Analisi dei dati con R

Stefano Bussolon

Analisi dei dati con R

Iscriviti al corso online

Un corso online di 9 lezioni (18 ore) più una sessione di esame (facoltativo) dove verrà introdotto R, il caricamento e la pulizia dei dati, le analisi descrittive e le forme principali di analisi inferenziale: chi quadro, correlazione, t-test, Anova. Verrà infine introdotto l'approccio simulativo all'analisi inferenziale.

Per maggiori informazioni e per iscriverti vai alla pagina del corso:

bussolon.it/formazione/analisi-dei-dati-con-r.html

Analisi dei dati con R

Stefano Bussolon

5 aprile 2019

Indice

Introduzione

Versione provvisoria

Perché R

Perché r

Perché è gratuito, open source, disponibile su più sistemi operativi (windows, mac, linux).

Perché dispone di una enorme catalogo di pacchetti per l'importazione di dati, la loro manipolazione, l'analisi statistica, la visualizzazione, la creazione di report, e per elaborazioni più complesse (ad esempio machine learning).

Questi pacchetti sono spesso molto aggiornati, e rappresentano lo stato dell'arte nell'analisi dei dati.

Il linguaggio è stato specificamente creato e sviluppato per l'analisi dei dati.

Internet è ricco di risorse, anche gratuite, che permettono di imparare ad usare R.

Per i più esperti, R offre anche delle caratteristiche avanzate: programmazine funzionale, possibilità di invocare funzioni in altri linguaggi (c, fortran).

R è estensibile, attraverso l'installazione di pacchetti esterni. In ambito statistico e in psicometria R è sostanzialmente uno standard

R vs excel

Excel funziona meglio su cose semplici e che non ho bisogno di replicare.

Quando è necessario fare delle analisi statistiche più complesse, integrare insiemi di dati, creare presentazioni grafiche, è decisamente meglio R.

Vantaggi di R

- R ha l'ecosistema più ricco ed avanzato di pacchetti per l'analisi dei dati.
- Dietro ad R c'è una grossa comunità di utilizzatori che condivide pacchetti, librerie e che è disponibile ad aiutare, attraverso forum e spazi di collaborazione
- R offre strumenti di visualizzazione più avanzati
- R è gratuito ed open source
- R è un linguaggio di programmazione.
- R riesce a leggere (importare) una vasta gamma di dati
- R può manipolare file di dati molto grandi
- negli insiemi di dati più grandi, R è più veloce
- con R è possibile organizzare meglio i progetti (separando ed integrando dati, codice/script, testi, figure)
- è molto più facile automatizzare i processi con R che con excel. Più in generale, è più facile automatizzare un processo in un ambiente di programmazione che con una GUI.
- accuratezza: excel è meno accurato, e più soggetto a bug, rispetto ad R.

Gli script in R

- sono più facili da documentare

- possono essere ri-utilizzati
- è più facile replicare una analisi
- possono essere utilizzati su basi di dati diverse
- se il codice viene commentato, è più facile ricordare cosa si sta facendo (o capire cosa ha fatto chi ha scritto il codice). Documentare i passaggi fatti con excel è molto meno naturale.

Riproducibilità

È molto più facile riprodurre analisi anche complesse. Questo ha almeno due vantaggi:

- è possibile applicare, in maniera piuttosto semplice, la stessa analisi su dati diversi
- questo rende più facile la verifica e la riproducibilità delle analisi, ad esempio da parte di ricercatori o laboratori diversi (open science?)
- è possibile applicare al codice dei software di controllo di versione (ad esempio git)
- è facile condividere il proprio codice, ed adottare delle metodologie di collaborazione

Riferimenti

- Excel vs R: When to use what - R for Excel Users
- R for beginners: How to transition from Excel to R | TrendCT
- R for Excel Users - Alteryx Community
- Understanding R programming over Excel for Data Analysis | gap intelligence
- Why would you learn R if you have Excel working well? - Quora

R vs Python

Se excel non è la soluzione migliore per fare analisi dei dati, R non è l'unico ambiente disponibile. Fra le alternative, una delle più accreditate è il linguaggio di programmazione python, integrato con alcuni pacchetti quali pandas e numpy.

Il dibattito Python vs R è molto acceso, in quanto entrambi i linguaggi costituiscono un'ottima scelta.

L'opinione più condivisa è che nella statistica inferenziale, R *batte* Python. Nella manipolazione dei dati i due ambienti se la cavano altrettanto bene. Python ha il vantaggio di essere un linguaggio di programmazione *all purpose*, mentre R è molto più specializzato all'analisi dei dati.

Approccio pragmatico

Generalmente, l'approccio più utile è quello di conoscere diversi strumenti, ed utilizzarli per fare cose diverse.

Excel, libreoffice o i fogli google per le operazioni più semplici, soprattutto se si deve collaborare con persone che non usano R.

Gli editor di testo possono essere molto utili per pulire file in formato tsv o csv.

Soprattutto in ambiente linux, vi sono programmi a linea di comando che permettono di fare semplici manipolazioni di dati.

Per le analisi più complesse, e per le presentazioni professionali, usare R o python.

Riferimenti

- Python vs R for Data Science
- R Vs Python: What's the difference?
- Python or R for Machine Learning and Data Science | Netguru Blog on Python
- Choosing R or Python for data analysis? An infographic (article) - DataCamp
- Python vs. R : statistics

Parte I

Introduzione a r

Capitolo 1

Primi passi con R

Primi passi con R

Scaricare R

R è un software free ed open source, che può essere liberamente scaricato dal sito R: The R Project for Statistical Computing, ad esempio utilizzando il mirror dell'università di Padova: The Comprehensive R Archive Network

R è disponibile per piattaforme Windows, Mac, Linux.

R è un software a riga di comando. Questo significa che ogni istruzione viene comunicata ad R attraverso il prompt della riga di comando.

RStudio

Per rendere più semplice la vita, è opportuno utilizzare RStudio, un IDE (integrated development environment) che rende più pratico scrivere codice in R. Anche RStudio può essere installato su Windows, Mac, Linux.

rdpeng - Getting started with R

Installing R and RStudio - stat545

Usare R come una calcolatrice

Per iniziare a prendere confidenza con la riga di comando di R, è possibile cominciare a giocarci, provando le funzioni più elementari. La riga di comando, ad esempio, può essere utilizzata per calcolare alcune semplici operazioni.

Operazioni aritmetiche di base

Addizione, sottrazione, moltiplicazione, divisione, elevazione a potenza

```
# addizione
7+4
```

```
## [1] 11
```

```
# sottrazione
13-5
```

```
## [1] 8
```

```
# moltiplicazione
7*7
```

```
## [1] 49
```

```
# divisione
23/3
```

```
## [1] 7.666667
```

```
# divisione intera
23 %/% 3
```

```
## [1] 7
```

```
# modulo (il resto della divisione)
23 %% 3
```

```
## [1] 2
```

```
# elevazione a potenza
2^3
```

```
## [1] 8
```

```
3^2 + (7-2)*3
```

```
## [1] 24
```

Assegnazione di variabili

La creazione di una variabile avviene attraverso la sintassi **<-**. In R non è necessario definire il tipo di variabile. Attraverso l'assegnazione, sarà R a creare un tipo opportuno di variabile (o di oggetto).

```
numero1 <- 5
etichetta1 <- "Antonio"
# le parentesi servono a stampare il risultato a video
(numero2 <- 5.12)
```

```
## [1] 5.12
```

```
numero3 <- 10/3
# invocando la variabile ne stampo il valore
numero3
```

```
## [1] 3.333333
```

```
# se alla stessa variabile assegno un nuovo valore
# perdo il valore precedente
numero3 <- 10*2
numero3
```

```
## [1] 20
```

```
# operazioni usando le variabili
numero1 * numero2
```

```
## [1] 25.6
```

```
# assign (variabile, valore) equivale a variabile <- valore
assign("numero4", 7)
numero4
```

```
## [1] 7
```

`<-` è l'operatore di assegnamento, ed equivale alla funzione `assign`. Pertanto `numero <- 5` è una scorciatoia della funzione `assign("numero", 5)`.

La funzione `ls()` mi permette di elencare tutte le variabili (oggetti) attualmente attivi nel framework.

```
ls()
```

```
## [1] "etichetta" "etichetta1" "giusto"    "numero"    "numero1"
## [6] "numero2"   "numero3"    "numero4"   "pippo"     "pluto"
## [11] "sbagliato"  "variabile1" "variabile2" "x"
```

La funzione `rm()` rimuove un oggetto. Ad esempio, con il comando `rm(variabile1)` l'oggetto variabile1 sarà cancellato, e non più disponibile.

Operazioni booleane

Le operazioni booleane sono finalizzate a confrontare due elementi. L'output di queste operazioni è di tipo booleano.

Attenzione: per valutare l'uguaglianza fra due elementi si usa ==

```
quattro <- 4
cinque <- 5
# la variabile quattro è uguale a 4?
quattro == 4
```

```
## [1] TRUE
```

```
cinque == 4
```

```
## [1] FALSE
```

```
# quattro è minore di cinque?
quattro < cinque
```

```
## [1] TRUE
```

```
# quattro è minore o uguale a 4?
quattro <= 4
```

```
## [1] TRUE
```

```
quattro < 4
```

```
## [1] FALSE
```

```
# quattro è diverso da cinque?
quattro != cinque
```

```
## [1] TRUE
```

```
giusto = TRUE
sbagliato = FALSE
# TRUE or FALSE = TRUE
giusto | sbagliato
```

```
## [1] TRUE
```

```
# TRUE and FALSE = FALSE
giusto & sbagliato
```

```
## [1] FALSE
```

```
# vero o falso = vero
(5>4) | (4>5)
```

```
## [1] TRUE
```

```
# vero e falso = falso
(5>4)&(4>5)
```

```
## [1] FALSE
```

```
(6<7)&(7<8)
```

```
## [1] TRUE
```

```
# ! significa not.
!TRUE
```

```
## [1] FALSE
```

```
!FALSE
```

```
## [1] TRUE
```

```
!sbagliato
```

```
## [1] TRUE
```

```
sbagliato & (quattro==4)
```

```
## [1] FALSE
```

Riferimenti

- Quick-R: Operators
- https://psyr.org/variables.html

:todo: linee guida / guida di stile: Style guide · Advanced R.

Guide di stile

Nello scrivere del codice, è opportuno seguire delle guide di stile. Questo garantisce una maggiore consistenza e codice più elegante.

Uno degli aspetti più importanti e meno banali è dare dei nomi appropriati alle variabili. Appropriati significa comprensibili ma non troppo lunghi.

Per separare le parole in un nome composto, Hadley Wickham suggerisce di usare l'underscore `_`. Ad esempio `nome_variabile`. Personalmente, però, non disdegno di utilizzare a volte il camelCase: `nomeVariabile` (peraltro suggerito nelle linee guida di Google).

Per approfondire, si consigliano queste risorse:

- The tidyverse style guide
- Google's R Style Guide

Help

Sebbene sia utile conoscere a memoria le funzioni più importanti, è assolutamente normale non ricordare tutte le funzioni, tutti i parametri.

Dalla sezione documenti del sito di R si possono scaricare delle R Reference Card, ovvero delle raccolte, con breve spiegazioni, delle funzioni più importanti. Strumenti come gli cheatsheets di RStudio sono estremamente utili per avere un riferimento veloce di una serie di argomenti.

Soprattutto, è opportuno imparare ad usare gli aiuti che l'ambiente R ci offre.

- help(nome) help sulla funzione 'nome'
- ?nome equivale a help(nome)
- apropos("nome") elenca le funzioni che contengono "nome"
- example(funzione) mostra degli esempi dell'uso della funzione
- RSiteSearch("kmeans") cerca informazioni relative alla funzione "kmeans" su internet.

Risorse

- The Very Basics | Hands-On Programming with R
- Tutorials for learning R | R-bloggers
- R-intro.pdf
- https://psyr.org/getting-started.html
- Learning - Efficient R programming

+++todo+++ da completare

Lo spazio di lavoro

Quando creiamo una variabile, o una funzione, tecnicamente *dove* la creiamo? Tecnicamente, nello *spazio di lavoro* (workspace).

Quick-R: The Workspace Working in the Console – RStudio Support Using Projects – RStudio Support R basics, workspace and working directory, RStudio projects Quick-R: The Workspace

RStudio IDE

PowerPoint Presentation

Capitolo 2

Le strutture di dati

Le strutture di dati

R opera su strutture di dati. Le tipologie di strutture di dati principali in R sono i *vettori*, le *matrici* e gli *array*, le *liste* e i *data frame*.

Vettori

Un vettore è una collezione ordinata di *celle* dello stesso tipo. Ogni vettore ha un tipo, una lunghezza e può avere degli *attributi*, o *metadati*. I tipi di vettori sono *logical*, *integer*, *double*, *complex*, *character* e *raw*.

Riprendiamo gli esempi del capitolo precedente.

```
booleano <- TRUE
intero1 <- 5L # L forza l'intero
etichetta <- "Antonio"
numerico1 <- 5.12
numerico2 <- 10/3
```

```
mode(booleano)
```

```
## [1] "logical"
```

```
mode(intero1)
```

```
## [1] "numeric"
```

```
mode(etichetta)
```

```
## [1] "character"
```

```
mode(numerico1)
```

```
## [1] "numeric"
```

```
mode(numerico2)
```

```
## [1] "numeric"
```

Apparentemente, le variabili appena create sono degli scalari. In realtà sono vettori di lunghezza 1.

```
length(intero1)
```

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

La funzione `c()`

Il modo più semplice per creare un vettore di lunghezza superiore a 1 è attraverso la funzione `c()`. Questa funzione permette di *combinare* o *concatenare* una lista di argomenti in un vettore.

```
vettore1 <- c(1,5,88,56314, 12,0,22)
(vettore2 <- c(1:5, sesto=10, settimo=17))
```

```
##                                         sesto settimo
##       1       2       3       4       5      10      17
```

```
c(TRUE,5)
```

```
## [1] 1 5
```

```
c(5, 10.5, TRUE, "next")
```

```
## [1] "5"    "10.5" "TRUE" "next"
```

Coercizione implicita

Poiché nell'ultimo comando abbiamo mescolato numeri interi, numeri decimali e stringhe, R ha trasformato il vettore in un vettore di stringhe.

In maniera simile, il vettore `c(TRUE,5)` ha combinato una variabile logica ed una numerica, e dunque r ha implicitamente trasformato la variabile logica (`TRUE`) in numerica (`1`).

- Coercion - R Programming for Data Science

Proprietà

I vettori hanno 3 proprietà di base: il *tipo*, la *lunghezza*, gli *attributi*.

Il tipo può essere analizzato con la funzione `typeof()`, la lunghezza con `length()`, gli attributi con `attributes()`.

```
typeof(vettore1) # il tipo
```

```
## [1] "double"
```

```
length(vettore1) # la lunghezza
```

```
## [1] 7
```

```
attributes(vettore1) # gli attributi
```

```
## NULL
```

Gli attributi sono un insieme di coppie `chiave-valore`

È possibile associare un attributo ad un oggetto attraverso la sintassi `attr(oggetto,chiave) <- valore`; il valore di un attributo può essere richiamato con la funzione `attr(oggetto,chiave)`

```
attr(vettore1, "capelli") <- "biondi"
attr(vettore1, "occhi") <- "verdi"
attributes(vettore1)
```

```
## $capelli
## [1] "biondi"
##
## $occhi
## [1] "verdi"
```

```
attr(vettore1, "occhi")
```

```
## [1] "verdi"
```

- R Language Definition - Attributes
- Data structures - Attributes

Vi sono 3 attributi dei vettori che hanno uno statuto speciale: `names`, `dim`, e `class`

```
names(vettore2)
```

```
## [1] ""      ""      ""      ""      ""      "sesto"  "settimo"
```

```
# si può usare anche la sintassi seguente
# ma è sconsigliata
attr(vettore2,"names")
```

```
## [1] ""      ""      ""      ""      ""      "sesto"  "settimo"
```

```
class(vettore2)
```

```
## [1] "numeric"
```

Names

names è un attributo che, se presente, attribuisce una etichetta agli elementi di un vettore o una lista. Si noti che `names(vettore2)` è equivalente a `attr(vettore2,"names")`, ma la prima forma è preferibile alla seconda.

