03-descRivi i dati
(e programma un pochino)

Ottavia M. Epifania, Ph.D

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

8-9 Giugno 2023

Table of contents

Table of Contents

summary()

summary() è un funzione applicata a diversi oggetti R, dai data set agli oggetti che contengono i risultati delle analisi (e.g., modelli lineari, t test, ecc.)

Può essere applicato ai dati qualitativi e quantitaivi:

```
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
```

O a data set interi:

```
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()

Crea delle tabelle di frequenza

Applicabile a una singola variabile:

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

```
f m
6 4
```

O per costruire tavole di contingenza:

```
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() e le percentuali

Caso semplice:

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

```
 f  m
60 40
```

Un po' più difficile:

```
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

Aggrega una variabile "dipendente" a seconda di una serie di variabili dipendenti e vi applica una funzione

```
# Una variabile dipendente (y) e una grouping variable (x)
aggregate(y ~ x, data = data, FUN, ...)

# Più variabili dipendenti e più variabili indipendenti
aggregate(cbind(y1, y2) ~ x1 + x2, data = data, FUN, ...)
```

Aggregating

Aggrega una variabile "dipendente" a seconda di una serie di variabili dipendenti e vi applica una funzione

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

```
# Una variabile dipendente (y) e 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: Esempi

Importiamo il data set del benessere con gli scoring fatti da noi:

```
dati = read.csv("data/benessereScores.csv", header = T, sep =",")
```

Calcoliamo la media a seconda del genere:

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

```
  genere score_au
1      1 33.08750
2      2 32.62857
```

Calcoliamo la media di entrambi gli score a seconda del genere:

```
aggregate(cbind(score_ben, score_au) ~ genere, data = dati, m
```

```
  genere score_ben score_au
1      1  14.46250 33.08750
2      2  14.41429 32.62857
```

Your turn!

- Ricodificate `frat` e assegnatela a una nuova variabile del data frame (`siblings`: >0 fratelli $\rightarrow$ no $>>1+ \rightarrow$ yes
- Calcolate la media di `score_ben` a seconda di `siblings`
- Calcolate media di `score_ben` e `score_au` a seconda di `sibilingse` genere (assegnatelo a `mean_dep`)
- Calcolate deviazione standard di `score_ben` e `score_au` a seconda di `sibilingse genere` (assegnatelo a `sd_dep`)
- unite `mean_dep` e `sd_dep` in un unico oggetto, `descr`

Attenzione!

Non tutte le colonne devono avere lo stesso nome quando usate merge

Risultato:

```
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

Soluzione

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

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

descr = merge(mean_dep, sd_dep)
```

Soluzione alternativa: `tidyverse()`

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

`R` appositamente per data science

All'inizio è un po' ostico ma poi rende la vota più semplice

Soluzione alternativa: `tidyverse()`

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

`R` appositamente per data science

All'inizio è un po' ostico ma poi rende la vota più semplice

\begin{center}

\texttt{%>%} (Pipe)

\end{center}

Si ottiene con la combo di tasti `shift + ctrl + M`

La logica:

```
oggetto %>%
  funzione
```

Le nostre statistiche descrittive

```
dati %>%
  group_by(siblings, genere) %>%
  summarise(m_benessere = mean(score_ben),
            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!

- Con i dati del benessere: Calcolate minimo, massimo e mediana di `score_au` e `score_ben` utilizzando `tidyverse`
- Importate i dati dei babies e calcolate le decsrittive del peso e dell'altezza con `tidyverse`

Table of Contents

Siate pront* a sbagliare (anche tanto)

Coding is ~~hard~~ art

Eyes on the prize, ma portate pazienza e smontate l'obiettivo in sotto obiettivi raggiungibili

You'll never walk alone $\rightarrow$ `stackoverflow`

La regola d'oro:

$$10 \text{ minutes of coding} = 10 \text{ hours of debugging}$$

`ifelse()`

Esecuzione condizionale:

Easy: `ifelse(test, if true, if false)`

```
ifelse(babies$peso > 7, "big boy", "small boy")
```

Pro

- è super facile da usare
- Si possono embeddare diversi `ifelse()`

Contro

- Funziona bene se non si vuole testare un valore specifico

if () {} else {}

Se si ha solo una condizione:

```
if (test_1) {
  command_1
} else {
  command_2
}
```

`if () {} else if () {}`

Più condizioni

```
if (test_1) {
  command_1
} else if (test_2) {
  command_2
} else {
  command_3
}
```

`test_1` (ed eventualmente `test_2`) **devono** essere `TRUE` o `FALSE`

```
if(!is.na(x)) y <- x^2 else stop("x is missing")
```

Loops

`for()`

Ripete un comando per una quantità di volte definite dall'utente:

```
# Don't do this at home
x <- rnorm(10)
y <- numeric(10)   # crea un contentitore vuoto
for(i in seq_along(x)) {
y[i] <- x[i] - mean(x)
}
```

Loops

`for()`

Ripete un comando per una quantità di volte definite dall'utente:

```
# Don't do this at home
x <- rnorm(10)
y <- numeric(10)   # crea un contentitore vuoto
for(i in seq_along(x)) {
y[i] <- x[i] - mean(x)
}
```

La soluzione migliore:

```
y = x - mean(x)
```

Evitate i loop

Don't loop, `apply()`!

`apply()`

```
X <- matrix(rnorm(20),
            nrow = 5, ncol = 4)
apply(X, 2, max) # trova il massimo per ogni colonna
```

```
[1] 1.177962 1.749394 0.400801 1.186019
```

Evitate i loop

Don't loop, `apply()`!

`apply()`

```
X <- matrix(rnorm(20),
            nrow = 5, ncol = 4)
apply(X, 2, max) # trova il massimo per ogni colonna
```

```
[1] 1.177962 1.749394 0.400801 1.186019
```

`for()`

```
y = NULL
for (i in 1:ncol(X)) {
  y[i] = max(X[, i])
}
```