1461.0983	m
2	7	1132.3637	f
3	7	1675.9004	m
4	2	328.9587	f
5	6	1370.0146	m
6	5	954.3540	f

Table of Contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics
- 4 Esport graphs
- 5 R for statistical computing
- 6 Classical hypothesis testing in R
- 7 Generalized Linear Models (GLMs)

Data set is in wide format

- Sum across columns → sum scores of the respondents
rowSums() (*rowMean()* for computing the mean)
- Sum across rows (righe) → sum scores of the items
colSums() (*colMean()* for computing the mean)

Well-being

```
library(readxl)
benessere = read_xlsx("data/datiBenessere.xlsx")
```

```
head(benessere,2)
```

```
# A tibble: 2 x 19
```

```
      ID  età genere  frat item1 item2 item3 item4 item5  au1  au
  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     1    16     1     0     1     2     4     3     4     5
2     2    21     1     1     2     2     3     4     3     5
```

```
# i 6 more variables: au5 <dbl>, au6 <dbl>, au7 <dbl>, au8 <dbl>, a
#   au10 <dbl>
```

```
rowSums() :
```

```
rowSums(benessere)
```

```
[1] 65 79 88 74 75 83 78 79 85 87 80 88 83 84 92 1
[19] 79 83 112 105 98 86 94 94 120 108 98 100 107 111 113 1
[37] 100 108 107 126 126 116 116 110 117 117 106 111 122 123 129 1
```

Conditioning according to variable labels

They must present a regular expression (common root):

```
colnames(benessere)
```

```
[1] "ID" "età" "genere" "frat" "item1" "item2"  
"item3" "item4"  
[9] "item5" "au1" "au2" "au3" "au4" "au5" "au6"  
"au7"  
[17] "au8" "au9" "au10"
```

Columns with item → well-being items

Columns with au → self-esteem items

`grep()` e `grepl()`: functions for filtering data according to a regular expression (regex)

```
grep("regex", vector)
```

Same functioning, different results

```
(my_vector = colnames(benessere))
```

```
[1] "ID"      "età"     "genere"  "frat"    "item1"   "item2"   "item3"   "item4"
[9] "item5"   "au1"     "au2"     "au3"     "au4"     "au5"     "au6"     "au7"
[17] "au8"     "au9"     "au10"
```

```
grep()
```

```
grep("au", my_vector)
```

```
[1] 10 11 12 13 14 15 16 17 18 19
```

```
grepl()
```

```
grepl("au", my_vector)
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TR
[13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Compute sum scores

`rowSums()` conditioned on specific columns:

```
cat(rowSums(benessere[, grep("item", colnames(benessere))])[1:15],
```

```
14 14 13 13 17 10 16 16 9 13 14 16 17 12 9 ...
```

Assign to a new column (new variable)

```
benessere$score_ben = rowSums(benessere[, grep("item", colnames(ben
```

summary()

```
babies <- read.table("data/babies.tab")  
summary(babies$peso)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
5.411	7.809	9.429	9.970	11.268	17.343

data set

```
summary(babies)
```

id	genere	peso	al
Length:10	Length:10	Min. : 5.411	Min.
Class :character	Class :character	1st Qu.: 7.809	1st Q
Mode :character	Mode :character	Median : 9.429	Media
		Mean : 9.970	Mean
		3rd Qu.:11.268	3rd Q
		Max. :17.343	Max.

table()

```
table(babies$genere)
```

```
f m
```

```
6 4
```

Contingency tables

```
benessere = read.csv("data/benessere.csv", header = T)
# creo una nuova variabile dicotomica per descrivere
# il livello di benessere
benessere$new_benessere = with(benessere,
                                ifelse(benessere > mean(benessere),
                                         "alto",
                                         "basso"))
with(benessere, table(new_benessere, genere))
```

	genere	
new_benessere	f	m
alto	22	18
basso	32	28

table() and percentages

Single variable

```
(table(babies$genere)/nrow(babies))*100
```

```
  f  m
60 40
```

Multiple variables

```
my_perc = with(benessere, table(new_benessere, genere))
(my_perc = cbind(my_perc, rowSums(my_perc)))
```

```
      f  m
alto  22 18 40
basso 32 28 60
```

```
# ta-dan!  
my_perc/my_perc[,3]
```

	f	m	
alto	0.5500000	0.4500000	1
basso	0.5333333	0.4666667	1

Aggregating

Aggregate a response variable according to grouping variable(s) (e.g., averaging per experimental conditions):

```
# one dependent variable (y) and single grouping variable  
aggregate(y ~ x, data = data, FUN, ...)
```

```
# Multiple response variables, multiple grouping variables  
aggregate(cbind(y1, y2) ~ x1 + x2, data = data, FUN, ...)
```

Aggregating: Examples

```
benessere = read.csv("data/benessereScores.csv", header = T,
head(benessere)
```

	ID	età	genere	frat	item1	item2	item3	item4	item5	au1	au2	au
1	1	16		1	0	1	2	4	3	4	5	4
2	2	21		1	1	2	2	3	4	3	5	4
3	3	28		2	4	2	3	5	1	2	4	4
4	4	15		2	2	3	4	2	2	2	4	5
5	5	23		1	3	4	5	1	5	2	3	2
6	6	31		1	2	2	3	2	1	2	5	1

	au8	au9	au10	score_ben	score_au
1	4	2	4	14	33
2	5	1	5	14	40
3	4	2	4	13	38
4	4	2	4	13	38
5	4	1	3	17	26
6	5	1	5	10	33

Compute the mean of the self esteem according to gender

```
aggregate(score_au ~ genere, data = benessere, mean)
```

	genere	score_au
1	1	33.08750
2	2	32.62857

Compute the mean according of self esteem and well being according to gender

```
aggregate(cbind(score_ben, score_au) ~ genere, data = benessere, mean)
```

	genere	score_ben	score_au
1	1	14.46250	33.08750
2	2	14.41429	32.62857

Your turn!



- Recode `frat` and assign it to a new var into the data frame (`siblings: > 0 siblings → no > > 1+ siblings → yes`)
- Compute the mean of `score_ben` according to `siblings`
- Compute the mean of `score_ben` and `score_au` according to `siblings` and `gender` (assign it to the object `mean_dep`)
- Compute the standard deviation of `score_ben` and `score_au` according to `siblings` and `gender` (assign it to the object `sd_dep`)
- merge `mean_dep` and `sd_dep` and assign the resulting object to `descr`

WARNING!