- R Language Definition - Names

Class

`class()` è l'attributo che permette ad r di implementare una logica orientata agli oggetti. Ad esempio, permette il polimorfismo delle funzioni

- oop - What is polymorphism, what is it for, and how is it used? - Stack Overflow
- Polymorphism (computer science) - Wikipedia

Str

La funzione `str(variabile)` permette di visualizzare la struttura sottostante alla variabile.

```
str(vettore1)
```

```
##  atomic [1:7] 1 5 88 56314 12 ...
##  - attr(*, "capelli")= chr "biondi"
##  - attr(*, "occhi")= chr "verdi"
```

NA - valori mancanti

Spesso i dati su cui lavoriamo sono incompleti: il valore di alcune *celle* (alcune variabili di alcune osservazioni) sono mancanti. Per rappresentare i valori mancanti, r utilizza la costante NA (*not available*). Tecnicamente, NA è un vettore logico di lunghezza 1.

```
incompleto <- c(1, 2, NA, 10, 3)
# is.na mi dice se l'elemento è NA
is.na(incompleto)
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE
```

NA tende a *contagiare* valori non NA in caso di operazioni logiche ed aritmetiche:

```
1 + NA
```

```
## [1] NA
```

```
NA == 5
```

```
## [1] NA
```

Per comprendere la logica, possiamo tradurre NA come *non so*. Quanto fa *non so* + 1? *non so*. *non so* è uguale a 5? *non so*.

```
TRUE | NA
```

```
## [1] TRUE
```

```
TRUE & NA
```

```
## [1] NA
```

```
FALSE | NA
```

```
## [1] NA
```

```
FALSE & NA
```

```
## [1] FALSE
```

Anche questi risultati apparentemente contraddittori sono in realtà coerenti. vero o qualcosa è sempre vero, non importa cosa sia quel qualcosa. Falso e qualcosa è sempre falso. Vero e *non so* è *non so*, falso o *non so* è *non so*.

- R Programming for Data Science

Matrici

Una matrice è una collezione bidimensionale, rettangolare di `celle` dello stesso tipo. Un array è una collezione che può avere più di 2 dimensioni o, detto in altro termini, una matrice è un array di due dimensioni.

In realtà una matrice è un vettore con attributo dim e con lunghezza righe*colonne.

Creare e manipolare matrici

```
(matrice1 <- matrix(c(11,12,13,14,21,22,23,24,31,32,33,34),nrow=4,ncol=3))
```

```
##      [,1] [,2] [,3]
## [1,]   11   21   31
## [2,]   12   22   32
## [3,]   13   23   33
## [4,]   14   24   34
```

```
# la matrice è comunque un vettore
str(matrice1)
```

```
##  num [1:4, 1:3] 11 12 13 14 21 22 23 24 31 32 ...
```

```
# dim restituisce la dimensione (righe, colonne)
dim(matrice1)
```

```
## [1] 4 3
```

```
# la lunghezza del vettore sottostante (r*c)
length(matrice1)
```

```
## [1] 12
```

```
# as.vector restituisce il vettore sottostante
as.vector(matrice1)
```

```
##  [1] 11 12 13 14 21 22 23 24 31 32 33 34
```

```
# t traspone la matrice:
# le righe diventano colonne, e viceversa
t(matrice1)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   11   12   13   14
## [2,]   21   22   23   24
## [3,]   31   32   33   34
```

Dim è l'attributo che permette ad un vettore di essere rappresentato come una matrice.

con `str(matrice1)` possiamo vedere che la matrice è comunque un vettore di lunghezza `righe*colonne` e con attributo `dim`, che rappresenta la dimensione nella forma `righe, colonne`. Il vettore sottostante può essere *estratto* con la funzione `as.vector`.

Attraverso le funzioni `colnames()` e `rownames()` posso assegnare o richiamare i nomi delle colonne e delle righe.

```
# colnames assegna i nomi alle colonne
colnames(matrice1) <- c("C_A", "C_B", "C_C")
# rownames assegna i nomi alle righe
rownames(matrice1) <- c("R1","R2","R3","R4")
matrice1
```

```
##    C_A C_B C_C
## R1  11  21  31
```

```
## R2  12  22  32
## R3  13  23  33
## R4  14  24  34
```

Fattori

Un fattore è un vettore che rappresenta una variabile nominale (categoriale). Un fattore ordinato rappresenta una variabile ordinale.

I fattori sono dunque utilizzati per rappresentare le variabili categoriali, in forma di stringhe di caratteri, che hanno un insieme fisso e noto di possibili valori.

- 15 Factors | R for Data Science

```
cat1 <- c("uno","due","uno","uno","due","uno","due")
nominale <- factor(cat1)
# levels implicitamente definisce l'ordine dei fattori
ordinale <- factor(cat1,levels = c("uno","due"))
# levels è un attributo del fattore
attributes(nominale)
```

```
## $levels
## [1] "due" "uno"
##
## $class
## [1] "factor"
```

```
# il tipo è integer
typeof(nominale)
```

```
## [1] "integer"
```

```
as.integer(nominale)
```

```
## [1] 2 1 2 2 1 2 1
```

```
as.integer(ordinale)
```

```
## [1] 1 2 1 1 2 1 2
```

```
# per estrarre i livelli
levels(ordinale)
```

```
## [1] "uno" "due"
```

Dietro le quinte R identifica (se non esplicitamente dichiarati) i *livelli* del vettore, ovvero i diversi valori. Se non esplicitamente dichiarati, i livelli sono ordinati alfabeticamente. Come possiamo vedere, un fattore è di tipo `integer` anche se è stato creato da un vettore di stringhe. Gli interi mappano l'indice del corrispondente livello. Con `as.integer(nominale)` possiamo vedere che la stringa "uno" è mappata con il numero 2, in quanto è il secondo valore dei livelli, e "due" con 1.

Il vettore `ordinale` è stato creato definendo esplicitamente i livelli e, implicitamente, il loro ordine (prima "uno" e poi "due"). `levels` è un vettore di caratteri, che può essere ottenuto con la funzione `levels`. Poiché `levels` è un attributo, `levels(ordinale)` è equivalente a `attr(ordinale,"levels")`.

Le liste

Una lista è una collezione ordinata ed eterogenea di oggetti anche complessi. Poiché la lista può contenere altre liste, attraverso la lista si possono costruire delle strutture ad albero. Per creare una lista, si usa la funzione `list()`

```
lista1 <-list(
    matrice=matrix(10:18,nrow=3),
    saluto=rep("ciao",3),
    colazione=c("cappuccio","brioche"))
lista1
```

```
## $matrice
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
##
## $saluto
## [1] "ciao" "ciao" "ciao"
##
## $colazione
## [1] "cappuccio" "brioche"
```

```
str(lista1)
```

```
## List of 3
##  $ matrice  : int [1:3, 1:3] 10 11 12 13 14 15 16 17 18
##  $ saluto   : chr [1:3] "ciao" "ciao" "ciao"
##  $ colazione: chr [1:2] "cappuccio" "brioche"
```

Data frame

Un data frame è una lista di vettori o fattori o matrici di uguale lunghezza (per le matrici conta il numero di righe). In quanto tale, può essere rappresentato come una matrice, dove i vettori corrispondono alle colonne. La differenza più importante fra data.frame e matrice è che mentre nella seconda tutti gli elementi sono dello stesso tipo, nel data.frame le colonne possono essere di tipo diverso.

I data frame sono le strutture che rappresentano i dati nella forma `variabili*osservazioni`, dove le variabili sono le colonne e le osservazioni le righe. L'attributo `names` permette di attribuire una etichetta alle variabili.

```
nome <- c("luigi","mario","antonella","luca")
anno <- c(1956,1945,1972, 1976)
condizione <- c("exp","controllo","exp","controllo")
soggetti <- data.frame (nome,anno,condizione)
soggetti
```

```
##        nome anno condizione
## 1     luigi 1956        exp
## 2     mario 1945  controllo
```

```
## 3 antonella 1972        exp
## 4      luca 1976  controllo
```

Riferimenti

- An Introduction to R
- Basic Types - R Language Definition
- Vectors | R for Data Science
- Quick-R: Data Types
- Vectors - psyr.org
- Vectors | R for Data Science
- Data types - psyr.org
- Dates - psyr.org
- R Nuts and Bolts - R Programming for Data Science

Capitolo 3

Filtri/subsetting

Filtri ed estrazione

A partire da una struttura di dati (un vettore, una matrice, un data frame) è spesso necessario estrarre un sottoinsieme di dati. Subsetting è l'operazione di estrazione di un sottoinsieme di valori da un vettore, una lista o un data frame.

Estrazione

Estrazione da vettori

Il modo più semplice per ottenere un sottoinsieme di dati da un vettore è attraverso l'operatore di estrazione `[]`. Per capirne il funzionamento, è utile ricordare che in R le strutture di dati principali (vettori, matrici, liste, data frame) sono delle collezioni **ordinate** di elementi. Questo significa che ogni elemento ha una posizione nella collezione. La sintassi `vettore1[5]` permettere di accedere al quinto elemento del vettore `vettore1`.

Nell'esempio precedente `5` è l'indice che viene usato per selezionare un elemento del vettore. In R non esistono scalari, e dunque `5` equivale a `c(5)`.

```
vettore1 <- c(2,4,6,8,10,12,14,16,18,20)
vettore1[5]           # seleziona il quinto elemento
```

```
## [1] 10
```

```
vettore1[c(5)]        # == vettore1[5]
```

```
## [1] 10
```

```
vettore1[c(-5)] # seleziona tutti gli elementi tranne il 5
```

```
## [1]  2  4  6  8 12 14 16 18 20
```

```
vettore1[c(FALSE,FALSE,FALSE,FALSE,TRUE,FALSE,FALSE,FALSE,FALSE,FALSE)]
```

```
## [1] 10
```

In questi esempi abbiamo visto alcune potenzialità dell'operatore di estrazione. Se l'indice è un vettore numerico, verranno estratti gli elementi corrispondenti agli indici del vettore. Se l'indice è di numeri negativi, verranno esclusi dall'estrazione gli elementi corrispondenti. Se l'indice è un vettore booleano della stessa lunghezza del vettore da cui si fa l'estrazione, verranno estratti gli elementi che corrispondono al valore `TRUE` dell'indice. Se agli elementi vettore di partenza è stato associato un nome, l'indice può essere un vettore stringa contenente i nomi degli elementi da estrarre.

```
names(vettore1) <- c("due","quattro","sei","otto","dieci","dodici","quatto
vettore1
```

```
##      due   quattro      sei     otto    dieci    dodici
##        2        4        6        8       10       12
## quattordici      sedici    diciotto       venti
##          14          16          18          20
```

```
vettore1["dieci"]
```

```
## dieci
##    10
```

```
vettore1[5]
```

```
## dieci
##    10
```

Riciclare i vettori

Cosa succede se il filtro booleano ha una lunghezza minore del vettore da filtrare?

```
vettore1[c(TRUE,TRUE,FALSE)]
```

```
##      due    quattro      otto     dieci quattordici    sedici
##        2         4         8        10        14        16
##       venti
##          20
```

Il vettore più corto viene *riciclato* - ovvero ripetuto tante volte da pareggiare la lunghezza del vettore più lungo.

- 20 Vectors | R for Data Science

Estrazione da matrici

```
matrice1 <- matrix(c(11,12,13,14,21,22,23,24,31,32,33,34),nrow=4,ncol=3)
colnames(matrice1) <- c("C_A", "C_B", "C_C")
rownames(matrice1) <- c("R1","R2","R3","R4")
matrice1
```

```
##    C_A C_B C_C
## R1  11  21  31
## R2  12  22  32
## R3  13  23  33
## R4  14  24  34
```

```
# estrae l'elemento della seconda riga, seconda colonna
matrice1[2,2]
```

```
## [1] 22
```

```
# estrae la prima e la terza colonna della prima riga
matrice1[1,c(1,3)]
```

```
## C_A C_C
##  11  31
```

```
# estrae tutta la terza riga
matrice1[3,]
```

```
## C_A C_B C_C
##  13  23  33
```

```
# estrae la riga "R2"
matrice1["R2",]
```

```
## C_A C_B C_C
##  12  22  32
```

```
# estrae righe e colonne per nome
matrice1["R3","C_A"]
```

```
## [1] 13
```

L'estrazione di elementi di una matrice è una estensione a due dimensioni dell'estrazione di un vettore. La sintassi è `matrice[indice righe, indice colonne]`. Per estensione, l'estrazione di un array tridimensionale sarà `array [indice dimensione 1, indice dimensione 2, indice dimensione 3]`. L'indice vuoto estrae l'intero vettore di quella dimensione.

```
# estrae tutti gli elementi del vettore
vettore1[]
```

```
##      due   quattro      sei     otto    dieci    dodici
##        2        4        6        8       10       12
## quattordici      sedici    diciotto       venti
##          14          16          18          20
```

```
# estrae tutte le righe della colonna 2
matrice1[,2]
```

```
## R1 R2 R3 R4
## 21 22 23 24
```

```
# estrae tutte le colonne della riga 2
matrice1[2,]
```

```
## C_A C_B C_C
##  12  22  32
```

```
# estrae l'intera matrice
matrice1[,]
```

```
##    C_A C_B C_C
## R1  11  21  31
## R2  12  22  32
## R3  13  23  33
## R4  14  24  34
```

```
(indici2d <- matrix(c(1,1,2,2,1,3,4,3),ncol = 2, byrow = TRUE))
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
## [3,]    1    3
## [4,]    4    3
```

```
matrice1[indici2d]
```

```
## [1] 11 22 31 34
```

indici2d è una matrice `r*2`. Ogni riga è una coppia di valori che corrisponde alla riga e alla colonna dell'elemento da estrarre. Dunque questa sintassi ha estratto gli elementi `matrice1[1,1]`, `matrice1[2,2]`, `matrice1[1,3]`, ‘matrice1[4,3].

