01-pRactice

Ottavia M. Epifania, Ph.D

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

13 June 2024

- Import data
- Compute sum scores
- Graphics
- Esport graphs
- R for statistical computing
- Classical hypothesis testing in R
- 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)

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

benessere stipendio genere

```
5 1461.0983
                             m
            1132.3637
3
            1675.9004
                             m
              328.9587
5
           6 1370.0146
                             m
6
              954.3540
```

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)

- Sum across columns \rightarrow sum scores of the respondents rowSums() (rowMean() for computing the mean)
- Sum across rows (righe) \rightarrow 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 \times 19
                                    ID
                                                                      età genere frat item1 item2 item3 item4 item5
                                                                                                                                                                                                                                                                                                                                                                                                                               ลบ1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    a113
               <dbl> 
                                                                               16
                                                                                                                                                                                                                                                                                                                                                         3
                                                                                                                                                                                                                                                                                                                                                                                                                                             5
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        4
                                                                                                                                                                                                                           2
                                                                                                                                                                                                                                                                                                               3
                                                                                                                                                                                                                                                                                                                                                                                                                                             5
                                                                              21
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        4
              i 6 more variables: au5 <dbl>, au6 <dbl>, au7 <dbl>, au8 <dbl>, au9 <dbl
                             au10 <dbl>
```

rowSums():

Well-being continue

```
rowSums(benessere)
```

65

```
[1]
                                        85
                                                    88
 [19]
      79
             112 105
                       98
                           86
                               94
                                   94 120
                                           108
                                                98
                                                   100
                                                       107
         108 107 126 126 116 116 110 117 117 106 111 122 123 129 1
         120 121 148 141 129 130 128 132 148 145 152 136
                                                           137 138 1
         140 143 151 148 158 156 143 149 157 165 162 159 170 163 1
         168 173 176 162 171 174 169 176 172 175 175 172 182 191 1
     178 190 189 186 184 174 187 189
                                      190 187
                                              199
                                                   192 191
                                                           194 199 1
     207 214 208 213 214 205 205 207 227 212 213 208 209 209 216 2
[145] 208 216 215 219 222 210
```

78

79

87

80

83

75

83

Is it meaningful....?

92 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 \rightarrow well-being items

Columns with au \rightarrow self-esteem items

grep()e grep1(): functions for filtering data according to a regular expression (regex)

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

```
Same functioning, different results
(my vector = colnames(benessere))
```

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

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

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

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

OME, UniPD (Lezioni UniCatt)

grepl()

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE. TR TRUE TRUE TRUE TRUE TRUE

coRso

Compute sum scores

```
rowSums() conditioned on specific columns:
```

```
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(benessere)
```

```
babies <- read.table("data/babies.tab")</pre>
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	altezza	
Length:10	Length:10	Min. : 5.411	Min. :46.80	
Class :character	Class :character	1st Qu.: 7.809	1st Qu.:62.90	
Mode :character	Mode :character	Median : 9.429	Median :78.73	
		Mean : 9.970	Mean :74.88	
		3rd Qu.:11.268	3rd Qu.:84.98	
		Max. :17.343	Max. :99.22	

coRso

table()

table(babies\$genere)

f m

OME, UniPD (Lezioni UniCatt)

new benessere

32 22 28 18

```
benessere = read.csv("data/benessere.csv", header = T)
# new dichotomous variable to identify the well-being level
# as high or low
benessere $new_benessere = with(benessere,
                                ifelse(benessere > mean(benessere),
                                       "low",
                                       "high"))
with(benessere, table(genere, new_benessere))
```

genere high low f

m

Single variable

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

```
60 40
```

Multiple variables

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

```
high low
 32 22 54
 28 18 46
```

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

high low f 0.5925926 0.4074074 1 m 0.6086957 0.3913043 1

17 / 107

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

```
# one depenent 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, ...)
```

```
benessere = read.csv("data/benessereScores.csv",
                     header = T, sep =",")
head(benessere, 3)
```

```
ID età genere frat item1 item2 item3 item4 item5 au1 au2 au3 au4 au5 au6
    16
                                           3
                                                               5
                                                                       3
                                                                           3
                               2
                                     4
                                     3
                                                      5
    21
                                                                       4
 3
    28
                                     5
                                                                       4
au8 au9 au10 score_ben score_au
```