When using `merge` the column names must be different

Result

```
descr
```

	siblings	genere	mean_score_ben	mean_score_au	sd_score_ben	sd_score_au
1	no	1	13.90909	33.27273	3.250042	4.452734
2	no	2	14.51852	32.11111	3.309315	3.004270
3	yes	1	14.67241	33.01724	3.347625	3.743961
4	yes	2	14.34884	32.95349	3.228472	4.396717

Solution

```
benessere$siblings = ifelse(benessere$frat == 0, "no", "yes")

mean_dep = aggregate(cbind(score_ben, score_au) ~ siblings + genere,
                      data = benessere,
                      mean)
colnames(mean_dep)[3:4] = paste("mean",
                                colnames(mean_dep)[3:4],
                                sep = "_")

sd_dep = aggregate(cbind(score_ben, score_au) ~ siblings + genere,
                   data = benessere,
                   sd)
colnames(sd_dep)[3:4] = paste("sd", colnames(sd_dep)[3:4],
                              sep = "_")

descr = merge(mean_dep, sd_dep)
```

tidyverse()

```
install.packages("tidyverse")  
library(tidyverse)
```

tidyverse()

```
install.packages("tidyverse")
library(tidyverse)
```

\begin{center}

\texttt{\%>\%} (Pipe)

\end{center}

Use the combo shift + ctrl + M

Logic:

```
object \%>\%
  grouping \%>\%
  function
```


Descriptive statistica

```
benessere %>% # object
  group_by(siblings, genere) %>% # groupings
  summarise(m_benessere = mean(score_ben), # functions
            sd_benessere = sd(score_ben),
            m_au = mean(score_au),
            sd_au = sd(score_au))
```

```
# A tibble: 4 x 6
```

```
# Groups:   siblings [2]
```

	siblings	genere	m_benessere	sd_benessere	m_au	sd_au
	<chr>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1	no	1	13.9	3.25	33.3	4.45
2	no	2	14.5	3.31	32.1	3.00
3	yes	1	14.7	3.35	33.0	3.74
4	yes	2	14.3	3.23	33.0	4.40

Your turn!



- Compute minimum, maximum, median of `score_au` and `score_ben` with `tidyverse`
- Import the `babies` data set and compute the descriptive stats of weight and height with `tidyverse`

Table of Contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics**
- 4 Esport graphs
- 5 R for statistical computing
- 6 Classical hypothesis testing in R
- 7 Generalized Linear Models (GLMs)

- Traditional graphics
- Grid graphics & ggplot2

For both:

- High level functions → actually produce the plot
- Low level functions → make it looks better =)

Traditional graphics I

High level functions

```
plot()           # scatter plot, specialized plot methods
boxplot()
hist()           # histogram
qqnorm()         # quantile-quantile plot
barplot()
pie()            # pie chart
pairs()          # scatter plot matrix
persp()          # 3d plot
contour()        # contour plot
coplot()         # conditional plot
interaction.plot()
```

`demo(graphics)` for a guided tour of base graphics!

Traditional graphics II

Low level functions

```
points()      # add points
lines()       # add lines
rect()
polygon()
abline()      # add line with intercept a, slope b
arrows()
text()        # add text in plotting region
mtext()       # add text in margins region
axis()        # customize axes
box()         # box around plot
legend()
```

Plot layout

Each plot is composed of two regions:

- The plotting regions (contains the actual plot)
- The margins region (contain axes and labels)

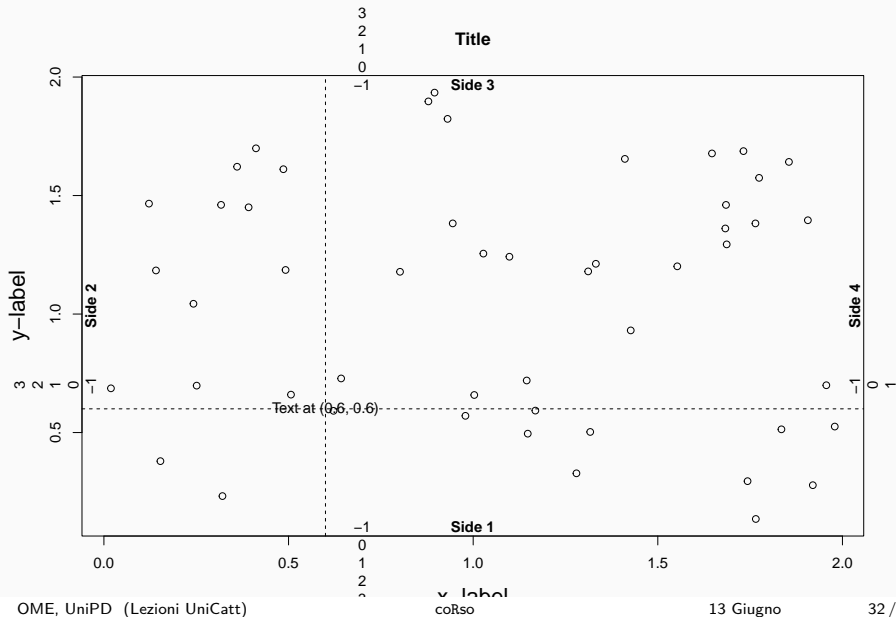
A scatter plot:

```
x <- runif(50, 0, 2) # 50 uniform random numbers
y <- runif(50, 0, 2)
plot(x, y, main="Title",
      sub="Subtitle", xlab="x-label",
      ylab="y-label") # produce plotting window
```

Now add some text:

```
text(0.6, 0.6, "Text at (0.6, 0.6)")
abline(h=.6, v=.6, lty=2) # horizont. and vertic.
                           # lines
```


Margins region



Rome wasn't built in a day and neither your graph

Display the interaction between the two factors of a two-factorial experiment:

```
dat <- read.table(header=TRUE, text="
A B rt
a1 b1 825
a1 b2 792
a1 b3 840
a2 b1 997
a2 b2 902
a2 b3 786
")
```

Force A and B to be factor:

```
dat[,1:2] = lapply(dat[,1:2], as.factor)
```

Modificare il layout dei plot

Vanno creati dei pannelli

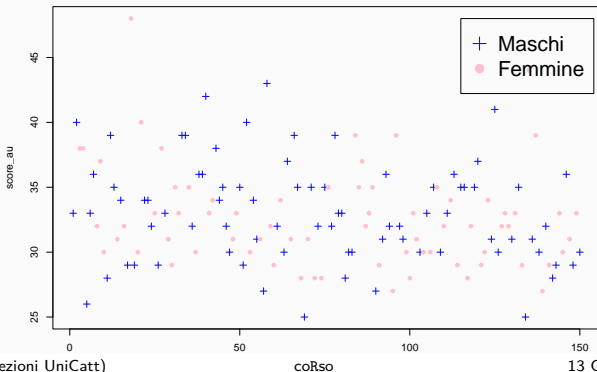
```
par(mfrow=c(nrighe, ncolonne)) # pannelli vengono riempiti per riga  
par(mfcol=c(nrighe, ncolonne)) # pannelli vengono riempiti per colono
```

plot()

```
plot(x) # solo una variabile  
plot(x, y) # due variabile (scatter plot)  
plot(y ~ x) # due variabile, y in funzione di x
```

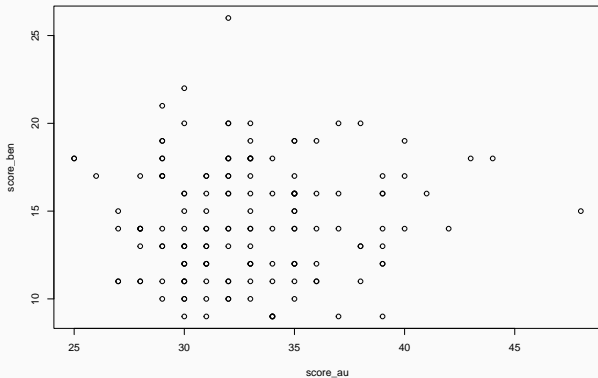
Esempi: plot(x)

```
with(benessere,
  plot(score_au,
    col = ifelse(genere == 1, "blue", "pink"),
    pch = ifelse(genere == 1, 3, 16)))
legend(x = 115, y = 48,
  c("Maschi", "Femmine"), pch = c(3, 16),
  col = c("blue", "pink"), cex = 2)
```