Estrazione da liste

```
lista1 <- list(nomi=c("Marco","Alessia"), amici=c("Luigi","Francesco","Raffaella"),
lista1$nomi
```

```
## [1] "Marco"   "Alessia"
```

```
# la sintassi [[]] estrae i valori dell'oggetto selezionato
lista1[[2]]
```

```
## [1] "Luigi"     "Francesco" "Raffaella"
```

```
# la sintassi [] estrae l'oggetto
lista1[2]
```

```
## $amici
## [1] "Luigi"     "Francesco" "Raffaella"
```

Estrazione da data frame

```
nome <- c("luigi","mario","antonella","luca")
anno <- c(1956,1945,1972, 1976)
condizione <- c("exp","controllo","exp","controllo")
soggetti <- data.frame (nome,anno,condizione)
soggetti
```

```
##        nome anno condizione
## 1     luigi 1956        exp
## 2     mario 1945  controllo
## 3 antonella 1972        exp
## 4      luca 1976  controllo
```

```
# stessa sintassi delle matrici
# tutte le righe, colonna 3
soggetti[,3]
```

```
## [1] exp       controllo exp       controllo
## Levels: controllo exp
```

```
# riga 3, tutte le colonne
soggetti[3,]
```

```
##        nome anno condizione
## 3 antonella 1972        exp
```

```
# cella 3,3
soggetti[3,3]
```

```
## [1] exp
## Levels: controllo exp
```

```
# il data frame è una lista, e dunque vale anche la sintassi $
soggetti$anno
```

```
## [1] 1956 1945 1972 1976
```

```
# il data frame è una lista, e dunque posso selezionare per colonna
soggetti[2]
```

```
##   anno
## 1 1956
## 2 1945
## 3 1972
## 4 1976
```

```
# sposta in filtri
soggetti[soggetti$condizione==”exp”,]
```

```
##        nome anno condizione
## 1     luigi 1956        exp
## 3 antonella 1972        exp
```

```
subset(soggetti,condizione==”controllo”)
```

```
##    nome anno condizione
## 2 mario 1945  controllo
## 4  luca 1976  controllo
```

Filtrare i dati

Come abbiamo visto parlando di estrazione da vettori, gli indici possono essere dei vettori logici. I vettori logici possono essere creati attraverso l'applicazione di operatori logici ai vettori.

```
vettore_logico1 <- vettore1 > 10
vettore1[vettore_logico1]
```

```
##     dodici quattordici     sedici   diciotto      venti
##         12         14         16         18         20
```

+++todo+++ approfondire

Usare la sintassi sql

sqldf package | R Documentation

```
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Loading required package: RSQLite
```

```
# seleziona tutte le variabili (*) di soggetti dove la condizione è 'exp'
risultato <- sqldf(”select * from soggetti where condizione == 'exp'”)
risultato
```

```
##        nome anno condizione
## 1     luigi 1956        exp
## 2 antonella 1972        exp
```

```
rm(risultato)
# seleziona la variabile 'nome' da soggetti dove anno è maggiore di 1970
sqldf("select nome from soggetti where anno>1970")
```

```
##        nome
## 1 antonella
## 2      luca
```

Which

Dato un filtro booleano, la funzione `which()` ritorna l'indice di quei valori che corrispondono a TRUE

```
vettore1
```

```
##      due    quattro      sei     otto     dieci    dodici
##        2        4        6        8       10       12
## quattordici      sedici    diciotto       venti
##          14          16          18          20
```

```
# quali valori sono uguali a 4
which(vettore1==4)
```

```
## quattro
##       2
```

```
# l'indice dei valori superiori a 7
which(vettore1>7)
```

```
##     otto     dieci    dodici quattordici    sedici   diciotto
##        4        5        6        7        8        9
##       venti
##          10
```

```
# quale cella ha il valore massimo?
which(vettore1==max(vettore1))
```

```
## venti
##    10
```

Riferimenti

- Subsetting - Advanced R.
- Extracting elements - psyr.org
- Subsetting - R Programming for Data Science
- Indexing - R Language Definition
- Subset Data in R with R Subset Examples - RProgramming.net
- indexing - Is there an R function for finding the index of an element in a vector? - Stack Overflow

Capitolo 4

Le funzioni

Funzioni

R mette a disposizione numerose funzioni. Inoltre, è possibile scaricare, installare e richiamare delle librerie esterne, che mettono a disposizione altri insiemi di funzioni.

Per iniziare a *giocare* con R, introduciamo le funzioni length, min, max, sum, mean.

`length(v)` serve per conoscere la lunghezza (il numero di elementi) del vettore `v`. `min(v)` ritorna il valore più basso di un vettore `v`, `max()` il più alto. `sum(v)` somma i valori del vettore. `prod(v)` calcola il prodotto dei valori. `mean(v)` calcola la media, `sd(v)` la deviazione standard

```
variabile1 <- c(2,3,6,4,8,4,1)
length(variabile1)
```

```
## [1] 7
```

```
min(variabile1)
```

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

```
max(variabile1)
```

```
## [1] 8
```

```
# somma tutti i valori
sum(variabile1)
```

```
## [1] 28
```

```
# moltiplica tutti i valori
prod(variabile1)
```

```
## [1] 4608
```

```
# mean calcola la media
mean(variabile1)
```

```
## [1] 4
```

```
# sd calcola la deviazione standard
sd(variabile1)
```

```
## [1] 2.380476
```

Generare delle sequenze

La funzione seq genera delle sequenze. La sintassi più semplice, `seq(from,to)` crea una sequenza dal primo valore all'ultimo con *step* 1. Il comando `c(from:to)` è del tutto equivalente, ed utilizza il cosiddetto Colon Operator.

`seq(0,1000, length=11)` genera una sequenza di 11 numeri da 0 a 1000: 0, 100, 200 ... 1000.

```
(vettore1 <- seq(1,10))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
# seq (n) è uguale a seq(1,n)
seq(10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
# c(a:b) == seq(a,b)
c(1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
# genera una sequenza di lunghezza length
seq(0,1000, length=11)
```

```
##  [1]   0  100  200  300  400  500  600  700  800  900 1000
```

Ripetizione

La funzione `rep(v,n)` ripete un vettore `n` volte.

```r
# ripete 2 10 volte
rep(2,times=10)
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2
```

```r
rep(c(2,4),times=10)
```

```
##  [1] 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4 2 4
```

```r
# crea un vettore di lunghezza 10
rep(c(2,4),length=10)
```

```
##  [1] 2 4 2 4 2 4 2 4 2 4
```

```r
# concateno due sequenze
sequenze1 <- c(seq(0,1000, length=11),seq(1000,0, length=11))
sequenze1
```

```
## [1]   0 100 200 300 400 500 600 700 800 900 1000 1000  900  800
## [15]  700  600  500  400  300  200  100    0
```

`rep(c(2,4,3),times=10)` ripete 10 volte i valori 2, 4 e 3, creando un vettore di 30 elementi. `rep(c(2,4),length=10)` crea un vettore di lunghezza 10 (dunque. `rep_len(x, n)` equivale a `rep(x,length=n)`

- R: Replicate Elements of Vectors and Lists

sample

La funzione `sample(v)` *mescola* l'ordine del vettore `v`. La funzione `sample(v,n)` estrae `n` valori, in ordine casuale, dal vettore `v`. `sample` *estrae* casualmente i valori dal vettore - la metafora è quella di un'urna da cui vengono estratti i *numeri*. Se `replace=FALSE` un valore estratto non viene re-immesso nell'urna. Con `replace=TRUE` viene re-immesso, e può essere estratto di nuovo.

```r
# mescola il vettore
(mischiato <- sample (vettore1))
```

```
##  [1]  3  6  2  5  4  9  1  7 10  8
```

```
# sample (v,n) fa il sample e restituisce i primi n numeri
sample(vettore1, 4)
```

```
## [1]  6  9  4 10
```

```
# se lo rifaccio, avrò numeri differenti
sample(vettore1, 4)
```

```
## [1] 3 5 1 2
```

```
# con replace = TRUE i valori estratti vengono rimessi nell'urna
sample(1:4,10,replace = TRUE)
```

```
##  [1] 1 2 3 2 2 2 1 1 2 2
```

Ordinare un vettore

```
# ordina un vettore
# decreasing = TRUE ordine decrescente
sort(mischiato)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
order(mischiato)
```

```
##  [1]  7  3  1  5  4  2  8 10  6  9
```

```
mischiato[order(mischiato)]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

`sort(mischiato)` ordina il vettore, dal valore minore al maggiore. `sort(mischiato,decreasing=TR` per l'ordine inverso.

`order(mischiato)` ritorna la permutazione necessaria per ordinare mischiato

Rev

`rev(v)` *rovescia* l'ordine degli elementi del vettore.

```
# rev(x) gira x
rev(vettore1)
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

cbind, rbind

`cbind` e `rbind` creano delle matrici legando vettori o matrici in colonne (cbind) o in righe (rbind).

```
# rbind prende gli argomenti e li mette una riga dopo l'altra
rbind(c(1,2),c(3,4),c(5,6))
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

```
# incolonna gli argomenti
cbind(c(1,2),c(3,4),c(5,6))
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

all, any

`all(l)` e `any(l)` possono essere interpretate come delle estensioni degli operatori logici `&` e `|` (and e or).

`all(v)` restituisce `TRUE` solo se *tutti* i valori del vettore logico *v* sono veri. `any(v)` restituisce `TRUE` se *almeno uno* dei valori del vettore logico *v* sono veri.

```
# l'operatore logico & equivale alla funzione all (con 2 argomenti)
TRUE & FALSE
```

```
## [1] FALSE
```

```
all(TRUE,FALSE)
```

```
## [1] FALSE
```

```
# l'operatore logico | equivale alla funzione any (con 2 argomenti)
TRUE | FALSE
```

```
## [1] TRUE
```

```
any(TRUE,FALSE)
```

```
## [1] TRUE
```

```
(logico1 <- mischiato>5)
```

```
##  [1] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
```

```
all(logico1)
```

```
## [1] FALSE
```

```
any(logico1)
```

```
## [1] TRUE
```

Operazioni su insiemi

+++todo+++ spiegare meglio

```
(vettore2 <- c(seq(6,15),rep(8,3)))
```

```
##  [1]  6  7  8  9 10 11 12 13 14 15  8  8  8
```

```
# intersect (v1, v2) restituisce i valori in comune
intersect(mischiato, vettore2)
```

```
## [1]  6  9  7 10  8
```

```
# tutti i valori che appaiono almeno una volta nell'unione dei due vettori
union(mischiato, vettore2)
```

```
##  [1]  3  6  2  5  4  9  1  7 10  8 11 12 13 14 15
```

```
# setdiff(v1,v2): i valori di v1 NON presenti in v2
setdiff(mischiato, vettore2)
```

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

```
# restituisce TRUE se i due vettori hanno gli stessi elementi
setequal(mischiato, vettore2)
```

```
## [1] FALSE
```

```
setequal(mischiato, vettore1)
```

```
## [1] TRUE
```

```
# perché?
setequal(mischiato, c(vettore1, 5,6,7,2))
```

```
## [1] TRUE
```

Aritmetica delle matrici

R permette sofisticate operazioni di aritmetica dei vettori e delle matrici. Riportiamo gli esempi più semplici.

```
vettore1 <- c(1:9)
vettore1*2
```

```
## [1]  2  4  6  8 10 12 14 16 18
```

Apparentemente, `vettore1*2` moltiplica vettore1 per lo scalare 2. In realtà, `2` non è uno scalare, ma un vettore di lunghezza 1, che viene *riciclato* per la lunghezza di `vettore1`. La logica diviene evidente se moltiplichiamo `vettore1` per un vettore di lunghezza 3.

```
vettore2 <- c(1,2,3)
vettore1*vettore2
```

```
## [1]  1  4  9  4 10 18  7 16 27
```

Come si può vedere, `vettore2` è stato riusato 3 volte per moltiplicare `vettore1`.

Questa operazione viene definita *element-wise multiplication*, e si distingue dalla moltiplicazione di matrici, che si ottiene con la sintassi `A %*% B`, ma che qui non verrà trattata.

Vediamo alcuni esempi con le matrici

```
matrice1 <- matrix (c(5,7,1,6,9,5,2,1,9,5,7,5), nrow =3, ncol=4)
matrice1
```

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

```
matrice1 * 3
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   15   18    6   15
## [2,]   21   27    3   21
## [3,]    3   15   27   15
```

```
# rep (10,12) ripete il numero 10 per 12 volte
matrice2 <- matrix (rep(10,12),nrow=3, ncol=4)
matrice2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   10   10   10   10
## [2,]   10   10   10   10
## [3,]   10   10   10   10
```

```
# sommo ogni elemento di matrice1 al
# corrispondente di matrice2
matrice1+matrice2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   15   16   12   15
## [2,]   17   19   11   17
## [3,]   11   15   19   15
```

```
# sottraggo, elmento per elemento
matrice1-matrice2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -5   -4   -8   -5
## [2,]   -3   -1   -9   -3
## [3,]   -9   -5   -1   -5
```

```
# divido
matrice1/matrice2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  0.5  0.6  0.2  0.5
## [2,]  0.7  0.9  0.1  0.7
## [3,]  0.1  0.5  0.9  0.5
```

```
matrice3 <- matrix(c(2,3,4,3,2,6,7,8,5,9,5,6),nrow=4,ncol=3, byrow=TRUE)
matrice3
```

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

```
## [4,]    9    5    6
```

```
# traspongo matrice2 e la sommo a matrice3
t(matrice2)+matrice3
```

```
##      [,1] [,2] [,3]
## [1,]   12   13   14
## [2,]   13   12   16
## [3,]   17   18   15
## [4,]   19   15   16
```

- Elementary arithmetic operations -R Language Definition
- Quick-R: Matrix Algebra

Programmazione funzionale

Functional programming · Advanced R.

le funzioni *apply

Data una matrice, la funzione `apply(matrice,margine,funzione)` applica la funzione ad ogni cella delle righe o delle colonne della matrice.

```
matrice1 <- matrix(sample(1:10,20,replace = TRUE),nrow = 5, ncol = 4)
matrice1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    7    4    4    5
## [2,]    4    7    9    7
## [3,]    6    3   10   10
## [4,]    9    8    1    1
## [5,]    1    6    4    8
```

```
# calcola la somma dei valori di ogni riga
apply(matrice1, 1, sum)
```

```
## [1] 20 27 29 19 19
```

```
# calcola la media di ogni colonna
apply(matrice1, 2, mean)
```

```
## [1] 5.4 5.6 5.6 6.2
```

`sapply(lista,funzione)` applica la funzione agli elementi della lista

```
lista1 <- list(a = 1:10, b = 11:20)
lista1
```

```
## $a
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $b
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
sapply(lista1, mean)
```

```
##    a    b
##  5.5 15.5
```

```r
replicate(3, c(1:5))
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    2    2    2
## [3,]    3    3    3
## [4,]    4    4    4
## [5,]    5    5    5
```

Riferimenti

- Vocab · Advanced R.
- adv-r-book-solutions/functions.r at master · peterhurford/adv-r-book-solutions
- R useful functions
- Useful R base functions - Teaching - RStudio Community
- r - Grouping functions (tapply, by, aggregate) and the *apply family - Stack Overflow
- Using aggregate and apply in R - Dave Tang's blog
- A brief introduction to "apply" in R | What You're Doing Is Rather Desperate
- R Apply Family
- How to Aggregate Data in R | R-bloggers
- r - How to sum a variable by group? - Stack Overflow

Capitolo 5

Programmazione

Programmazione

Nelle lezioni precedenti abbiamo utilizzato R come un ambiente statistico a linea di comando, dove far *girare* degli script, invocando delle funzioni.

In questa lezione ci focalizzeremo sulla creazione di funzioni.