13 June

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

```
genere score au
    1 33.08750
```

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 14.46250 33.08750
- 2 14.41429 32.62857



- Recode frat and assign it to a new var into the data frame (siblings: 0 siblings \rightarrow no, 1+ sibilings \rightarrow yes. Hint: use ifelse()
- Compute the mean of score_ben according to siblings
- Compute the mean of score ben and score au according to sibilingsand gender (assign it to the object mean dep)
- Compute the standard deviation of score_ben and score_au according to sibilingsand 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	VAS	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)
```

coRso

tidyverse()

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

```
install.packages("tidyverse")
library(tidyverse)
%>% (Pipe)
Use the combo shift + ctrl + M
Logic:
object %>%
  grouping %>%
  function
```

coRso

```
benessere %>% # object
```

```
# 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
                         13.9
                                     3.25 33.3 4.45
                        14.5
                                      3.31 32.1 3.00
2 no
                         14.7
                                      3.35 33.0 3.74
3 yes
                         14.3
                                     3.23 33.0 4.40
4 yes
```

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

- 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:

- ullet High level functions o actually produce the plot
- Low level functions \rightarrow 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()
```

coRso

Plot layout

Each plot is composed of two regions:

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

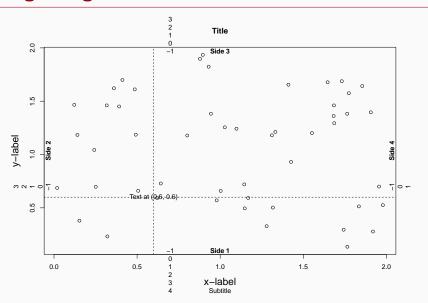
A scatter plot:

```
x \leftarrow runif(50, 0, 2) # 50 uniform random numbers
v \leftarrow 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

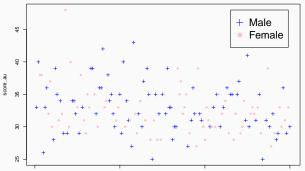


plot()

```
plot(x) # one variable
plot(x, y) # scatter plot
plot(y ~ x) # scatter plot (unless X is categorical)
```

coRso

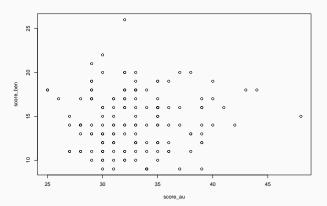
Example: plot(x)



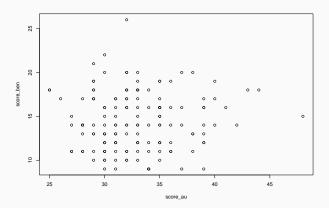
OME, UniPD (Lezioni UniCatt)

Example: plot(x, y)

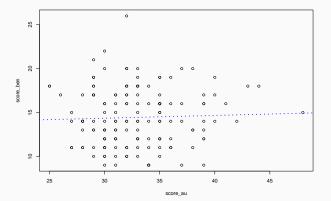
```
with(benessere,
    plot(score_au, score_ben))
```



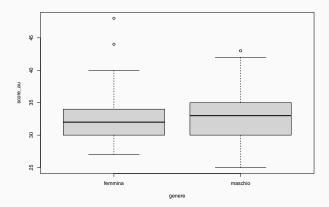
```
with(benessere,
     plot(score_ben ~ score_au))
```



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

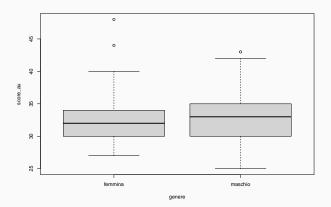