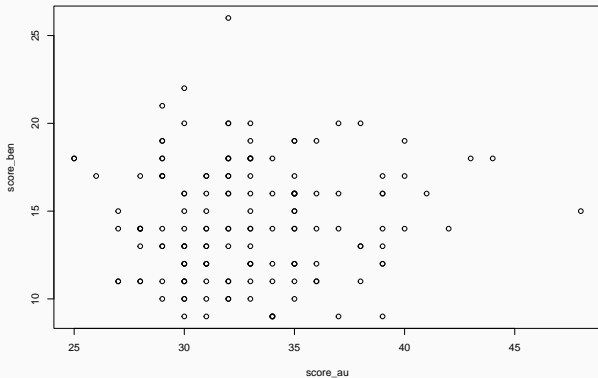
Esempi: plot(x, y)

```
with(benessere,  
     plot(score_au, score_ben))
```



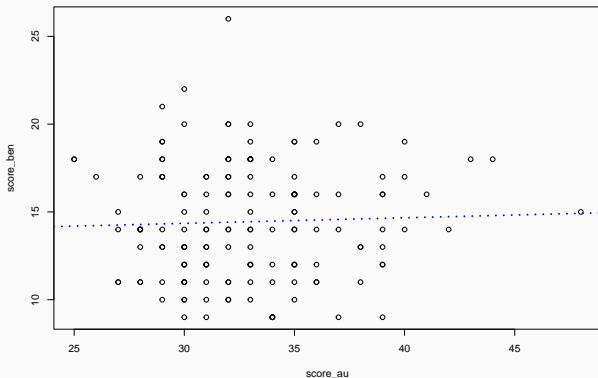
Esempi: `plot(y ~ x)`

```
with(benessere,  
     plot(score_ben ~ score_au))
```



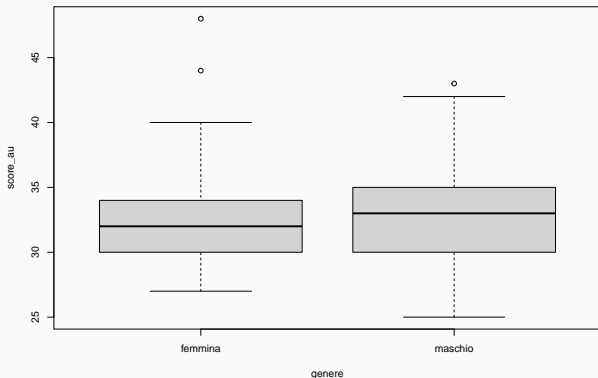
Esempi: `plot(y ~ x)` con retta di regressione

```
with(benessere,
      plot(score_ben ~ score_au))
abline(lm(score_ben ~ score_au, data = benessere),
       col = "blue", lty = 3, lwd = 3)
```



Esempi: `plot(y ~ x)` con `x` categoriale

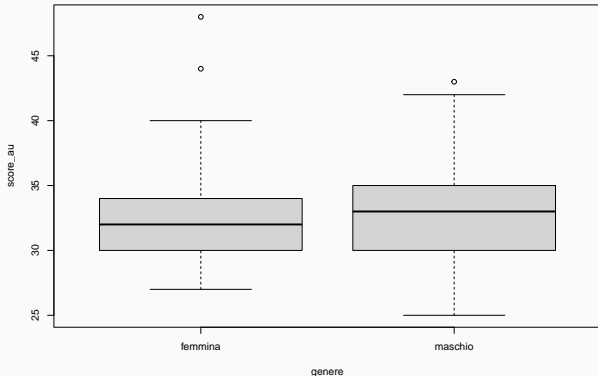
```
benessere$genere <- factor(ifelse(benessere$genere == 1,  
                                "maschio", "femmina"))  
plot(score_au ~ genere, data = benessere)
```



Attenzione!

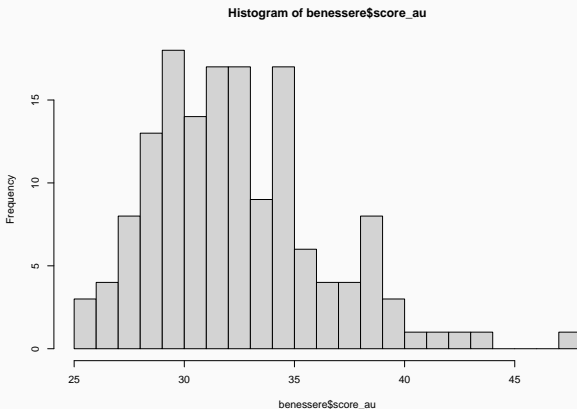
`plot(y ~ x)` con `x` categoriale è uguale a `boxplot(y ~ x)`

```
boxplot(score_au ~ genere, data = benessere)
```



hist(): Frequenze

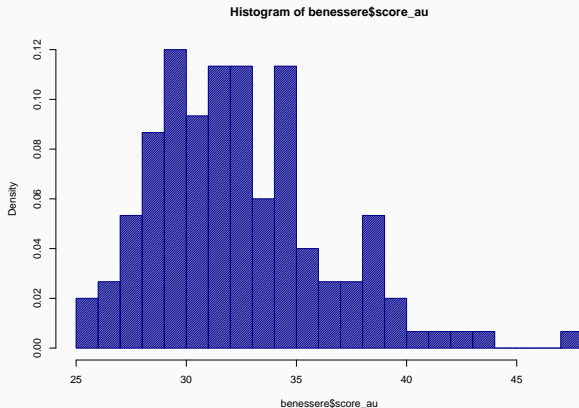
```
hist(benessere$score_au, breaks = 20)
```



hist(): Densità

Densità

```
hist(benessere$score_au,  
     density=50, breaks=20,  
     prob=TRUE, col = "darkblue")
```

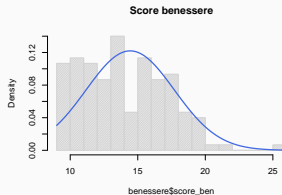
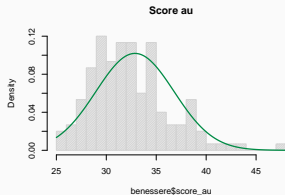


Multi plot (in riga)

```
par(mfrow=c(1, 2))

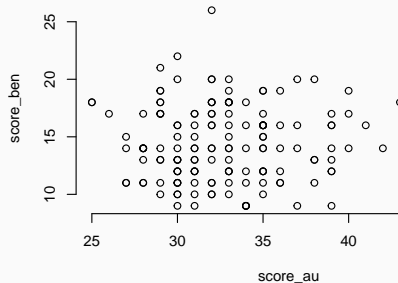
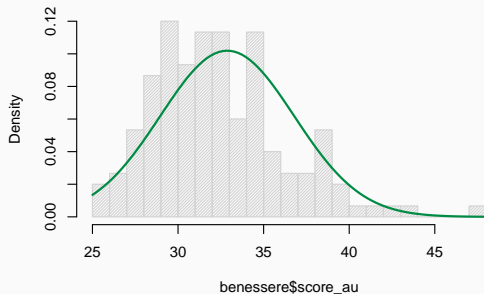
hist(benessere$score_au,density=50, breaks=20, prob=TRUE,
     main = "Score au")
curve(dnorm(x, mean=mean(benessere$score_au),
                  sd=sd(benessere$score_au)),
      col="springgreen4", lwd=2, add=TRUE, yaxt="n")

[...]
```