Ciao

Iniziamo con un esempio molto semplice: la creazione della funzione `ciao()`, il cui scopo è quello di salutare.

```
ciao <- function() {
    messaggio <- "ciao a tutti"
    return (messaggio)
}
```

Invochiamo la funzione

```
ciao()
```

```
[1] "ciao a tutti"
```

Passaggio di un argomento

La versione seguente richiede un parametro (un nome) che verrà usato per personalizzare il saluto

```
ciaoNome <- function (nome) {
    messaggio <- paste("ciao ", nome, ", come stai?", sep="" )
    return (messaggio)
}
```

Se invochiamo la funzione senza il parametro, R segnalerà un errore.

```
ciaoNome("Francesco")
```

```
[1] "ciao Francesco, come stai?"
```

```
# ciaoNome() # il parametro è obbligatorio
```

Valore di default

In questa iterazione, assegnamo al parametro `nome` un valore di default, che verrà usato se, nell'invocazione, non verrà specificato alcun valore.

```
ciaoNomeAmico <- function (nome="amico") {
    messaggio <- paste("ciao ", nome, ", come stai?", sep="" )
    return (messaggio)
}

ciaoNomeAmico("Marika")
```

```
[1] "ciao Marika, come stai?"
```

```
ciaoNomeAmico()
```

```
[1] "ciao amico, come stai?"
```

If

In questa iterazione gestiamo la presenza o meno del parametro nell'invocazione.
Se il parametro non c'è, salutiamo l'anonimo.

```
ciaoNomeForse <- function (nome=NULL) {
    messaggio <- ""
    if(!is.null(nome)) {
        messaggio <- paste("ciao ", nome, ", come stai?", sep="" )
    } else {
        messaggio <- "Ciao anonimo ;)"
    }
    return (messaggio)
}
```

```
ciaoNomeForse("Matteo")
```

```
[1] "ciao Matteo, come stai?"
```

```
ciaoNomeForse()
```

```
[1] "Ciao anonimo ;)"
```

For

In questa iterazione, gestiamo un vettore di più nomi, introducendo il ciclo `for`.

```
ciaoNomi <- function (nomi=c()) { # c() vettore vuoto
    messaggio <- "ciao "
    for (nome in nomi) {
        messaggio <- paste(messaggio, nome, ", ", sep="")
    }
    messaggio <- paste(messaggio, "come state?", sep="")
    return (messaggio)
}

nomi <- c("Mario", "Antonella", "Lucia")
ciaoNomi(nomi)
```

```
[1] "ciao Mario, Antonella, Lucia, come state?"
```

```
ciaoNomi()
```

```
[1] "ciao come state?"
```

If, else, for

Una versione più complessa, che gestisce un parametro (con valore di default, ovvero il vettore vuoto), e 4 diversi scenari: vettore vuoto, vettore con un solo valore, vettore con meno di 4 valori (di fatto 2 o 3 nomi), vettore con almeno 4 valori.

```
ciaoDipende <- function (nomi=c()) { # c() vettore vuoto
    if (length(nomi)==0) {
        return ("ciao anonimo, come stai? Sei timido?")
    }
    if (length(nomi)==1) {
        messaggio <- paste ("ciao ", nomi[1], ", come stai?",
                                                sep="")
        return (messaggio)
    }
    messaggio <- "ciao "
    if (length(nomi)<4) {
        for (nome in nomi) {
            messaggio <- paste(messaggio, nome, ", ", sep="")
        }
    } else {
        messaggio <- paste(messaggio, "a tutti, ", sep="")
    }
    messaggio <- paste(messaggio, "come state?", sep="")
    return (messaggio)
}

ciaoDipende(c("Giovanna", "Barbara"))

[1] "ciao Giovanna, Barbara, come state?"

ciaoDipende(c("Giovanna", "Barbara",
                          "Enrico", "Tommaso", "Gianluca"))

[1] "ciao a tutti, come state?"

ciaoDipende()

[1] "ciao anonimo, come stai? Sei timido?"

ciaoDipende("Enrico")

[1] "ciao Enrico, come stai?"
```

Funzioni

```
nomeFunzione <- function (argomento1=default, argomento2) {
    # corpo della funzione
    return(valore) # valore da sostituire

}
```

Funzione, corpo, argomenti

```
# il corpo della funzione
body(ciaoNomi)
```

```
{
    messaggio <- ”ciao ”
    for (nome in nomi) {
       messaggio <- paste(messaggio, nome, ”, ”, sep = ””)
    }
    messaggio <- paste(messaggio, ”come state?”, sep = ””)
    return(messaggio)
}
```

```
# gli argomenti della funzione
formals(ciaoNomi)
```

```
$nomi
c()
```

If

```
if (condizioneLogica) {
    # se la condizione è vera,
    # esegui il codice fra le parentesi
} else if (secondaCondizione) { # facoltativo
    # se la prima è falsa ma la seconda è vera
    # esegui la seconda
} else { # facoltativo
    # se le condizioni precedenti sono tutte false,
    # esegui invece questo codice
}
```

For

```
for (elemento in vettore) { # o sequenza
    # esegui questo codice
    # per ognuno degli elementi
}
```

Altre strutture di controllo

while: esegue un loop finché la condizione è vera

break: interrompe un loop (for, while, repeat)

repeat: loop infinito (è necessario interromperlo con un break)

next: passa alla iterazione successiva del loop

Esercizi

La media

Creare la funzione `media(vettore)` senza usare le funzioni `mean()` e `sum()`.

Test:

```
vettore1 <- sample(1:10,20, replace = TRUE)
media(vettore1)
```

```
[1] 5
```

```
mean(vettore1)
```

```
[1] 5
```

Lancio di dadi

Creare la funzione lancio (numDadi) che simula il lancio di `numDadi` dadi e la somma del loro risultato

Test

```
lancio(1)
```

```
[1] 1
```

```
lancio(1)
```

```
[1] 4
```

```
lancio(2)
```

```
[1] 12
```

```
lancio(2)
```

```
[1] 11
```

```
lancio(3)
```

```
[1] 8
```

Simulare mille lanci di un dado (salvando i risultati in `lanci1`), e mille lanci di due dadi (salvando i risultati in `lanci2`)

Creare un file in rMarkdown (es dadi.Rmd), inserire i chunk relativi alla funzione dati e allo script dei lanci, con una breve descrizione dei due passaggi.

Usando ggplot2, mostrare gli istogrammi dei due vettori `lanci1` e `lanci2` ed aggiungerli al file rMarkdown.

```
Attaching package: 'cowplot'
```

```
The following object is masked from 'package:ggplot2':

    ggsave
```

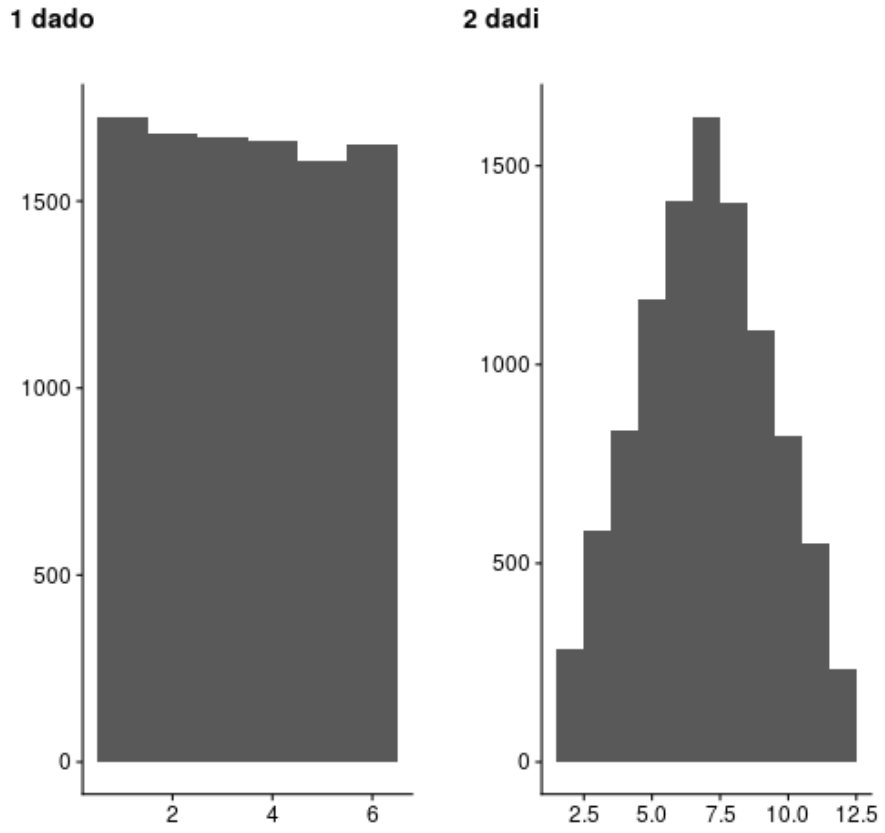


Figura 5.1: plot of chunk prog_lanci_his

Parte II

L'analisi dei dati

Capitolo 6

Pulire e sistemare i dati

Sistemare i dati

L'analisi dei dati è un processo finalizzato ad esplorare, ripulire, trasformare ed analizzare un insieme di dati.

Ripulire e sistematizzare i dati è il primo passaggio, necessario per poter procedere nelle analisi esplorative, descrittive, inferenziali.

La *pulizia* dei dati consiste in primo luogo nell'analizzare i dati grezzi per verificare che la loro codifica sia corretta, ed identificare e correggere eventuali problemi di codifica. Questo passaggio trasforma i dati *grezzi* in dati *tecnicamente corretti*.

In secondo luogo, questi dati vanno analizzati per verificare che la base di dati non contenga errori sostanziali. Questa *pulizia* trasforma i dati *tecnicamente corretti* in dati *consistenti*.

An introduction to data cleaning with R

La *trasformazione* / *sistematizzazione* dei dati consiste nell'organizzarli in maniera da facilitare la successiva analisi statistica.

Il passaggio successivo costituisce l'analisi vera e propria (descrittiva, esplorativa, inferenziale). Infine, i risultati dell'analisi vanno sintetizzati attraverso la creazione di report.

Una volta caricato il file di dati, è opportuno in primo luogo controllarne la correttezza, attraverso un processo di *pulizia*.

In secondo luogo, può essere necessario sistematizzare i dati se non sono rappresentati in forma canonica.

Pulizia dei dati

I dati sono *tecnicamente corretti* se

- sono caricati in un data frame, i cui nomi di colonne siano corretti e significativi
- ogni colonna è del tipo corretto: i dati numerici in vettori `numeric`, i dati testuali in vettori `character`, e le variabili categoriali in `factor` con livelli appropriati.

Spesso può essere utile pre-processare i dati grezzi con strumenti diversi da R, ad esempio con degli editor di testo.

Lo svantaggio di ripulire i dati a mano è che un processo di questo tipo non è facilmente automatizzabile, e dunque replicabile.

+++todo+++ conversione di tipo

Se le colonne contengono delle date, può essere necessario convertire il testo che rappresenta le date nei formati `POSIXlt` o `POSIXct`, ad esempio con la funzione `strptime`.

Verifica di consistenza

Il passaggio successivo consiste nel verificare che i dati, che da grezzi sono stati trasformati in *tecnicamente corretti*, siano anche *consistenti*. Si tratta di verificare se ci sono degli errori o dei problemi a livello di contenuti, ovvero valutare se ci sono dei dati esplicitamente sbagliati, dei dati incompleti o compromessi, degli outlier fuori norma. Questi dati vanno corretti se possibile, oppure eliminati se necessario.

La verifica di consistenza è un processo che prevede

- la definizione di vincoli che definiscono quel tipo di variabile o osservazione
- l'identificazione di dati che violano i vincoli
- la correzione dei dati, o loro eliminazione

Particolare attenzione va prestata ai missing (NA).

In questo contesto gli outlier sono osservazioni o insiemi di osservazioni che appaiono inconsistenti con il set di dati. Gli outlier vanno analizzati, per valutare se vi sono motivi per considerarli delle violazioni dei vincoli o se - semplicemente - sono delle osservazioni che si collocano ai margini delle distribuzioni.

Alcune inconsistenze appaiono più palesi. Ad esempio, l'età di una persona non può essere negativa.

An introduction to data cleaning with R

Un esempio

Per spiegare la necessità di *pulire i dati*, può essere utile fare un esempio.

Global Footprint Network è una organizzazione no profit che si occupa di sostenibilità ambientale. Nel sito dell'organizzazione vi è una sezione open data, dove sono riportate delle statistiche relative all'*impronta ecologica* dei paesi del mondo. La pagina Sustainable Development confronta tre indici relativi a 178 paesi:

1. lo *Human Development Index (HDI)*, sviluppato dalle nazioni unite
2. l'impronta ecologica, definita in termini di area biologicamente produttiva di mare e di terra necessaria a rigenerare le risorse consumate da una popolazione umana (o - nel caso dei dati, pro-capite) e ad assorbire i rifiuti prodotti
3. la popolazione di quel paese.

Possiamo usare `read.csv()` per scaricare la tabella di dati

```
sustain <- read.csv("https://s3.eu-central-1.amazonaws.com/bussolon/dati/Sustainabl
str (sustain)
```

```
## 'data.frame':    184 obs. of  5 variables:
## $ Country.Name: chr  "Armenia" "Afghanistan" "Albania" "Algeria" ...
## $ countryName : chr  "Armenia" "Afghanistan" "Albania" "Algeria" ...
##  $ EFConsPerCap: chr  "2.02" "0.77" "2.14" "2.45" ...
##  $ HDI         : chr  "0.74" "0.48" "0.76" "0.74" ...
## $ Population  : num  3006154 31627506 2889676 38934336 24227524 ...
```

Attraverso la funzione `str()` possiamo analizzare la struttura del data frame. Due cose saltano all'occhio. La prima: `Country.Name` sembra uguale a `countryName`

o, quantomeno, le due colonne appaiono ridondanti. La seconda, forse meno evidente, è che anche le colonne apparentemente numeriche (`EFConsPerCap` e `HDI`) vengono *viste* da r come testo (`chr`).

Utilizzando la funzione `View(sustain)` è possibile visualizzare il data frame. Si scopre così che:

a. le prime due colonne sono davvero uguali
b. il nome di alcuni paesi si è spalmato sulle colonne adiacenti, sporcando i dati.

Diventa dunque necessario quantomeno ripulire le righe in cui i nomi si sono spalmati nelle colonne adiacenti; in secondo luogo, è opportuno eliminare una delle prime due colonne.

Per ripulire le righe, l'approccio più ovvio può essere quello di aprire il file con excel o libre office. Ma forse la soluzione più semplice è quella di aprire il file con un editor di testo. Se andiamo alla riga 70 vediamo che l'Iran è definito come `Iran, Islamic Republic of`.

```
Iran, Islamic Republic of,Iran, Islamic Republic of,3.4,0.77,78143640
```

Ma la virgola viene interpreatata come separatore di campo (comma separated). La cosa più semplice è cancellare le virgole di troppo.