```
benessere$genere <- factor(ifelse(benessere$genere == 1,
                           "maschio", "femmina"))
plot(score_au ~ genere, data = benessere)
```

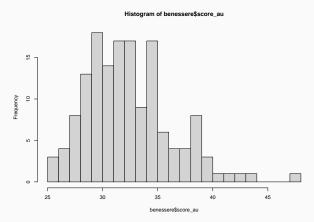


Warning!

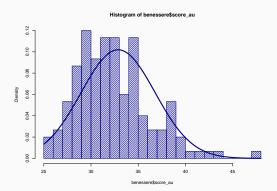
plot(y ~ x) with x categorical is equivalent to boxplot(y ~ x)
boxplot(score_au ~ genere, data = benessere)



hist(benessere\$score_au, breaks = 20)



13 June



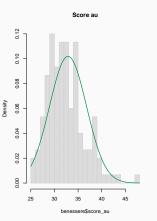
Modify the layout

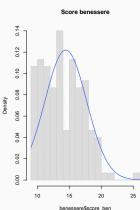
Create the panels

```
par(mfrow=c(nrows, ncolumns)) # panels filled by rows
par(mfcol=c(nrows, ncolumns)) # panels filled by columns
```

coRso

Multi plot: Rows

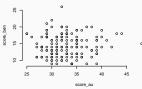


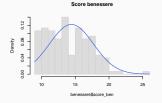


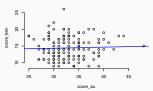
```
par(mfrow=c(1, 2))
hist(benessere$score au,density=50, breaks=20, prob=TRUE,
     main = "Score au")
curve(dnorm(x, mean=mean(benessere$score au),
             sd=sd(benessere$score_au)),
      col="springgreen4", lwd=2, add=TRUE, yaxt="n")
hist(benessere\$core_ben,density=\frac{50}{0}, breaks=\frac{20}{0}, prob=\frac{TRUE}{0},
     main = "Score benessere")
curve(dnorm(x, mean=mean(benessere$score_ben),
             sd=sd(benessere$score_ben)),
      col="royalblue", lwd=2, add=TRUE, yaxt="n")
```

Multiplot columns





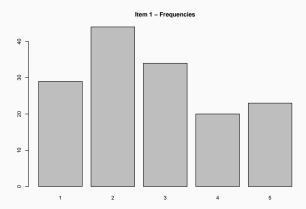




```
par(mfcol = c(2,2))
hist(benessere\$score_au,density=\footnote{50}, breaks=\footnote{20}, prob=\footnote{TRUE},
     main = "Score au")
curve(dnorm(x, mean=mean(benessere$score au),
            sd=sd(benessere$score au)).
      col="springgreen4", lwd=2, add=TRUE, vaxt="n")
hist(benessere$score ben,density=50, breaks=20, prob=TRUE,
     main = "Score benessere")
curve(dnorm(x, mean=mean(benessere$score ben),
            sd=sd(benessere$score ben)).
      col="rovalblue", lwd=2, add=TRUE, vaxt="n")
with(benessere.
     plot(score au, score ben, frame = FALSE))
with(benessere.
     plot(score au. score ben. frame = FALSE))
abline(lm(score ben ~ score au, data = benessere), col = "blue", lwd = 2)
```

barplot(): Absolute frequency

Discrete variables

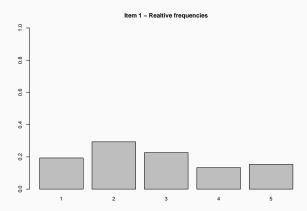


48 / 107

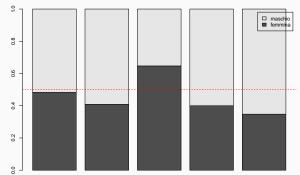
barplot(): Relative frequency

First step: Frequency tables

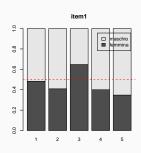
```
perc_item1 = freq_item1/sum(freq_item1)
barplot(perc_item1, ylim = c(0, 1),
        main = "Item 1 - Realtive frequencies")
```

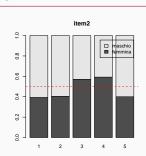


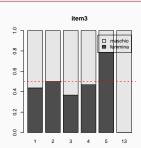
49 / 107

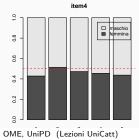


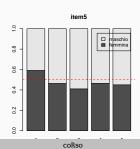
Multi plot: Example











Multi plot: Example