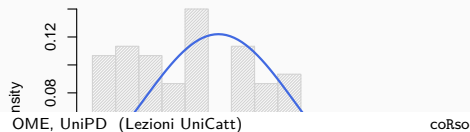


Multiplot (in colonna)

Score au



Score benessere



Multiplot (in colonna), codice

```
par(mfcol= c(2,2))
hist(benessere$score_au,density=50, breaks=20, prob=TRUE,
     main = "Score au")
curve(dnorm(x, mean=mean(benessere$score_au),
                  sd=sd(benessere$score_au)),
      col="springgreen4", lwd=2, add=TRUE, yaxt="n")

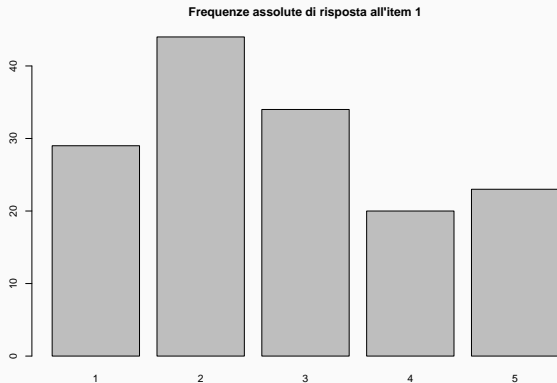
hist(benessere$score_ben,density=50, breaks=20, prob=TRUE,
     main = "Score benessere")
curve(dnorm(x, mean=mean(benessere$score_ben),
                  sd=sd(benessere$score_ben)),
      col="royalblue", lwd=2, add=TRUE, yaxt="n")

with(benessere,
     plot(score_au, score_ben, frame = FALSE))
```

barplot(): Frequenze assolute

Per creare i grafici a barre quando si hanno variabili discrete o categoriali

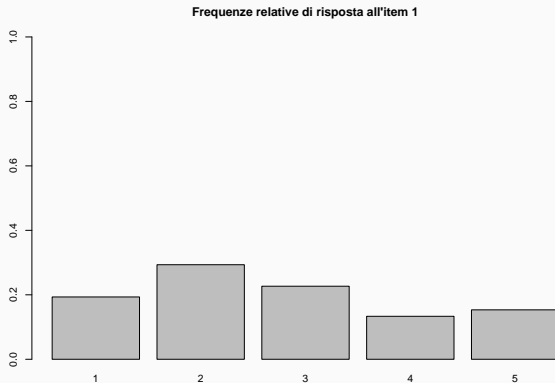
```
freq_item1 = table(benessere$item1)
barplot(freq_item1,
        main = "Frequenze assolute di risposta all'item 1")
```



barplot(): Frequenze relative

Richiede uno step in più → la creazione della tabella delle frequenze

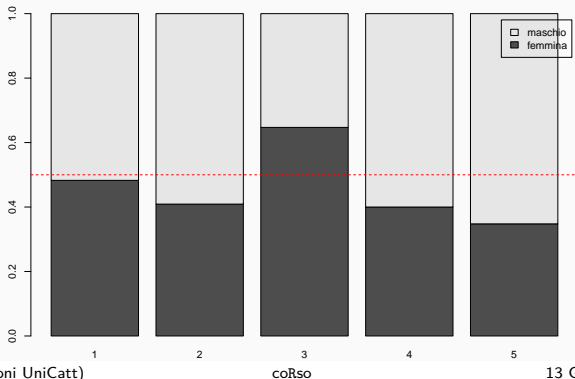
```
perc_item1 = freq_item1/sum(freq_item1)
barplot(perc_item1, ylim = c(0, 1),
        main = "Frequenze relative di risposta all'item 1")
```



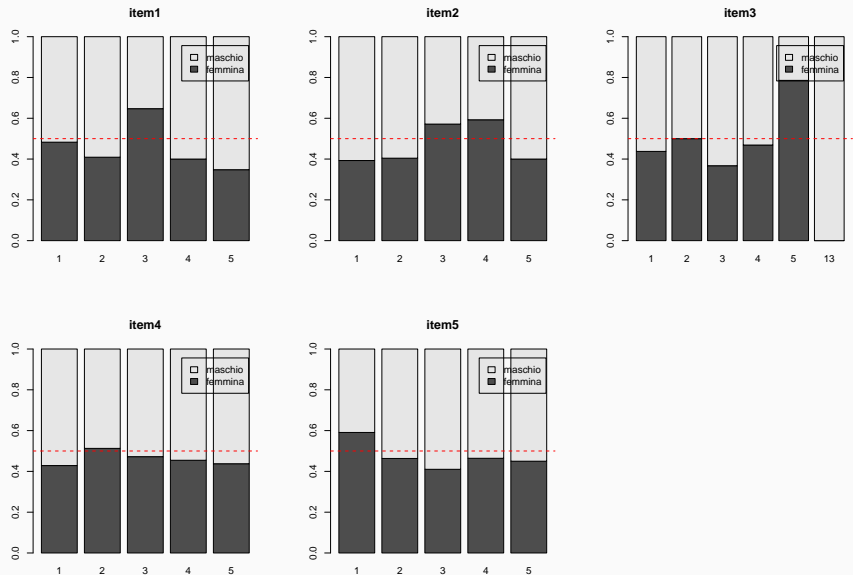
barplot() con più variabili

```
perc_item1_gender = table(benessere$genere, benessere$item1)
perc_item1_gender[1,] = perc_item1_gender[1,]/table(benessere$item1)
perc_item1_gender[2,] = perc_item1_gender[2,]/table(benessere$item1)

barplot(perc_item1_gender, ylim=c(0,1),
        legend = rownames(perc_item1_gender))
abline(h = .5, lty = 2, col = "red")
```



Un altro esempio di multiplot



Un esempio di multiplot (codice)

```
item_ben = benessere[, grep("item", colnames(benessere))]  
  
par(mfrow = c(2, round(ncol(item_ben)/2 + 0.2)))  
temp = NULL  
for (i in 1:ncol(item_ben)) {  
  temp = table(benessere$genere, item_ben[,i])  
  for (j in 1:nrow(temp)) {  
    temp[j,] = temp[j,]/table(item_ben[,i])  
  }  
  barplot(temp, ylim=c(0,1), legend = rownames(temp),  
          main = colnames(item_ben)[i])  
  abline(h = .5, lty = 2, col = "red")  
}
```

interaction.plot()

Permette di vedere l'interazione tra due variabili a seconda di una terza variabile:

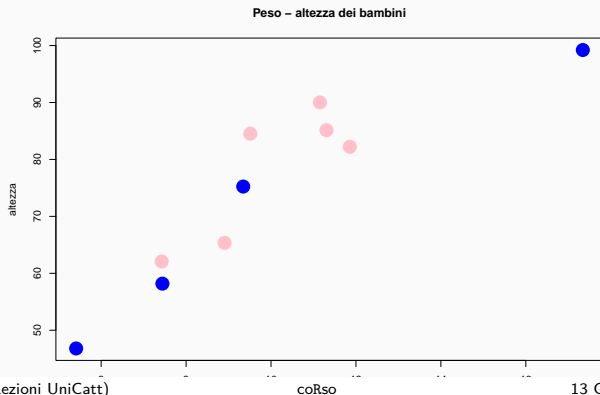
```
interaction.plot(x, v. categoriale, y)
```

interaction.plot()

Permette di vedere l'interazione tra due variabili a seconda di una terza variabile:

```
interaction.plot(x, v. categoriale, y)
```

La relazione tra peso e altezza cambia a seconda del genere?