Una volta fatta questa prima correzione, è possibile ricaricare il file per verificare che il problema sia risolto.

```
sustainable <- read.csv("/home/bussolon/documenti/didattica/r_dottorato/da
str(sustainable)
```

```
## 'data.frame':    178 obs. of  5 variables:
## $ Country.Name: Factor w/ 178 levels "Afghanistan",..: 7 1 2 3 4 5 6 8 9 11 .
## $ countryName : Factor w/ 178 levels "Afghanistan",..: 7 1 2 3 4 5 6 8 9 11 .
## $ EFConsPerCap: Factor w/ 160 levels "0.5","0.57","0.6",..: 62 6 67 74 40 1
## $ HDI         : num  0.74 0.48 0.76 0.74 0.53 0.78 0.83 0.94 0.89 0.79 ...
## $ Population  : num  3006154 31627506 2889676 38934336 24227524 ...
```

```
levels(sustainable$EFConsPerCap)
```

```
##   [1] "0.5"     "0.57"    "0.6"     "0.67"    "0.76"
##   [6] "0.77"    "0.78"    "0.79"    "0.82"    "0.85"
##  [11] "0.87"    "0.95"    "0.96"    "0.97"    "0.98"
##  [16] "1.01"    "1.03"    "1.04"    "1.09"     "1.1"
##  [21] "1.11"    "1.12"    "1.19"    "1.2"     "1.21"
##  [26] "1.22"    "1.23"    "1.27"    "1.28"     "1.3"
##  [31] "1.31"    "1.32"    "1.36"    "1.46"    "1.47"
##  [36] "1.48"    "1.53"    "1.54"    "1.55"    "1.56"
##  [41] "1.59"    "1.61"    "1.63"    "1.64"    "1.66"
##  [46] "1.73"    "1.75"    "1.76"    "1.77"    "1.78"
##  [51] "1.79"    "1.84"    "1.85"    "1.9"     "1.91"
##  [56] "1.93"    "1.96"    "1.98"    "12.28"   "15.65"
##  [61] "2"       "2.02"    "2.04"    "2.05"    "2.07"
##  [66] "2.09"    "2.14"    "2.17"    "2.29"     "2.3"
##  [71] "2.32"    "2.39"    "2.4"     "2.45"    "2.49"
##  [76] "2.51"    "2.53"    "2.54"    "2.55"    "2.72"
```

```
##  [81] "2.8"      "2.84"     "2.87"     "2.92"     "2.94"
##  [86] "2.96"     "2.98"     "3.02"     "3.07"     "3.08"
##  [91] "3.1"      "3.17"     "3.21"     "3.27"     "3.29"
##  [96] "3.3"      "3.32"     "3.35"     "3.4"      "3.42"
## [101] "3.5"      "3.55"     "3.6"      "3.63"     "3.64"
## [106] "3.68"     "3.69"     "3.71"     "3.8"      "3.81"
## [111] "3.9"      "4.03"     "4.18"     "4.2"      "4.29"
## [116] "4.33"     "4.42"     "4.44"     "4.64"     "4.68"
## [121] "4.69"      "4.7"      "4.71"     "4.74"      "4.8"
## [126] "4.82"     "4.85"     "4.89"     "5.05"     "5.19"
## [131] "5.55"     "5.56"     "5.57"     "5.6"      "5.63"
## [136] "5.68"     "5.8"      "5.81"     "5.82"     "5.84"
## [141] "5.86"     "5.88"     "5.92"      "6"       "6.03"
## [146] "6.09"     "6.32"     "6.59"     "6.69"     "6.71"
## [151] "6.89"     "6.97"     "7.13"     "7.65"     "8.05"
## [156] "8.37"    "8.71"     "9.5"     "9.75"    "undefined"
```

Il prossimo passaggio consiste nell'eliminare una delle due colonne. Il passaggio successivo consiste nel capire perché la colonna relativa all'impronta ecologica (`EFConsPerCap`) sia vista come fattore. Analizzando i *livelli* si scopre che i *missing* sono etichettati come `undefined`.

```
sustainable$EFConsPerCap[sustainable$EFConsPerCap=="undefined"] <- NA
sustainable$EFConsPerCap <- as.numeric(sustainable$EFConsPerCap)
```

È dunque necessario trasformare gli `undefined` in `NA`, e poi convertire il vettore da fattore a numerico, con la funzione `as.numeric()`.

A questo punto il dataframe è tecnicamente corretto.

Trasformare i dati

+++todo+++

- Pulire i dati (scrub, wrangle) data-wrangling-cheatsheet - data-wrangling-cheatsheet.pdf - direi che fatto as old style, devi imparare (e scrivere) new style (tidyverse)
  - la logica di R (righe=osservazioni, colonne = variabili)
  - Pick observations by their values (filter()). R for Data Science - Exploratory Data Analysis with R
  - Reorder the rows (arrange()).
  - Pick variables by their names (select()).
  - Create new variables with functions of existing variables (mutate()).
  - Collapse many values down to a single summary (summarise()). https://psyr.org/manipulating-data.html

Risorse

- Tidy Data ottima introduzione accademica
- sfirke/janitor: simple tools for data cleaning in R da valutare

- Cleaning Data In R - Learn With Our Online Course | DataCamp
- r - Organized processes to clean data - Data Science Stack Exchange
- An introduction to data cleaning with R
- RPubs - Cleaning Data in R
- 12 Tidy data | R for Data Science
- data-wrangling-cheatsheet

Capitolo 7

L'analisi descrittiva

L'analisi descrittiva

Finalità

Come è stato detto nel capitolo di introduzione metodologica, le statistiche descrittive sono finalizzate a:

- avere una prima visione, qualitativa, delle variabili raccolte;
- controllare la presenza di errori, ad esempio di data-entry;
- far emergere outliers e anomalie;
- valutare qualitativamente ipotesi e assunti, determinare qualitativamente le relazioni fra le variabili;
- identificare l'entità e la direzione delle relazioni fra le variabili;
- selezionare i modelli statistici appropriati;

Sappiamo inoltre che si usano indicatori e strumenti diversi in base alla tipologia delle variabili (categoriali, ordinali, quantitative) ed in base al numero di variabili prese in considerazione (univariate, bivariate, multivariate).

Analizziamo ora le funzioni più comuni nell'analisi descrittiva.

Variabili categoriali

Tabelle di contingenza

Le tabelle di contingenza permettono di rappresentare la distribuzione di frequenza di variabili categoriali e di fattori.
In R, si usa la funzione `table(variabile)` per la rappresentazione univariata, e `table(variabile1, variabile2)` per la rappresentazione bivariata.

Barplot

Un metodo grafico per visualizzare la distribuzione di frequenza di una variabile categoriale o ordinale è il barplot, usando la forma `barplot(table(variabile))`.

La moda

In R non esiste una funzione per calcolare la moda, ovvero la categoria (o il valore, in caso di variabili numeriche) con la frequenza più alta. Per calcolarla, si identifica il valore più alto della tabella delle frequenze.

```
# creiamo una variabile nominale con 3 livelli
nominale <- factor(sample (c("rosso","bianco","verde"), 100,replace = TRUE
# la tabella delle frequenze
(frequenze<-table(nominale))
```

```
## nominale
## bianco  rosso  verde
##     35     34     31
```

```r
# calcoliamo la moda
moda<-which(frequenze == max(frequenze))
names(frequenze)[moda]
```

```
## [1] "bianco"
```

```r
# il grafico a barre
barplot(frequenze)
```

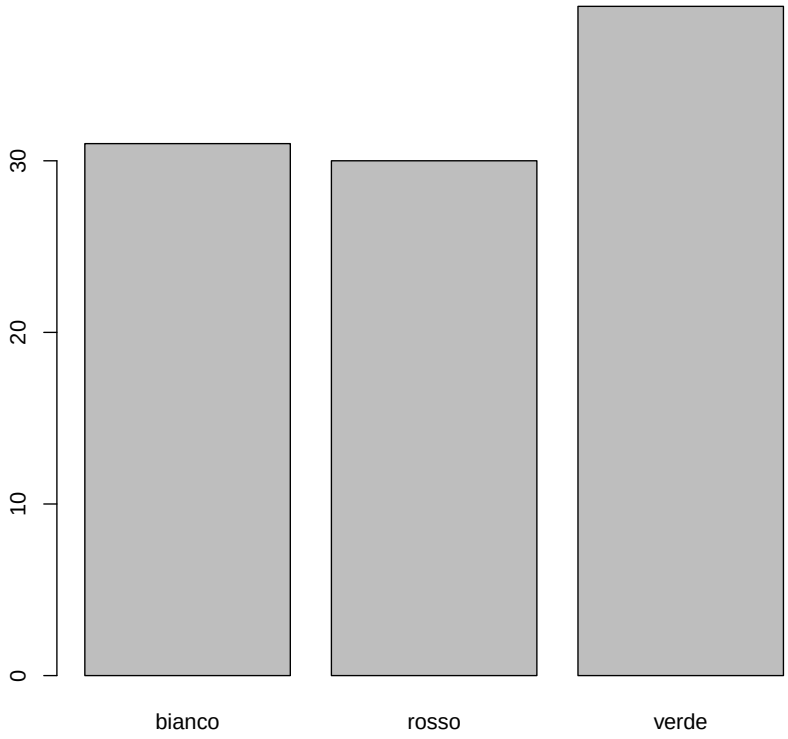


Figura 7.1: plot of chunk desc_nominale_1

Variabili ordinali

Gli indici che si possono calcolare con le variabii ordinali sono, oltre al numero di livelli e la moda, anche il minimo, il massimo, la mediana, i quartili ed il range interquantile.

Come abbiamo visto nel capitolo dedicato alle tipologie di dati, in R le variabili ordinali vanno rappresentate come fattori ordinati.

La funzione `summary()` però, se applicata ad un fattore, non restituisce questi dati, perché non vede il fattore come variabile numerica. Per aggirare l'ostacolo, è necessario utilizzare il vettore numerico sottostante, attraverso la chiamata alla funzione `as.integer()`.

```
# creiamo una variabile ordinale con 5 livelli
cat_ord <- c("A","B","C","D", "E")
ordinale <- factor(sample (cat_ord, 100,replace = TRUE), levels = cat_ord,
# summary su ordinale
summary(ordinale)
```

```
##  A  B  C  D  E
## 24 21 22 12 21
```

```
# calcolare i quartili sul vettore sottostante
quantile(as.integer(ordinale))
```

```
##   0%  25%  50%  75% 100%
##    1    2    3    4    5
```

Variabili a intervalli

Oltre a moda, mediana, minimo, massimo e quartili, per le variabili ad intervalli si calcolano la media e la varianza / deviazione standard.

Dal punto di vista grafico, si usano le funzioni grafiche `boxplot`, e `hist` per rappresentare l'istogramma delle frequenze

```
# generiamo una variabile numerica con distribuzione normale
# 100 osservazioni, media=10, ds=2
intervalli <- rnorm(100, mean=10, sd=2)
summary(intervalli)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.816   8.769   9.916  10.101  11.148  14.780
```

```
sd(intervalli)
```

```
## [1] 1.956492
```

```
boxplot(intervalli)
```

```
hist(intervalli)
```

Due variabili categoriali

Il rapporto fra due variabili categoriali (nominali o ordinali) può essere rappresentato attraverso la tabella di contingenza a due vie. Graficamente, il rapporto può essere rappresentato attraverso il grafico `mosaicplot`.

```
(contingenza <- table(nominale, ordinale))
```

```
##         ordinale
## nominale  A  B  C  D  E
```

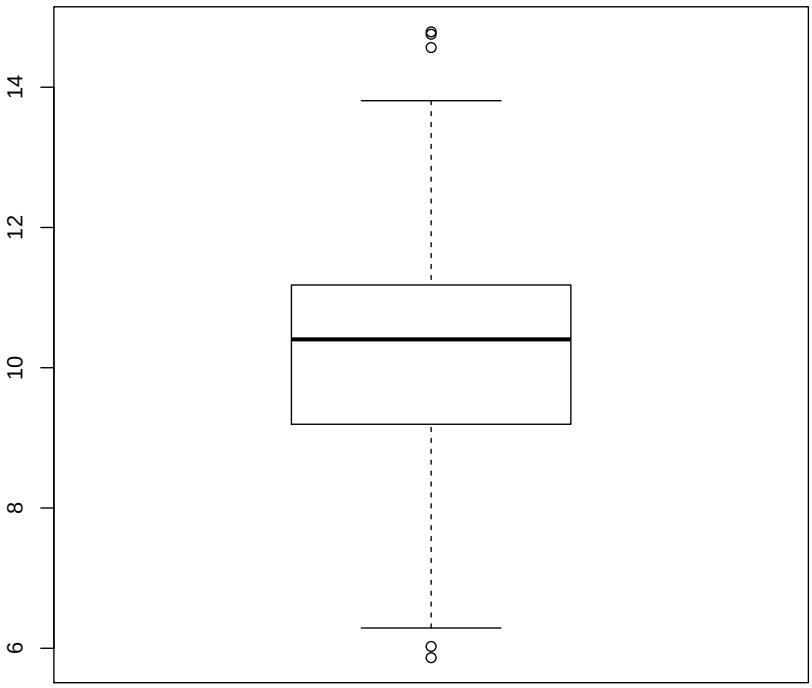


Figura 7.2: plot of chunk desc_intervalli

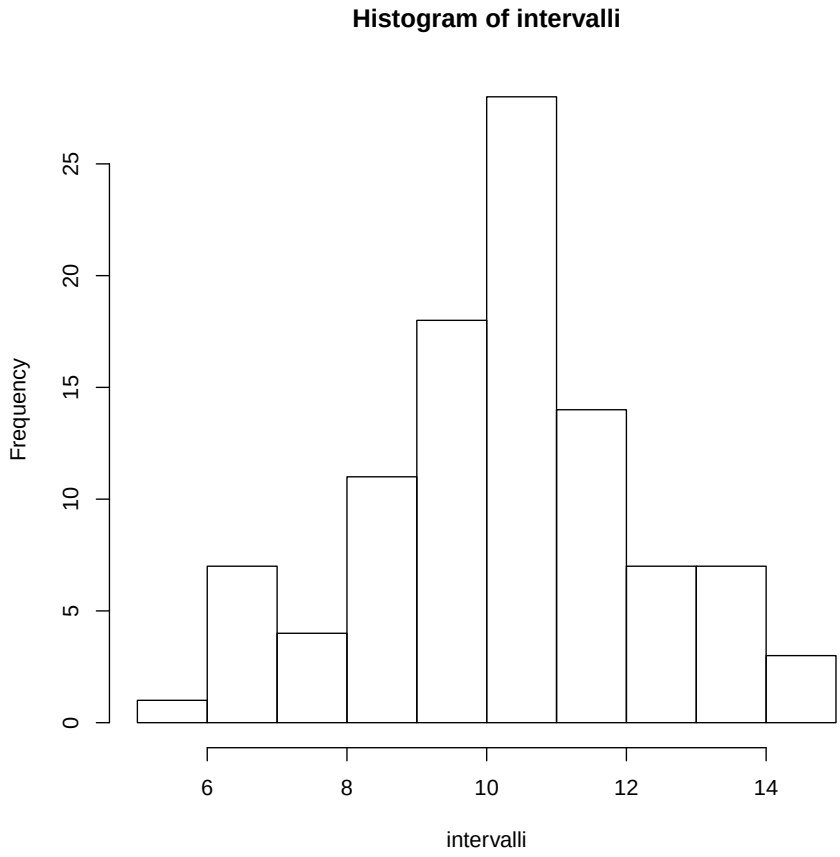


Figura 7.3: plot of chunk desc_intervalli

```
##   bianco  5  7  8  4 11
##   rosso  11  8  6  3  6
##   verde   8  6  8  5  4
```

```
mosaicplot(contingenza)
```