```
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, ltv = 2, col = "red")
```

coRso

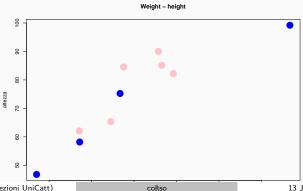
interaction.plot()

Interaction between x and y given a categorical variable z interaction.plot(x, z, y)

Interaction between x and y given a categorical variable z

interaction.plot(x, z, y)

Does the relationship between height and weight change according to gender?



53 / 107

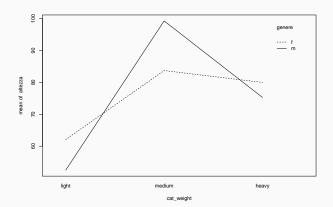
```
babies$cat_weight = with(babies,
                          ifelse(
                            peso <= quantile(babies$peso)[2],</pre>
                            "light", ifelse(peso > quantile(babies$peso)[2]
                                     peso > quantile(babies$peso)[4], "medi
         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

"heavy"))

babies

```
altezza cat_weight
       id genere
                     peso
   baby1
                7.424646 62.07722
1
                                        light
2
   baby2
              m 7.442727 58.18877
                                       light
3
   baby3
                 9.512598 84.52737
                                       heavy
   baby4
              f 11.306349 85.13573
                                       medium
```

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



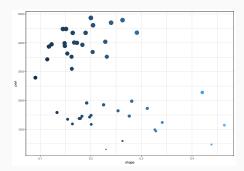
13 June

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")
```



Some new data

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

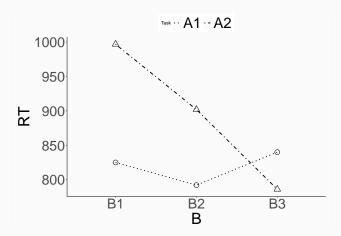
Force A and B to be factor:

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

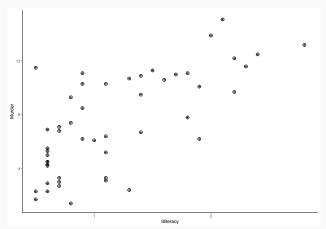
The code for the interaction plot

```
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"))
```

The result



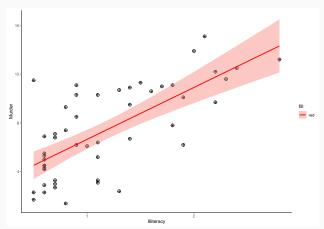
Scatter plot



OME, UniPD (Lezioni UniCatt)

Linear model

```
ggplot(states, # raw data
               aes(x = Illiteracy, y = Murder)) +
  geom_point(size =3, pch=10) + theme_classic() +
  geom_smooth(method="lm", color="red", aes(fill="red"))
```



coRso

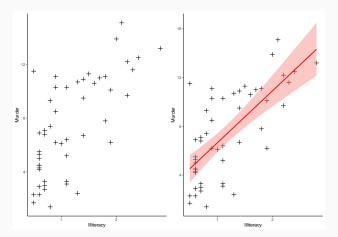
62 / 107

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)) +
```

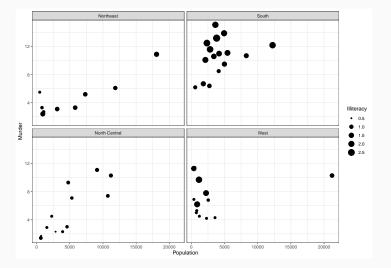
Combine the plots together:

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



coRso

13 June



Multi panel (automatic)

```
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	Α	3.477760
2	sbj1	В	1.748681
3	sbj2	Α	2.405326
4	sbj2	В	2.915242
5	sbj3	Α	3.294477
6	sbi3	В	2.763332

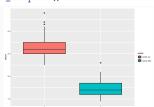
```
small = benessere[, c("ID", "score_au", "score_ben")]
score_long = reshape(small,
        idvar = "ID".
        times = names(small)[-1].
        timevar = "score", v.names = "value",
        varying = list(names(small)[-1]),
        direction = "long")
head(score long)
```

```
TD
               score value
                        33
1.score_au 1 score_au
2.score_au 2 score_au 40
3.score au 3 score au
                      38
                        38
4.score au 4 score au
5.score au 5 score au
                        26
                        33
6.score au
           6 score au
```

13 June

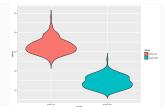
Boxplot

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



Violinplot

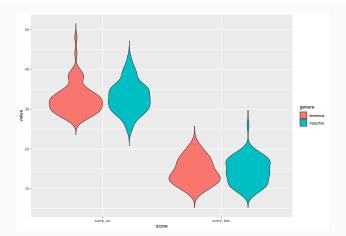
```
ggplot(score_long,
       aes(x = score, y = value,
           fill = score)) +
  geom_violin(trim = FALSE)
```



13 June

coRso

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



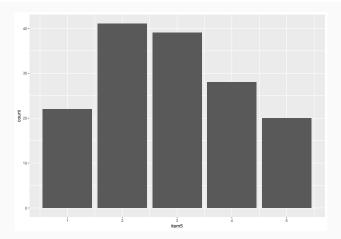
barplot and histogram

```
geom hist(): histogram (continuous variables)
geom bar(): bar plot
```

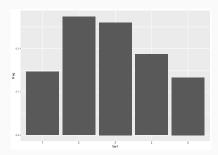
Arguments:

- 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

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



coRso



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
```

```
TD item value
1.item1 1 item1
2.item1
         2 item1
3.item1 3 item1
4.item1
                    3
         4 item1
5.item1 5 item1
. . . .
```

```
proporzione = new_item %>%
  group_by(item, value) %>%
  summarise(prop = n()/nrow(benessere))
proporzione
 A tibble: 26 \times 3
# Groups: item [5]
   item value prop
   <chr> <int> <dbl>
 1 item1 1 0.193
 2 item1 2 0.293
```

The plot

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

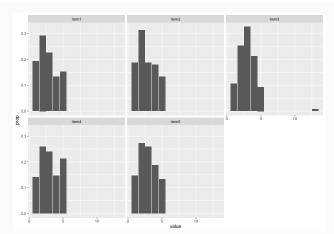


Table of Contents

- 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)

Automatically

```
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

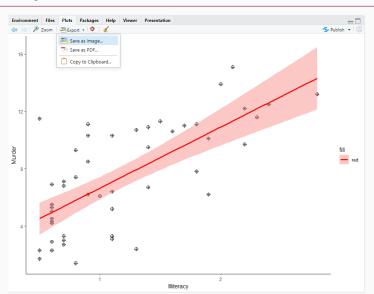


Table of Contents

- 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 ${\tt R}$) 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

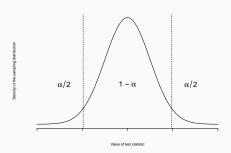
- cor.test(): association between paired samples
- t.test(): one- and two-sample t tests (also for paired data)
- $lm(y \sim x1 + x2 + x3 ...)$: fit a linear model
- glm(y ~ x1 + x2 + x3 ...): fit a generalize linear model

What is the *p*-value?

p-value:

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

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



Independence of observations

Hypothesis:

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

Test statistic:

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

```
tab <- xtabs(~ education + induced, infert)
tab</pre>
```

 ${\tt induced}$

education 0 1 2

0-5yrs 4 2 6 6-11yrs 78 27 15

12+ yrs 61 39 16

chisq.test(tab)

Pearson's Chi-squared test

data: tab

X-squared = 16.531, df = 4, p-value = 0.002384

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:

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

In R:

Pearson's product-moment correlation