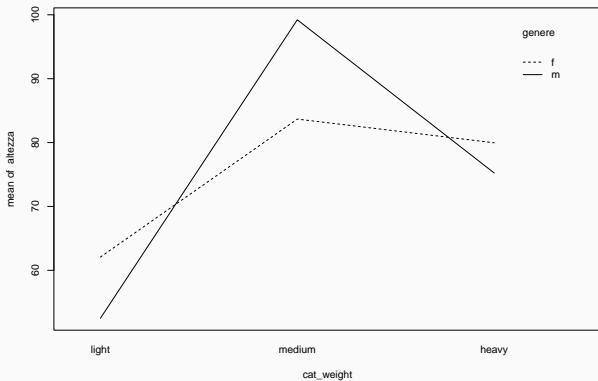


```
babies$cat_weight = with(babies,
  ifelse(
    peso <= quantile(babies$peso)[2],
    "light",
    ifelse(peso > quantile(babies$peso)[2] &
      peso > quantile(babies$peso)[4],
      "medium",
      "heavy")))
babies$cat_weight = factor(babies$cat_weight,
  levels = c("light",
    "medium",
    "heavy"))
```

```
babies
```

	id	genere	peso	altezza	cat_weight
1	baby1	f	7.424646	62.07722	light
2	baby2	m	7.442727	58.18877	light
3	baby3	f	9.512598	84.52737	heavy
4	baby4	f	11.306349	85.13573	medium
5	baby5	m	9.345165	75.23783	heavy
6	baby6	m	5.411290	46.80163	light

```
with(babies,  
      interaction.plot(cat_weight, genere, altezza))
```



ggplot2

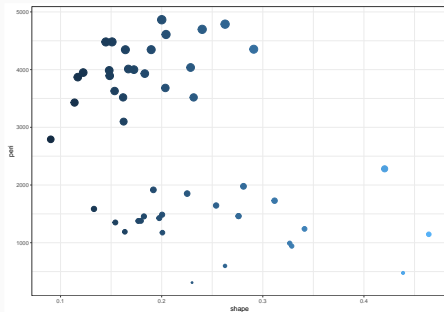
ggplot2 (Grammar of Graphics plot, Wickman, 2016) is one of the best packages for plotting raw data and results:

```
install.packages("ggplot2") ; library(ggplot2)
```

```
ggplot(data,  
  ars(x = var.x,  
    y = var.y,  
    col = var.color, # factor or character  
    fill = var.filling, # factor or character  
    shape = var.shape, # actor or character  
    size = var.size, # numeric  
    ...)) + geom_graph.type() + ...
```

Raw data

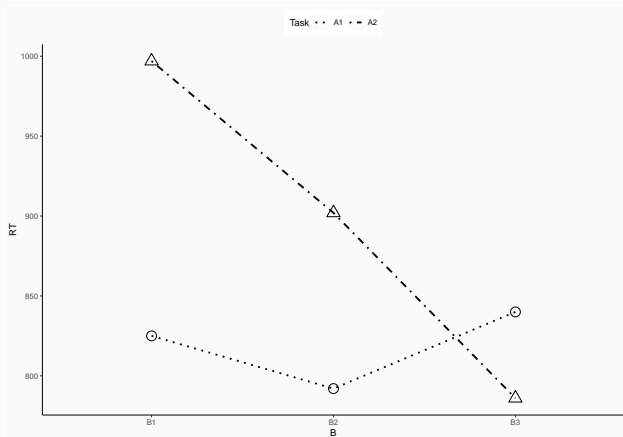
```
ggplot(rock,  
  aes(y=peri,x=shape, color =shape,  
    size = peri)) + geom_point() +  
  theme_bw() + theme(legend.position = "none")
```



The code for the interaction plot

```
ggplot(dat, aes(x = B, y = rt, group = A)) +  
  geom_point(pch=dat$A, size = 5) +  
  geom_line(aes(linetype=A), size=1) + theme_classic() +  
  ylab("RT") + scale_linetype_manual("Task", values =c(3,4),  
                                     labels = c("A1", "A2")) +  
  scale_x_discrete(labels = c("B1", "B2", "B3")) +  
  theme(legend.position="top",  
        panel.background = element_rect(fill = "#FAFAFA",  
                                          colour = "#FAFAFA"),  
        plot.background = element_rect(fill = "#FAFAFA"),  
        legend.key = element_rect(fill = "#FAFAFA"))
```

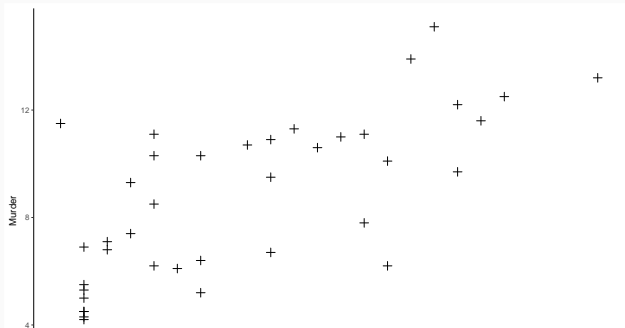
The result



Scatter plot

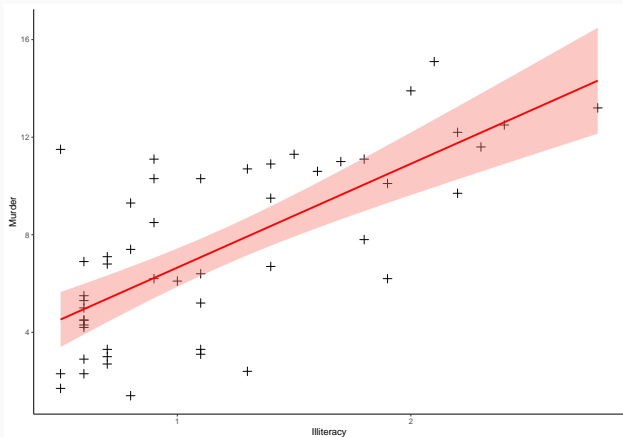
```
states = data.frame(state.x77)
ggplot(states, # raw data
        aes(x = Illiteracy, y = Murder)) + geom_point()
```

```
states = data.frame(state.x77)
ggplot(states, # raw data
        aes(x = Illiteracy, y = Murder)) + geom_point()
```



Linear model

```
ggplot(states, # raw data
  aes(x = Illiteracy, y = Murder)) + geom_point(
  geom_smooth(method="lm", color="red", aes(fill="red")) + th
```



Different plots in the same panel

use `grid.arrange()` function from the `gridExtra` package:

```
install.packages("gridExtra") ; library(gridExtra)
```