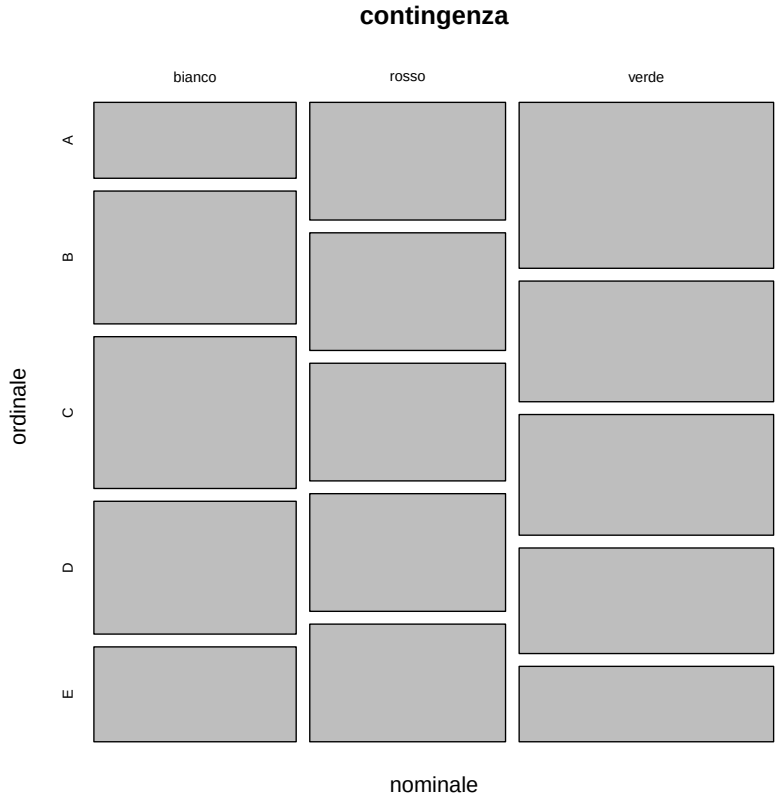


Figura 7.4: plot of chunk desc_contingenza

Una variabile categoriale, una numerica

By

`by()` è una funzione della famiglia di `apply`. La logica di `by` è di separare le righe un data frame, una matrice o un vettore in base ad un fattore. Ad esempio, la funzione `by(intervalli,ordinale,mean)` divide la variabile `intervalli` in cinque gruppi, in base al fattore `ordinale`, e per ogni gruppo applica la media.

```
by(intervalli,ordinale,mean)
```

```
## ordinale: A
```

```
## [1] 10.01521
## ------------------------------------------------------------
--
## ordinale: B
## [1] 10.67687
## ------------------------------------------------------------
--
## ordinale: C
## [1] 10.10872
## ------------------------------------------------------------
--
## ordinale: D
## [1] 9.614963
## ------------------------------------------------------------
--
## ordinale: E
## [1] 9.892453
```

Boxplot

Boxplot può essere usato per confrontare una variabile ad intervalli su una variabile categoriale (nominale o ordinale).

```
boxplot(intervalli ~ ordinale)
```

Due variabili numeriche

In caso di due variabili numeriche, la rappresentazione grafica è il grafico di dispersione. Se dal grafico appare che vi sia una correlazione, può essere utile calcolare la regressione lineare, con la funzione `lm(y ~ x)` e disegnare la retta di regressione, con la funzione `abline()`.

```
# creo una seconda variabile ad intervalli
# che *correla* con la prima
intervalli2 <- intervalli + rnorm(100, mean=2, sd=3)
# calcolo la regressione lineare
lineare <- lm(intervalli2 ~ intervalli)
# il grafico di dispersione
plot(intervalli,intervalli2)
# disegno la retta di regressione
abline(lineare)
```

Summarytools

è un pacchetto bla bla

```
library(summarytools)
#st_options(bootstrap.css = FALSE,     # Already part of the theme
#           footnote = NA,
```

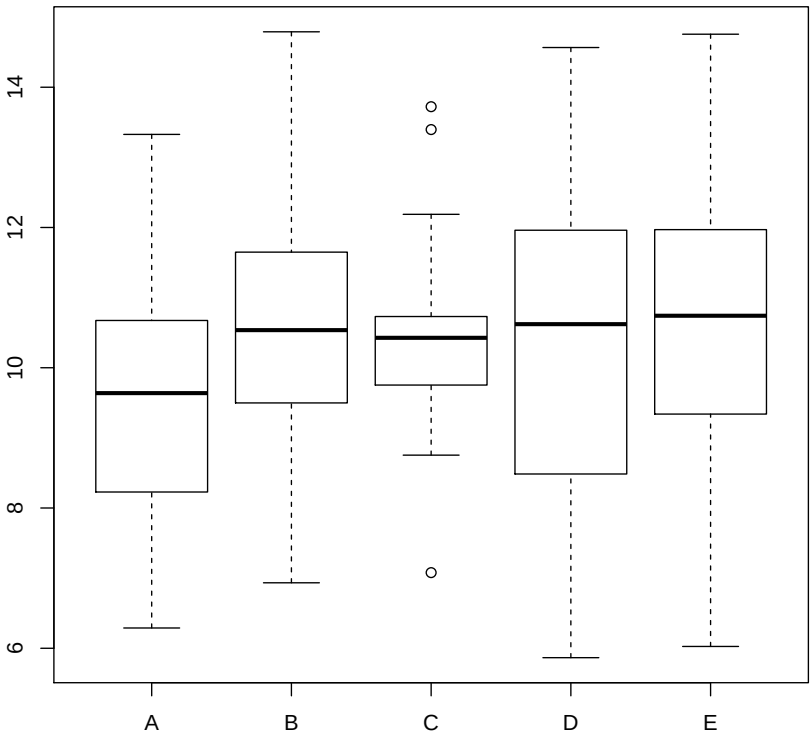


Figura 7.5: plot of chunk desc_boxplot_intervalli_ordinale

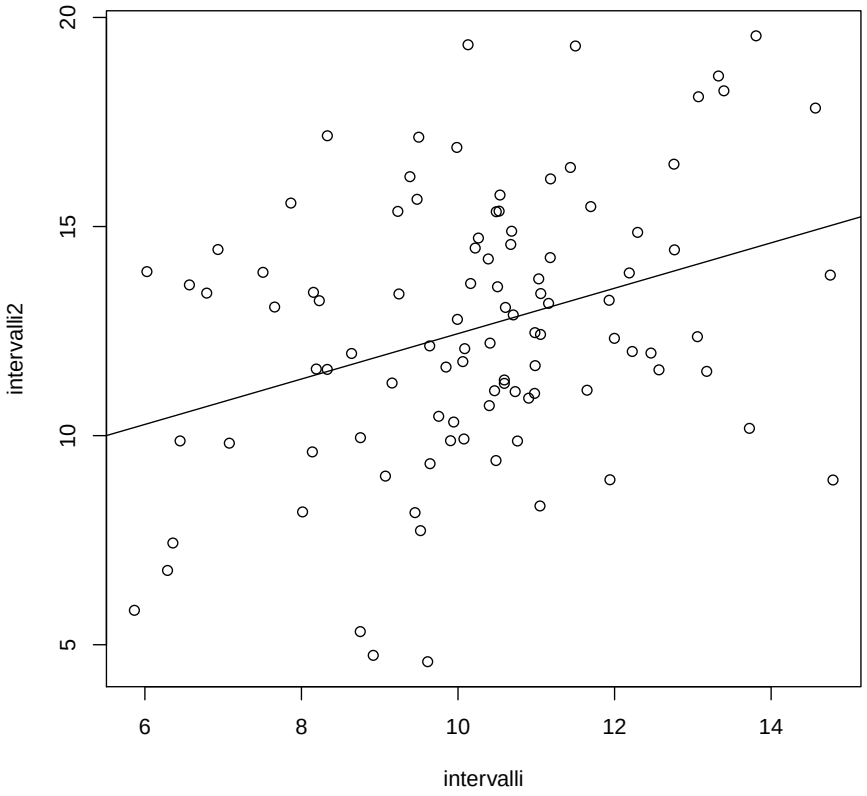


Figura 7.6: plot of chunk desc_intervalli2

```
#           subtitle.emphasis = FALSE)
#knitr::opts_chunk$set(comment=NA, prompt=FALSE, cache=FALSE, echo=TRUE, results='a
# freq(nominale, style = "rmarkdown")

ctable(nominale, ordinale, style = "rmarkdown")
```

Cross-Tabulation, Row Proportions

nominale * ordinale

ordinale	A	B	C	D	E	Total
nominale						
bianco	5 (14.3%)	7 (20.0%)	8 (22.9%)	4 (11.4%)	11 (31.4%)	35 (100.0%)
rosso	11 (32.4%)	8 (23.5%)	6 (17.6%)	3 (8.8%)	6 (17.6%)	34 (100.0%)
verde	8 (25.8%)	6 (19.4%)	8 (25.8%)	5 (16.1%)	4 (12.9%)	31 (100.0%)
Total	24 (24.0%)	21 (21.0%)	22 (22.0%)	12 (12.0%)	21 (21.0%)	100 (100.0%)

Risorse

- Exploratory Data Analysis Using R – Datazar Blog
- The Personality Project's Guide to R
- Exploratory Data Analysis with R
- visualising data - psyr.org

Esercizio

In questo esercizio caricheremo un file di dati, in formato tsv (valori separati da tab), puliremo il data.frame, e faremo alcune analisi descrittive sui dati.

```
partecipanti <- read.delim("https://s3.eu-central-1.amazonaws.com/bussolon/dati/par
```

Controllo

- la dimensione della tabella (righe e colonne)
- i nomi delle righe

Ci si aspetta che le righe corrispondano alle osservazioni (in questo caso, i partecipanti), mentre le colonne sono le variabili.

```
dim(partecipanti)
```

```
## [1] 768   7
```

```
names(partecipanti)
```

```
## [1] "Data"      "sex"       "age"       "scol"      "professione"
## [6] "risposte"    "corrette"
```

```
str(partecipanti)
```

```
## 'data.frame':    768 obs. of  7 variables:
## $ Data     : Factor w/ 767 levels "2003/06/25 18:45:44",..: 1 2 3 4 5 6 7 8 9
## $ sex      : Factor w/ 3 levels "femmina","maschio",..: 1 2 1 1 2 1 3 1 1 1 .
##  $ age        : int  25 40 45 31 50 31 NA 26 50 18 ...
##  $ scol       : int  13 13 13 13 18 13 NA 13 16 8 ...
## $ professione: Factor w/ 195 levels "?","/","///",..: 102 6 194 71 92 71 19
##  $ risposte   : int  48 49 44 47 49 49 47 49 49 49 ...
##  $ corrette   : int  30 39 37 24 34 30 29 35 38 30 ...
```

Se non altrimenti specificato, le funzioni `read.[table|csv|delim]` tendono a considerare le colonne stringa come fattori (`stringsAsFactors = TRUE`). Nel caso della variabile `sex` questo è corretto. Nel caso di `professione` è formalmente corretto, ma in pratica i dati sono talmente sporchi (perché raccolti via internet con una domanda a risposta aperta) che ci ritroviamo con 195 `livelli`. Nel caso della variabile `Data` la trasformazione in fattore è del tutto sbagliata.

```
summary(partecipanti)
```

```
##                Data        sex         age           scol
##  2003/08/29 15:50:24:  2  femmina:458  Min.   : 2.00  Min.   : 5.00
##  2003/06/25 18:45:44:  1  maschio:287  1st Qu.:24.00  1st Qu.:13.00
##  2003/06/26 10:34:00:  1  null   : 23  Median :29.00  Median :13.00
##  2003/06/26 11:10:21:  1               Mean   :32.33  Mean   :14.22
##  2003/06/26 12:03:05:  1               3rd Qu.:40.00  3rd Qu.:18.00
##  2003/06/26 12:48:03:  1               Max.   :99.00  Max.   :18.00
##  (Other)         :761               NA's   :37     NA's   :38
##       professione     risposte        corrette
##  XX          :242   Min.   : 0.00   Min.   : 0.00
##  studente    : 87   1st Qu.:48.00   1st Qu.:30.00
##  impiegata   : 50   Median :49.00   Median :34.00
##  studentessa: 40   Mean   :46.22   Mean   :32.43
##  impiegato   : 31   3rd Qu.:49.00   3rd Qu.:37.00
##  medico      : 20   Max.   :49.00   Max.   :49.00
##  (Other)     :298
```

I dati erano stati raccolti su internet, attraverso una servlet java. Nel caso della variabile `sex`, i dati mancanti erano stati contrassegnati con la stringa `null`. Pertanto, è necessario trasformare i `sex == null` in `NA`.

```
partecipanti$sex[partecipanti$sex=='null'] <- NA
levels(partecipanti$sex)
```

```
## [1] "femmina" "maschio" "null"
```

```
## per togliere il livello "null", che ora è vuoto
## [r - Drop factor levels in a subsetted data frame - Stack Overflow](htt

partecipanti$sex <- factor(partecipanti$sex)
levels(partecipanti$sex)
```

```
## [1] "femmina" "maschio"
```

```
# in alternativa https://stackoverflow.com/a/17218028/1042167
# y <- droplevels(y)
```

Nonostante la trasformazione dei `null` in `NA`, il fattore mantiene il livello `null`, anche se con 0 elementi. Per pulire i livelli, è necessario ri-applicare la funzione `factor()` alla colonna da pulire.

Scolarità

La scolarità dei partecipanti era stata codificata con un numero che corrispondeva agli anni:

- 5: elementari
- 8: scuole dell'obbligo
- 13: diploma
- 16: laurea triennale
- 18: laurea magistrale

Questa variabile, però, non è propriamente numerica, e può costituire un fattore su cui fare alcune analisi. Più in particolare può essere interessante capire se vi è un rapporto fra scolarità e numero di risposte corrette.

Trasformare in fattore

Per poter fare questo tipo di analisi, è opportuno trasformare questa variabile da numerica (int) ad un fattore. La funzione per operare questa trasformazione è `as.factor()`.

```
partecipanti$scol <- as.factor(partecipanti$scol)
levels(partecipanti$scol)
```

```
## [1] "5"  "8"  "13" "16" "18"
```

```
levels(partecipanti$scol) <- c("elementari", "medie","diploma","triennale","magistr
```

```
as.integer(partecipanti$scol)[1:20] # i primi 20 elementi
```

```
##  [1]  3  3  3  3  5  3 NA  3  4  2  3  5  3  3  5  5  5  5  3  3
```

Le date

Attraverso la funzione `strptime(stringa, formato)` possiamo fare il parsing della colonna data, rendendo esplicito il formato da utilizzare, ed otterremo un vettore di date.

```
partecipanti$Data <- strptime(partecipanti$Data, format = "%Y/%m/%d %H:%M:%S")
summary(partecipanti$Data)
```

```
##                Min.             1st Qu.              Median
## "2003-06-25 18:45:44" "2003-07-22 19:00:15" "2003-08-
21 12:19:56"
##                Mean             3rd Qu.                Max.
## "2003-08-21 00:29:27" "2003-09-18 02:39:42" "2003-10-
21 00:19:14"
```

```
partecipanti$professione[partecipanti$professione=="XX"] <- NA
str(partecipanti$professione)
```

```
##  Factor w/ 195 levels "?","/","///",..: 102 6 NA 71 92 71 NA 160 117 170 ...
```

```
partecipanti$professione <- as.factor(tolower(partecipanti$professione))
```

Nella colonna `professione`, i dati mancanti erano contrassegnati con i caratteri XX. Attraverso il filtro, abbiamo trasformato XX in NA.

Essendo questo un campo a testo libero, alcune persone scrivevano tutto minuscolo, altre maiuscolo. Per semplificare le cose, abbiao trasformato i valori in minuscolo, con la funzione `tolower()`.

Boxplot

Una modalità efficace per valutare visivamente se ci sono delle differenze fra diversi gruppi è utilizzando i grafici Box plot.

Ad esempio, potremmo chiederci se vi è una relazione fra scolarità e numero di risposte corrette.

```
boxplot(partecipanti$corrette ~ partecipanti$scol)
```

Il grafico boxplot permette di visualizzare la mediana e la distanza interquartilica (ovvero fra il primo ed il terzo quartile). Permette inoltre di stimare la varianza (attraverso i *baffi*) e gli eventuali *outlier*, ovvero delle osservazioni che si discostano in maniera evidente dalla distribuzione. Dal grafico si può osservare che vi sono numerosi dati anomali. Nel test i partecipanti dovevano fare un numero di scelte vero o falso su 50 item. Chi rispondeva a caso aveva il 50% di probabilità di rispondere correttamente. Tutti i punteggi inferiori a 20 risultano dunque *sospetti*. L'ipotesi più plausibile è che quei punteggi rappresentino dei soggetti che avevano fatto molte omissioni (ovvero risposto soltanto ad alcuni dei 50 item).