```
data: speed and dist
t = 9.464, df = 48, p-value = 1.49e-12
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
    0.6816422  0.8862036
sample estimates:
```

cor

0.8068949

Correlation matrix

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

```
item1
                                    item3
                         item2
                                                 item4
                                                             item5
       1.00000000
                   0.07820461 0.11579134 -0.05894317 -0.18117877
item1
       0.07820461
                    1.00000000 0.24890831
                                           0.21963611 - 0.07757221
item2
item3
       0.11579134
                   0.24890831 1.00000000
                                           0.04977145
                                                        0.03830005
item4 - 0.05894317
                   0.21963611 0.04977145
                                            1.00000000
                                                        0.05751880
item5 -0.18117877 -0.07757221 0.03830005
                                           0.05751880
                                                        1,00000000
```

coRso

13 June

Two (indepdent) 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:

$$T = rac{ar{x_1} - ar{x_2}}{\sigma_{ar{x_1} - ar{y_2}}}, \ T \sim t(n_1 + n_2 - 2)$$

In R:

Welch Two Sample t-test

mean in group OJ mean in group VC

20.66333 16.96333

```
data: len by supp
t = 1.9153, df = 55.309, p-value = 0.06063
alternative hypothesis: true difference in means between group OJ a
95 percent confidence interval:
   -0.1710156   7.5710156
sample estimates:
```

OME, UniPD (Lezioni UniCatt)

Two (depedent) 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:

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

In R function:

```
with(sleep,
     t.test(extra[group == 1],
            extra[group == 2], paired = TRUE), alternative = "great
    Paired t-test
data: extra[group == 1] and extra[group == 2]
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -2.4598858 -0.7001142
sample estimates:
mean difference
          -1.58
```

coRso

13 June

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)

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

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

Linear models

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

In R:

$$lm(y \sim x1 + x2 + ... + xk, data)$$

Extractor functions I

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
```

coRso

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

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

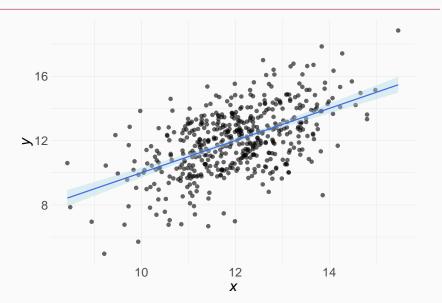
A GLM is composed of three elements: The response distribution, the link function, and the linear combination of predictors

In R:

$$glm(y \sim x1 + x2 + ... + xk, family(link), data)$$

OME, UniPD (Lezioni UniCatt)

LM: data



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

```
Call:
```

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

Residuals:

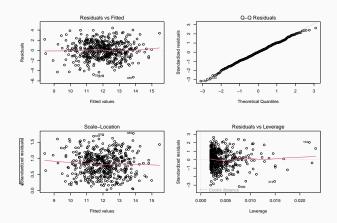
```
Min 1Q Median 3Q Max
-5.2559 -1.1314 0.0162 1.1889 4.4123
```

Coefficients:

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

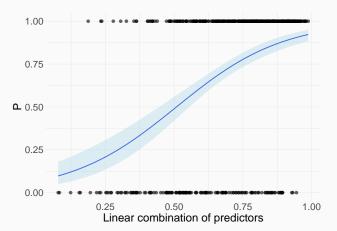
Residual standard error: 1.672 on 498 degrees of freedom Multiple R-squared: 0.3025, Adjusted R-squared: 0.3011 F-statistic: 216 on 1 and 498 DF, p-value: < 2.2e-16

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



https://library.virginia.edu/data/articles/diagnostic-plots

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|)
0.53713 0.06486 8.281 <2e-16 ***
x
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
ATC: 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 probabilty values
binomial()$linkinv(my_coef)
(Intercept)
                      X
  0.4880139 0.6311456
```

13 June

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	<pre>1/mu^2, identity, inverse, log</pre>
poisson	log, identity, sqrt
quasi	logit, probit, cloglog, identity, inverse,
	log, 1/mu ² , sqrt

13 June