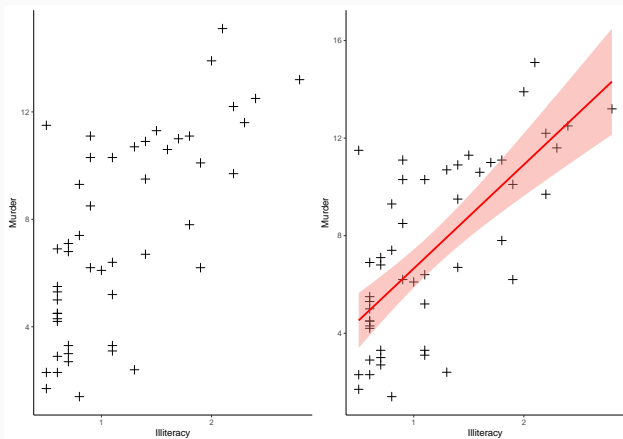
```
murder_raw = ggplot(states, # raw data
                    aes(x = Illiteracy, y = Murder)) +
  ....
```

```
murder_lm = ggplot(states, # lm
                   aes(x = Illiteracy, y = Murder)) +
  ....
```

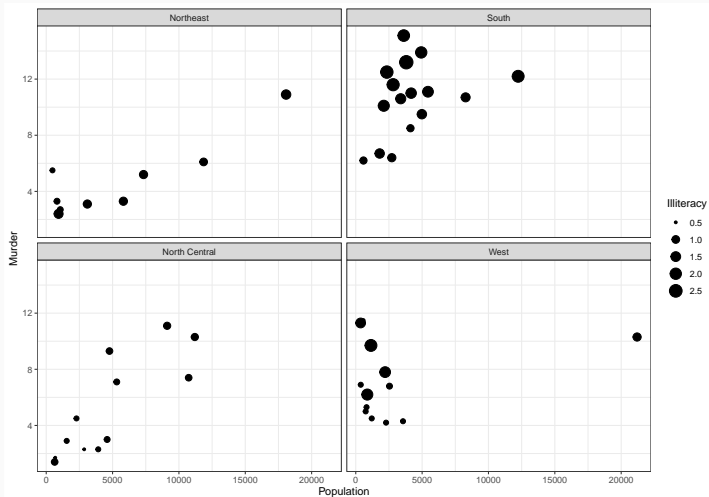
Combine the plots together:

```
grid.arrange(murder_raw, murder_lm,
              nrow=1) # plots forced to be the same row
```

Combine the plots together



Multi Panel



Multi panel code

```
states = data.frame(state.x77, state.name = state.name,  
                     state.region = state.region)  
  
ggplot(states,  
        aes(x = Population, y = Murder,  
            size = Illiteracy)) + geom_point() +  
  facet_wrap(~state.region) + theme_bw()
```

boxplot() e violinplot()

Data need to be in long format:

	id	condition	mean_time
1	sbj1	A	3.477760
2	sbj1	B	1.748681
3	sbj2	A	2.405326
4	sbj2	B	2.915242
5	sbj3	A	3.294477
6	sbj3	B	2.763332

```

small = benessere[, c("ID", "score_au", "score_ben")]
score_long = reshape(small,
  idvar = "ID",
  times = names(small)[-1],
  timevar = "score", v.names = "value",
  varying = list(names(small)[-1]),
  direction = "long")
head(score_long)

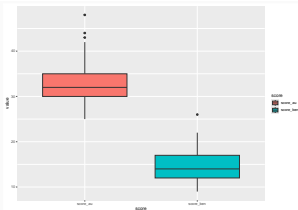
```

	ID	score	value
1.score_au	1	score_au	33
2.score_au	2	score_au	40
3.score_au	3	score_au	38
4.score_au	4	score_au	38
5.score_au	5	score_au	26
6.score_au	6	score_au	33

boxplot vs violinplot

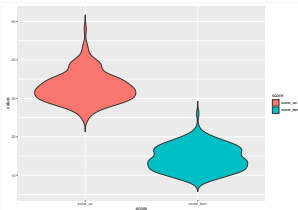
Boxplot

```
ggplot(score_long,
  aes(x = score, y = value,
    fill = score)) +
  geom_boxplot()
```



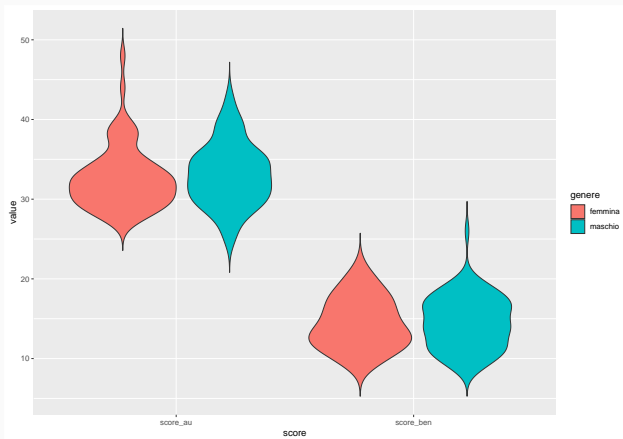
Violinplot

```
ggplot(score_long,
  aes(x = score, y = value,
    fill = score)) +
  geom_violin(trim = FALSE)
```



```
score_long = merge(score_long, benessere[, c("ID", "genere")])

ggplot(score_long,
       aes(x = score, y = value,
           fill = genere)) + geom_violin(trim = FALSE)
```



barplot and histogram

`geom_hist()`: histogram (continuous variables)

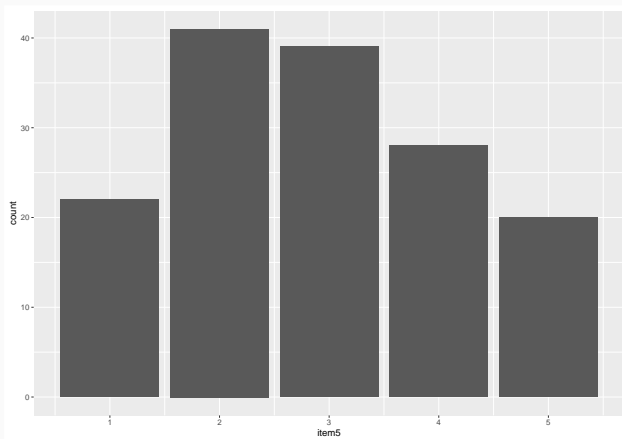
`geom_bar()`: bar plot

Arguments:

- `geom_bar(stat = "count")`: automatically counts the frequencies for each category, **does not need a y variable**
- `geom_bar(stat = "identity")`: plots a value associated to each category **does need a y variable**

geom_bar(stat = "count"):

```
ggplot(benessere,  
       aes(x = item5)) + geom_bar(stat = "count")
```



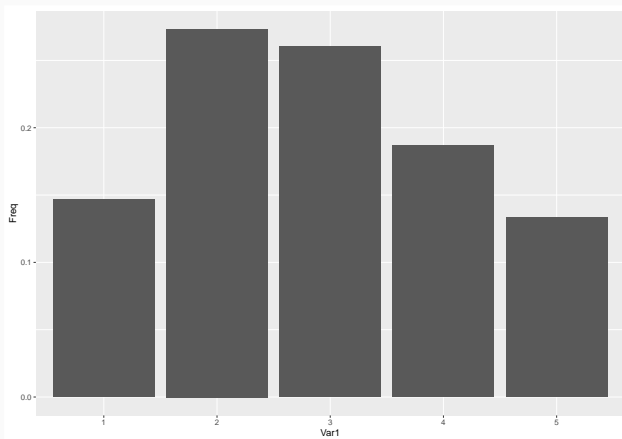
geom_bar(stat = "identity")

```
item_5 = data.frame(table(benessere$item5)/nrow(benessere))  
item_5
```

	Var1	Freq
1	1	0.1466667
2	2	0.2733333
3	3	0.2600000
4	4	0.1866667
5	5	0.1333333

Now the plot

```
ggplot(item_5,  
  aes(x = Var1, y = Freq)) + geom_bar(stat = "identity")
```



All together

Long format

```
new_item = benessere[, c(grep("ID", colnames(benessere)),
                        grep("item", colnames(benessere)))]
new_item = reshape(new_item,
                    idvar = "ID",
                    times =names(new_item)[-1],
                    timevar = "item", v.names = "value",
                    varying = list(names(new_item)[-1]),
                    direction = "long")
new_item
```

	ID	item	value
1.item1	1	item1	1
2.item1	2	item1	2
3.item1	3	item1	2
4.item1	4	item1	3
5.item1	5	item1	4
....			