Per valutare questa ipotesi, possiamo verificare la colonna *risposte*.

```
hist(partecipanti$risposte)
```

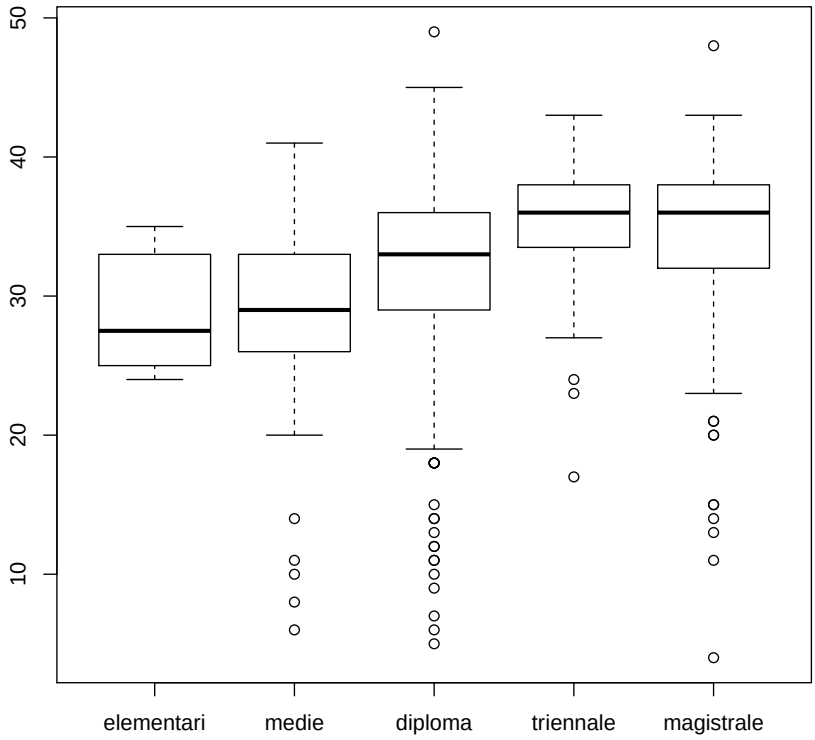
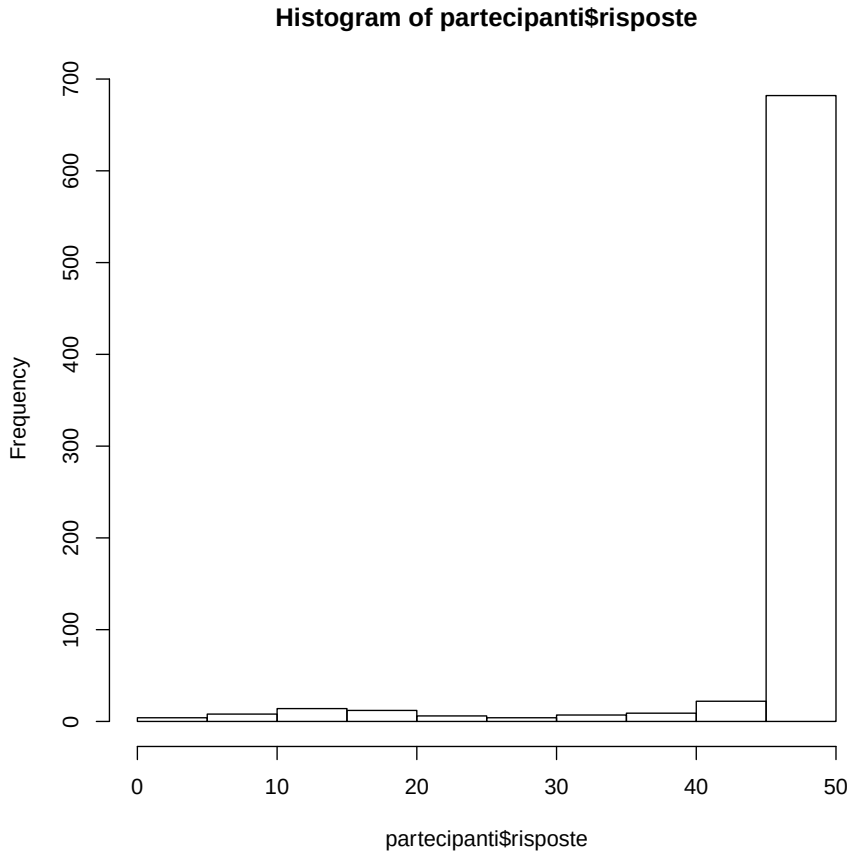


Figura 7.7: plot of chunk desc_boxplot_partecipanti



L'istogramma della variabile `risposte` lascia intendere che sebbene la maggior parte dei partecipanti abbia risposto ad almeno 40 delle 50 domande, alcuni hanno fatto un numero di omissioni più alto. Per avere una misura dei partecipanti che hanno risposto a meno di 40 domande, possiamo usare un filtro (`partecipanti$risposte>40`) e la funzione `table()`

```
table(partecipanti$risposte>=40)
```

```
##
## FALSE  TRUE
##    61   707
```

Su 768 partecipanti, 64 hanno risposto a meno di 40 domande.

```
plot(partecipanti$risposte, partecipanti$corrette, col = as.factor(parteci
```

Il grafico conferma la nostra ipotesi: i punteggi più bassi corrispondono a quei partecipanti che hanno risposto a meno domande.

Attraverso il parametro `col = as.factor(partecipanti$risposte<40)` abbiamo colorato i punti del plot in base alla soglia delle 40 risposte.

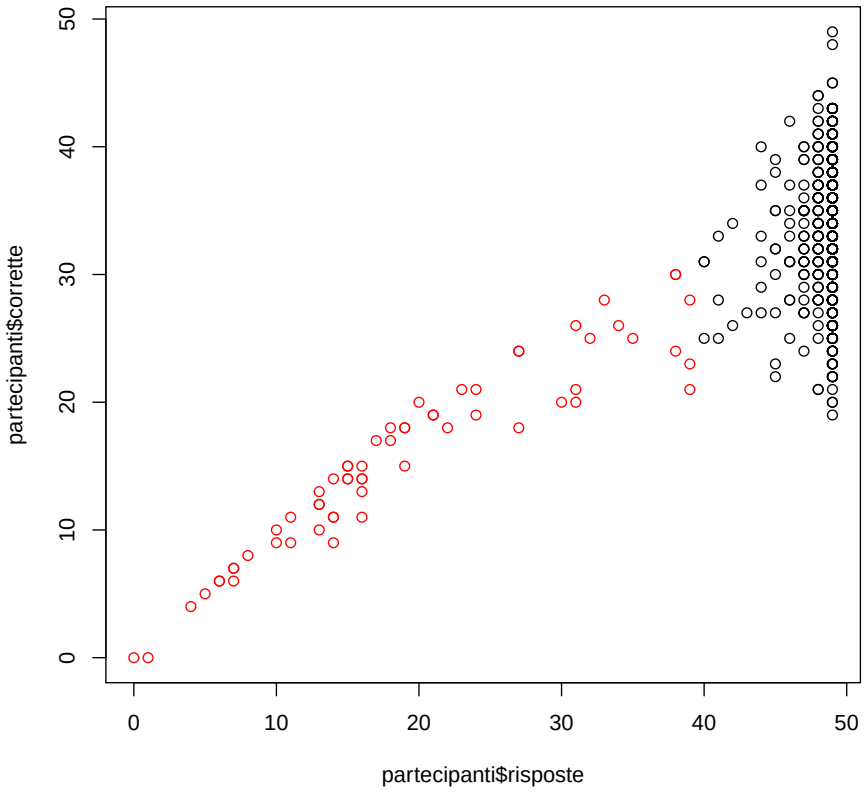


Figura 7.8: plot of chunk desc_plot_risposte_corrette_1

Filtrare i partecipanti

A questo punto, possiamo decidere di considerare validi solo i partecipanti che hanno risposto ad almeno 40 domande.

```
partecipanti_validi <- partecipanti[partecipanti$risposte>=40,]
dim(partecipanti_validi)
```

```
## [1] 707   7
```

Il nuovo data frame ha ancora 7 colonne, e 707 righe.

Per verificare la distribuzione di risposte date e risposte corrette, possiamo fare il plot sul nuovo data frame.

```
plot(partecipanti_validi$risposte, partecipanti_validi$corrette)
```

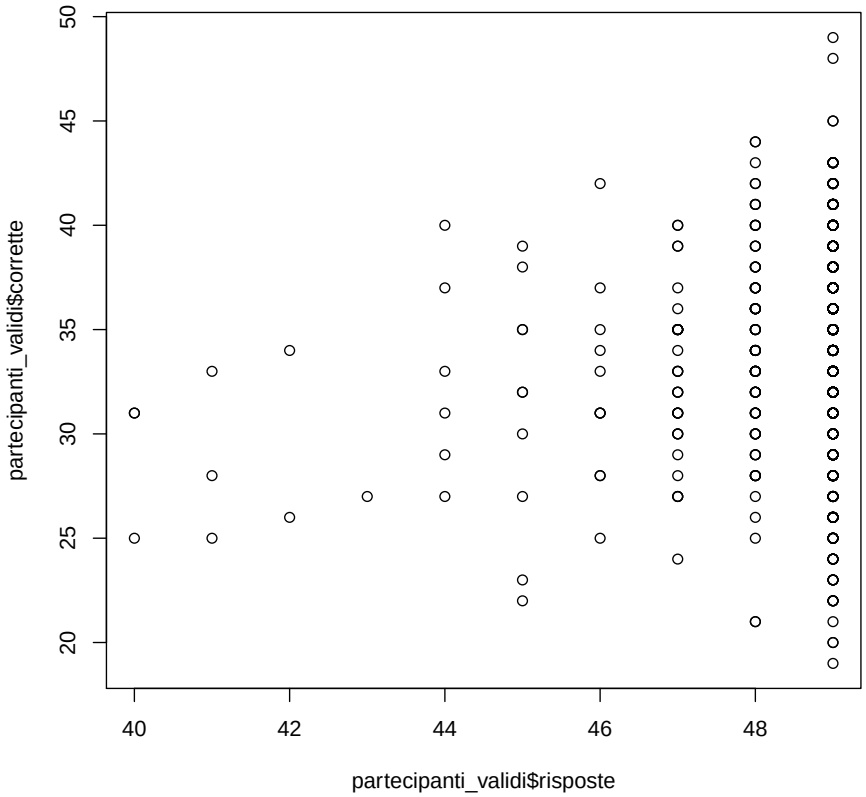


Figura 7.9: plot of chunk desc_plot_part_validi_1

Come prevedibile, nel nuovo data frame il numero minimo di risposte corrette è 19.

Creiamo il boxplot corrette ~ scol sul nuovo data frame.

```
boxplot(partecipanti_validi$corrette ~ partecipanti_validi$scol)
```

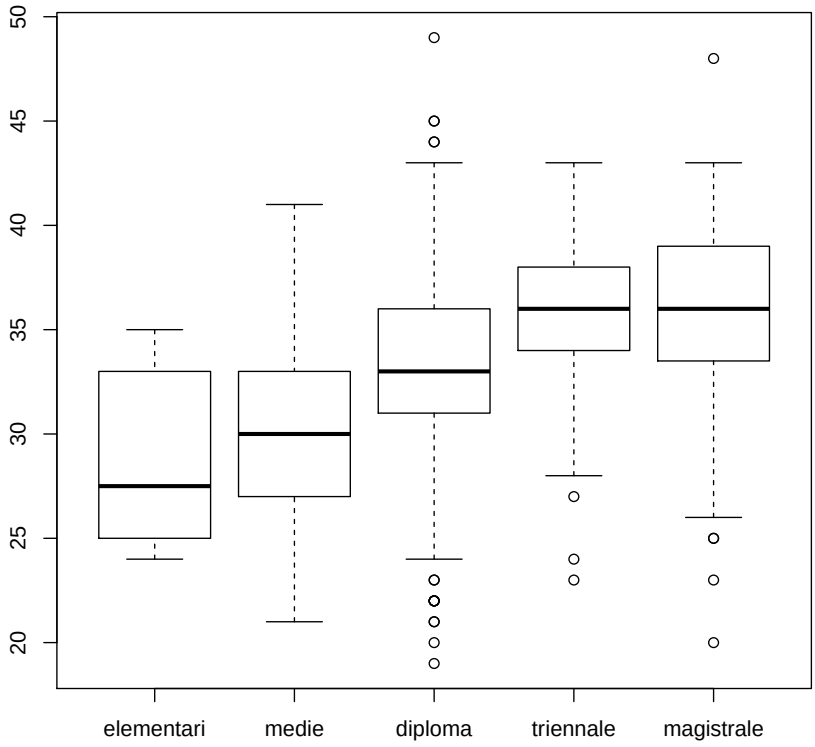


Figura 7.10: plot of chunk desc_part_validi_boxplot_1

Attach

Per evitare di ripetere ogni volta il nome del data frame, possiamo utilizzare la funzione `attach ()`, che ci permette di richiamare direttamente i nomi delle variabili del data frame.

```
## summary(scol) ---> Error in summary(scol) : oggetto "scol" non trovato
attach(partecipanti_validi)
summary(scol)
```

```
## elementari    medie   diploma  triennale magistrale     NA's
##         10       65      334         59       212       27
```

Analisi delle variabili nominali

Possiamo calcolare la frequenza attraverso la funzione table().

```
(freq_sex <- table (sex))
```

```
## sex
## femmina maschio
##     431     265
```

```
freq_sex / sum(freq_sex) # frequenza
```

```
## sex
##   femmina   maschio
## 0.6192529 0.3807471
```

```
prop.table(freq_sex)     # = freq_sex / sum(freq_sex)
```

```
## sex
##   femmina   maschio
## 0.6192529 0.3807471
```

Grafici su variabili nominali

La funzione barplot mi permette di fare un grafico a barre.

```
barplot (freq_sex)
```

pie è una funzione che permette di generare dei grafici a torta. Qualsiasi guida di data visualization considera i grafici a torta come una pessima modalità di visualizzazione. Dunque il consiglio è: usateli con molta, molta parsimonia. Nel caso di un fattore con due livelli, come in questo caso, il grafico a torta può dare un'idea visiva della frequenza nei due generi. È comunque sconsigliabile utilizzarlo in una pubblicazione accademica.

```
pie(freq_sex)
```

Calcolo della moda

```
t_sex<-tabulate(sex)
(mode_sex<-which(t_sex == max(t_sex)))
```

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

```
sex[mode_sex]
```

```
## [1] femmina
## Levels: femmina maschio
```

```
t_sex[mode_sex]
```

```
## [1] 431
```