Calcolo le proporzioni

Di ogni opzione di risposta value per ogni item item

```
proporzione = new_item %>%  
  group_by(item, value) %>%  
  summarise(prop = n()/nrow(benessere))
```

proporzione

```
# A tibble: 26 x 3  
# Groups:   item [5]  
   item value prop  
  <chr> <int> <dbl>  
1 item1     1 0.193  
2 item1     2 0.293  
....
```

The plot (finally)

```
ggplot(proporzione,  
       aes(x = value, y = prop)) +  
  geom_bar(stat = "identity") +  
  facet_wrap(~item)
```

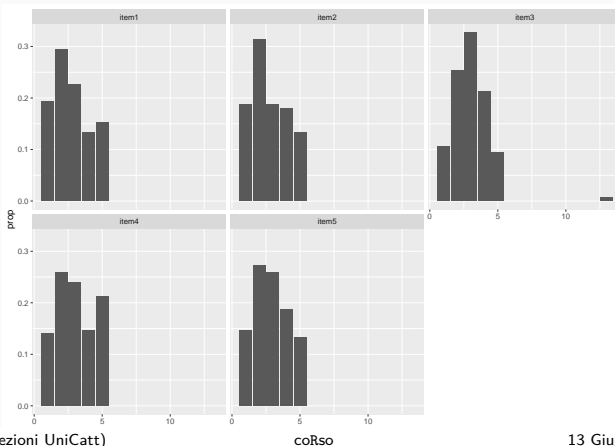


Table of Contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics
- 4 Esport graphs**
- 5 R for statistical computing
- 6 Classical hypothesis testing in R
- 7 Generalized Linear Models (GLMs)

In automatico

```
pdf("nome_grafico.pdf")  
png("nome_grafico.png")  
tiff("nome_grafico.tiff")  
jpeg("nome_grafico.jpeg")  
bmp("nome_grafico.bmp")
```

Remember to turn off the graphical device

```
dev.off()
```

Example

```
pdf("il_mio_violin.pdf")
ggplot(score_long,
       aes(x = score, y = value,
           fill = score)) + geom_violin(trim = FALSE)
dev.off()
```


Manually

```
knitr::include_graphics("data/esporta.png")
```

Table of Contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics
- 4 Esport graphs
- 5 R for statistical computing**
- 6 Classical hypothesis testing in R
- 7 Generalized Linear Models (GLMs)

The `stats` package (built-in package in R) contains function for statistical calculations and random number generator

see `library(help=stats)`

Table of Contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics
- 4 Esport graphs
- 5 R for statistical computing
- 6 Classical hypothesis testing in R**
- 7 Generalized Linear Models (GLMs)

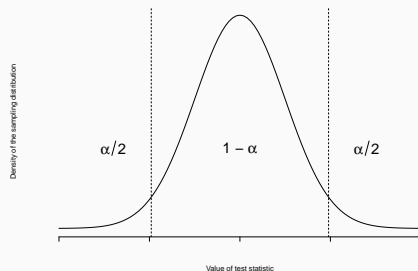
- `chisq.test()`: contingency table χ^2 tests
- `cor.test()`: association between paired samples
- `t.test()`: one- and two-sample t tests (also for paired data)
- `lm(y ~ x1 + x2 + x3 ...)`: fit a linear model
- `glm(y ~ x1 + x2 + x3 ...)`: fit a generalize linear model

What is the p -value?

p -value:

conditional probability of obtaining a test statistic that is at least as extreme as the one observed, given that the null hypothesis is true

If $p < \alpha$ (i.e., the probability of rejecting the null hypothesis when it is true) \rightarrow the null hypothesis is rejected



χ^2 test

Independence of observations

Hypothesis:

- $H_0: P(X_{ij} = k) = p_k$ for all $i = 1, \dots, r$ and $j = 1, \dots, c$
- $H_0: P(X_{ij} = k) \neq P(X_{i'j} = k)$ for *at least* one $i \in \{1, \dots, r\}$ and $j \in \{1, \dots, c\}$

Test statistic:

$$\chi^2 = \sum_{i=1}^n \frac{(x_{ij} - \hat{x}_{ij})^2}{\hat{x}_{ij}}, \quad \chi^2 \sim \chi^2(r-1)(c-1)$$

In R:

```
tab <- xtabs(~ education + induced, infert)
chisq.test(tab)
```

Correlation test

Hypothesis:

- $H_0: \rho_{XY} = 0, H_1: \rho_{xy} \neq 0$
- $H_0: \rho_{XY} = 0, H_1: \rho_{xy} < 0$
- $H_0: \rho_{XY} = 0, H_1: \rho_{xy} > 0$

Test statistic:

$$T = \frac{r_{xy}}{\sqrt{1 - r_{xy}^2}} \sqrt{n - 2}, \quad T \sim t(n - 2)$$

In R:

```
cor.test(~ speed + dist, cars,  
         alternative = "two.sided")
```


Correlation matrix

```
cor(benessere[, grep("item", colnames(benessere))])
```

	item1	item2	item3	item4	it
item1	1.00000000	0.07820461	0.11579134	-0.05894317	-0.18117
item2	0.07820461	1.00000000	0.24890831	0.21963611	-0.07757
item3	0.11579134	0.24890831	1.00000000	0.04977145	0.03830
item4	-0.05894317	0.21963611	0.04977145	1.00000000	0.05751
item5	-0.18117877	-0.07757221	0.03830005	0.05751880	1.00000

Two (independent) sample t test

Independent samples from normally distributions where σ^2 are unknown but homogeneous

- $H_0: \mu_{x_1-x_2} = 0, H_1: \mu_{x_1-x_2} \neq 0$
- $H_0: \mu_{x_1-x_2} = 0, H_1: \mu_{x_1-x_2} < 0$
- $H_0: \mu_{x_1-x_2} = 0, H_1: \mu_{x_1-x_2} > 0$

Test statistic:

$$T = \frac{\bar{x}_1 - \bar{x}_2}{\sigma_{\bar{x}_1 - \bar{y}_2}}, \quad T \sim t(n_1 + n_2 - 2)$$

R function:

```
t.test(len ~ supp, data = ToothGrowth,
       var.equal = FALSE)
```

Two (depended) sample t test

Observations on the same sample

Hypothesis:

- $H_0: \mu_D = 0, H_1: \mu_D \neq 0$
- $H_0: \mu_D = 0, H_1: \mu_D < 0$
- $H_0: \mu_D = 0, H_1: \mu_D > 0$

Test statistic:

$$T = \frac{d}{\sigma_d}, \quad T \sim t(m-1)$$

R function:

```
with(sleep,
      t.test(extra[group == 1],
              extra[group == 2], paired = TRUE))
```

Table of Contents

- 1 Import data
- 2 Compute sum scores
- 3 Graphics
- 4 Esport graphs
- 5 R for statistical computing
- 6 Classical hypothesis testing in R
- 7 Generalized Linear Models (GLMs)**

Formulae

Statistical models are represented by formulae which are extremely close to the actual statistical notation:

in R	Model
$y \sim 1 + x$	$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$
$y \sim x$	(same but short)
$y \sim 0 + x$	$y_i = \beta_1 x_i + \varepsilon_i$
$y \sim x_A * x_B$	$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_j + (\beta_1 \beta_2) x_{ij} + \varepsilon_{ij}$

Linear models

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \varepsilon$$

In R:

```
lm(y ~ x1 + x2 + ... + xk, data)
```

Extractor functions I

```
coef()      # Extract the regression coefficients
summary()   # Print a comprehensive summary of the results of
            # the regression analysis
anova()     # Compare nested models and produce an analysis
resid()     # Extract the (matrix of) residuals
plot()      # Produce four plots, showing residuals, fitted
            # values and some diagnostics
model.matrix()
            # Return the design matrix
```

Extractor functions II

```
vcov()      # Return the variance-covariance matrix of the
            # main parameters of a fitted model object
predict()   # A new data frame must be supplied having the
            # same variables specified with the same labels
            # as the original. The value is a vector or
            # matrix of predicted values corresponding to
            # the determining variable values in data frame
step()      # Select a suitable model by adding or dropping
            # terms and preserving hierarchies. The model
            # with the smallest value of AIC (Akaike's
            # Information Criterion) discovered in the
            # stepwise search is returned
```


Generalized linear models

$$g(\mu) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \varepsilon$$

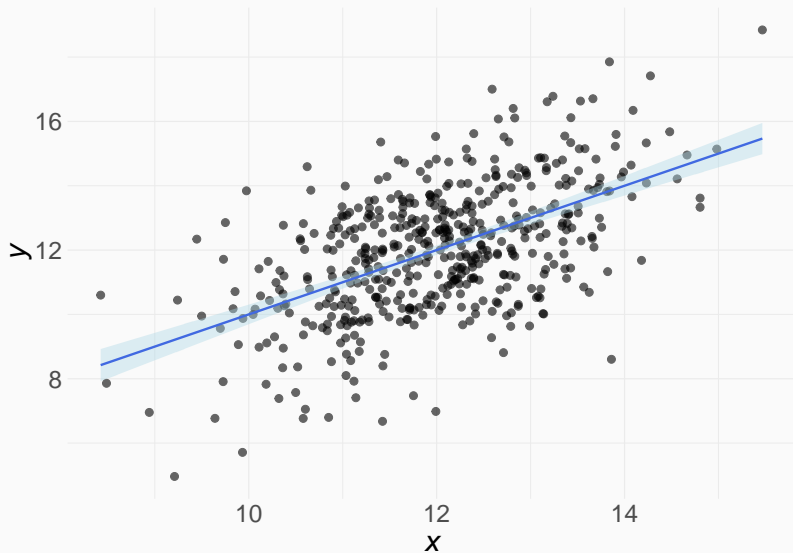
$g()$ is the link functions that connects the mean to the linear combination of predictors.

A GLM is composed of three elements: The response distribution, the link function, and the linear combination of predictors

In R:

```
glm(y ~ x1 + x2 + ... + xk, family(link), data)
```

LM: data



LM: Model

```
model = lm(y ~ x, data = data)
summary(model)
```

Call:

```
lm(formula = y ~ x, data = data)
```

Residuals:

Min	1Q	Median	3Q	Max
-5.2559	-1.1314	0.0162	1.1889	4.4123

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	10.90000	0.10580	103.0	<2e-16 ***
x	0.55000	0.03742	14.7	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

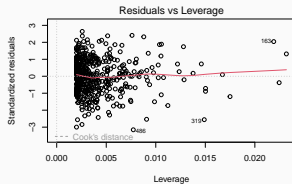
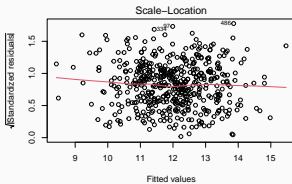
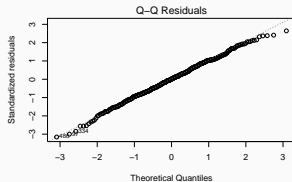
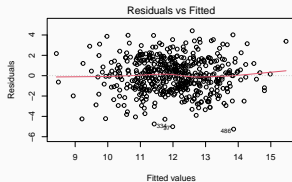
Residual standard error: 1.672 on 498 degrees of freedom

Multiple R-squared: 0.3025, Adjusted R-squared: 0.3011

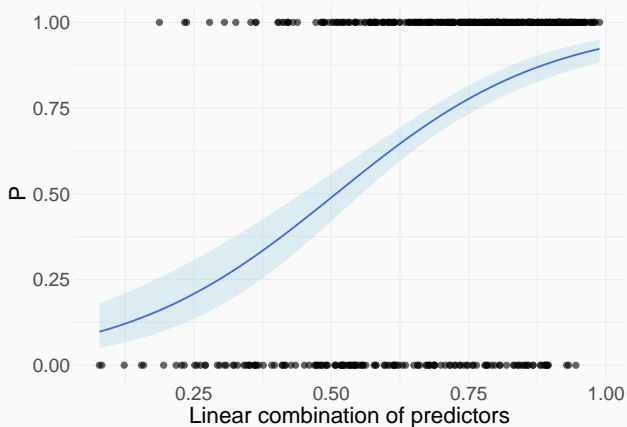
F-statistic: 216 on 1 and 498 DF, p-value: < 2.2e-16

Diagnostic

```
par(mfrow = c(2,2))
plot(model)
```



GLM: Data



GLM: Model

```
model_bin <- glm(z ~ x, data = data, family = "binomial")
summary(model_bin)
```

Call:

```
glm(formula = z ~ x, family = "binomial", data = data)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-0.04795	0.14021	-0.342	0.732
x	0.53713	0.06486	8.281	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 612.55 on 499 degrees of freedom
 Residual deviance: 521.91 on 498 degrees of freedom
 AIC: 525.91

Number of Fisher Scoring iterations: 4

Interpreting the parameters

```
# the coefficients are on the log-odds scale  
my_coef = coef(model_bin)  
# this function yields the probability values  
binomial()$linkinv(my_coef)
```

(Intercept)	x
0.4880139	0.6311456

Families

A special link function to each response variable. In R some different link functions are available by default:

## Family name	Link functions
Binomial	logit, probit, log, cloglog
gaussian	identity, log, inverse
Gamma	identity, inverse, log
inverse.gaussian	$1/\mu^2$, identity, inverse, log
poisson	log, identity, sqrt
quasi	logit, probit, cloglog, identity, inverse, log, $1/\mu^2$, sqrt