:todo: tabulate

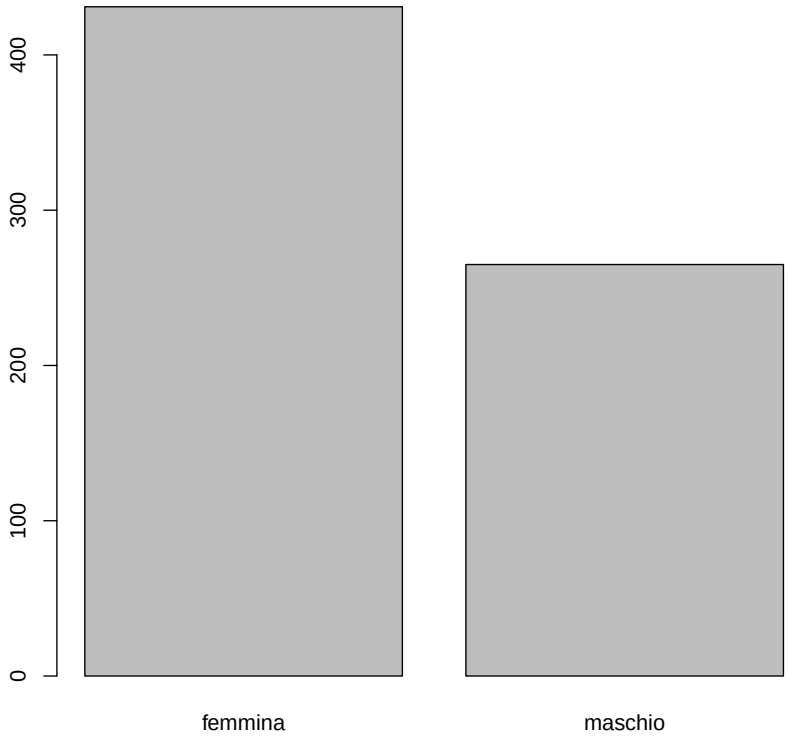


Figura 7.11: plot of chunk desc_barplot_sex

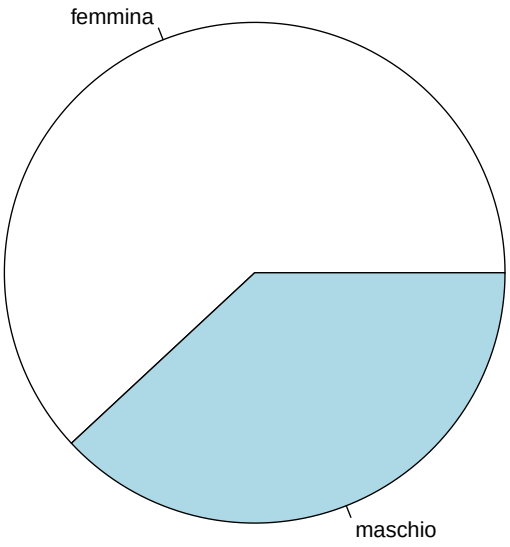


Figura 7.12: plot of chunk desc_pie_sex

Scolarità

```
freq_scol <- table (scol)
freq_scol
```

```
## scol
## elementari      medie    diploma  triennale magistrale
##         10         65        334         59        212
```

```
prop.table(freq_scol)
```

```
## scol
## elementari      medie    diploma  triennale magistrale
## 0.01470588 0.09558824 0.49117647 0.08676471 0.31176471
```

```
barplot (freq_scol)
```

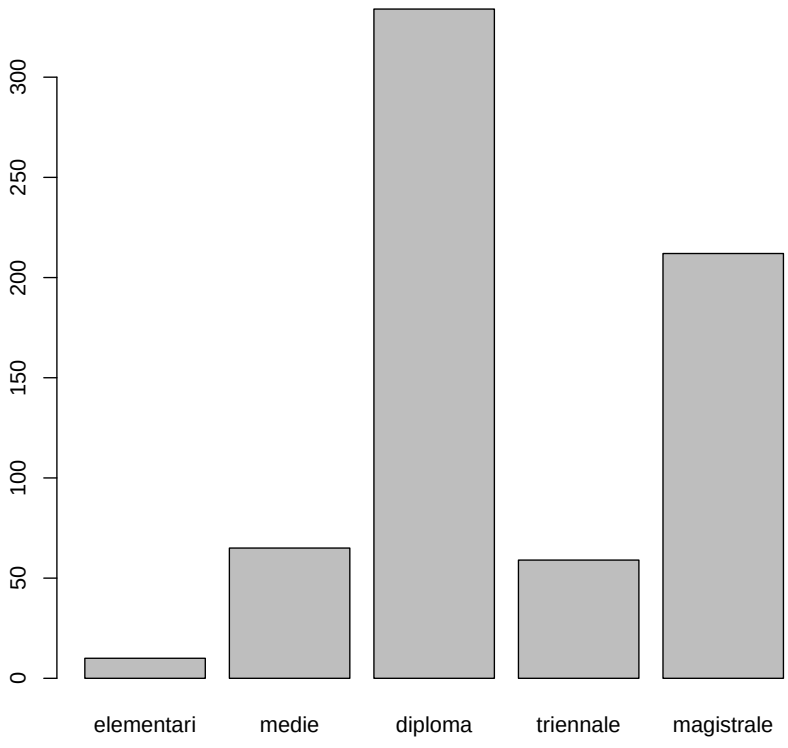


Figura 7.13: plot of chunk desc_barplot_scol

Calcolo della moda

```
t_scol<-tabulate(scol)
mode_scol<-which(t_scol == max(t_scol))
mode_scol
```

```
## [1] 3
```

```
scol[mode_scol]
```

```
## [1] diploma
## Levels: elementari medie diploma triennale magistrale
```

```
t_scol[mode_scol]
```

```
## [1] 334
```

+++todo+++ distribuzione delle risposte corrette

Distribuzione delle risposte corrette

attraverso `hist()` possiamo visualizzare la distribuzione di variabili a intervalli o a rapporti. Usiamola per visualizzare la distribuzione di risposte corrette.

```
hist(partecipanti_validi$corrette)
```

Una alternativa, per le variabili numeriche, è utilizzare la funzione `plot(density())`.

```
plot(density(partecipanti_validi$corrette))
```

`fivenum()` è la versione sintetica di `summary()`.

```
fivenum(partecipanti_validi$corrette)
```

```
## [1] 19 31 34 37 49
```

```
summary(partecipanti_validi$corrette)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   31.00   34.00   33.88   37.00   49.00
```

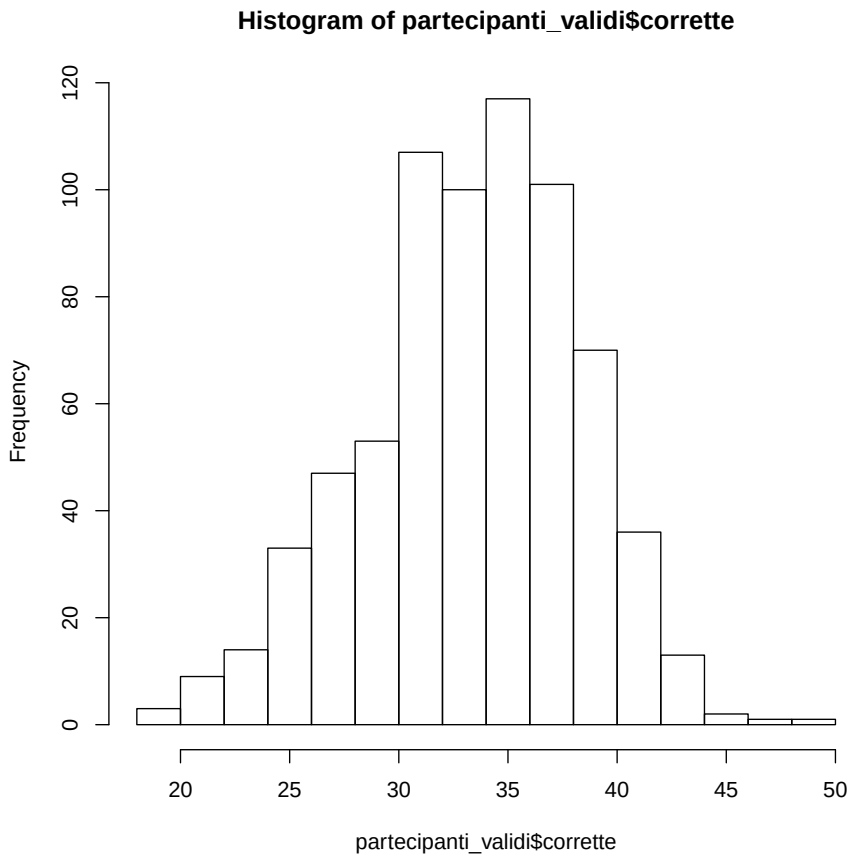


Figura 7.14: plot of chunk descr_hist_corrette

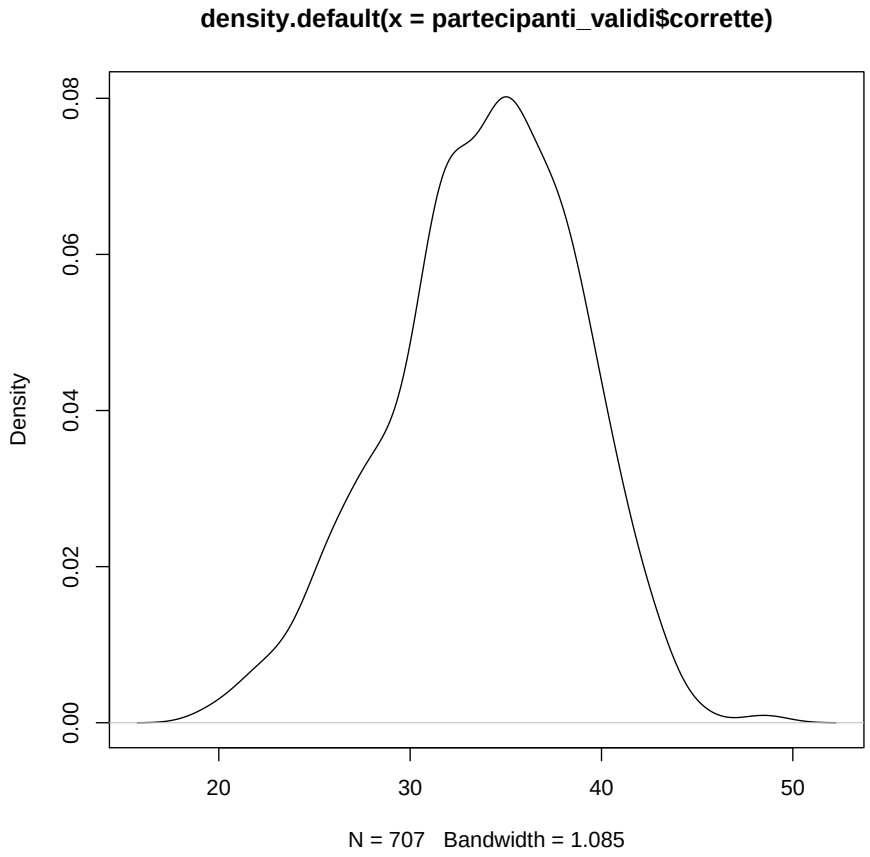


Figura 7.15: plot of chunk desc_plot_density

Capitolo 8

rMarkdown

RMarkdown

Markdown

Markdown è un linguaggio di markup con una sintassi del testo semplice progettata in modo che possa essere convertita in HTML e in molti altri formati usando degli appropriati strumenti (ad esempio pandoc).

Esempio di codice markdown

```
# Titolo di primo livello

Testo normale con un [link](http://example.com)

## Titolo di secondo livello

Testo con *corsivo*, **grassetto**, `monospace`

> questa è una citazione

![immagine](https://via.placeholder.com/150)

* una lista
* di punti
  * indentata
* non ordinata

1. una lista
2. numerata
   2.1. indentata
   2.2. con una propria numerazione
3. di punti

Una nota [^nota1] a pie di pagina
[^nota1]: il testo della nota
```

Omettendo i titoli di primo e secondo livello, il codice precedente produce la seguente formattazione:

Testo normale con un link

Testo con *corsivo*, **grassetto**, `monospace`

> questa è una citazione

Figura 8.1: immagine

- una lista
- di punti
  - indentata

- non ordinata

1. una lista
2. numerata
    1. indentata
    2. con una propria numerazione
3. di punti

Una nota [1] a pie di pagina

Per una guida dettagliata della versione di markdown utilizzata da Pandoc via RStudio, si veda Pandoc Markdown

Una citazione bibliografica: @KeselmanWilcoxOthmanFradette:2002

Rmarkdown

RMarkdown si basa sull'idea di literate programming: integrare in uno stesso documento codice eseguibile e testo.

RMarkdown utilizza `knitr` per processare la parte di codice, creare l'output (compresi i grafici) ed includerli in un file markdown dinamicamente generato. Questo file viene poi trasformato in un documento (ad esempio html, pdf o word) utilizzando il programma `Pandoc`. Pandoc è uno strumento estremamente potente, capace di trasformare documenti di numerosi formati.

Il documento che stai leggendo è stato creato in RMarkdown.

Per usare RMarkdown è necessario installare la corrispondente libreria.

```
install.packages("rmarkdown")
install.packages("knitr")
```

Anatomia di RMarkdown

La prima parte di un documento RMarkdown (opzionale, ma estremamente utile) è una sezione YAML che descrive i *metadati* del documento: titolo, autore, data di creazione, abstract.

Il resto del documento integra delle parti testuali, scritte in Markdown, con dei `chunk` di codice `R`. In realtà RMarkdown supporta altri linguaggi, oltre ad R (ad esempio python).

```
title: "r markdown"
author: "Stefano Bussolon"
date: "febbraio 2019"
output: html_document
```

[1] il testo della nota

I `chunk`

Un chunk è un pezzo di codice che inizia con “‘{r} e termina con “‘

Ad esempio, il codice

````
```r
vettore1 <- c(2,4,6,8,10,12)
# seleziona il quinto elemento
vettore1[5]
# moltiplica il vettore
vettore1*3
```
````

genera il seguente risultato:

```r
vettore1 <- c(2,4,6,8,10,12)
# seleziona il quinto elemento
vettore1[5]
```

```
## [1] 10
```

```r
# moltiplica il vettore
vettore1*3
```

```
## [1]  6 12 18 24 30 36
```

Posso mostrare nel testo il valore della variabile con la sintassi `_r variabile`.

Ad esempio: il valore di `vettore1[3]` è 6.

I file Rmarkdown di questo corso

Il modo forse più efficace per farsi un’idea di come funziona Rmarkdown è quello di scaricare, far girare e modificare i file usati per generare le slide di questo corso, che sono pubblicate in Materiale

Risorse

- Markdown - Wikipedia
- Markdown Cheatsheet · adam-p/markdown-here Wiki

Capitolo 9

Visualizzare i dati - ggplot2

Visualizzazione dei dati

I grafici costituiscono una forma di rappresentazione visiva, su di un'area bidimensionale, di un insieme di informazioni. Le dimensioni spaziali e visive vengono utilizzate per rappresentare dati quantitativi, ordinali o categoriali.

Uno dei vantaggi della rappresentazione grafica è che sfrutta alcune caratteristiche del sistema visivo:

- la capacità di elaborare preattentivamente, in parallelo, un'ampia quantità di dati;
- la capacità di identificare pattern, distribuzioni, relazioni fra i dati;
- la capacità di percepire l'allineamento, orizzontale e verticale, degli elementi grafici;
- la capacità di stimare, anche se approssimativamente, differenze di lunghezza, di luminosità, di colore e di forma degli elementi

Più in generale, le rappresentazioni grafiche sfruttano alcuni dei princìpi della gestalt:

- prossimità
- continuità
- allineamento
- connessione
- chiusura

Un buon grafico riesce a comunicare una appropriata quantità di informazioni in maniera non distorta e facile da comprendere; permette di far emergere pattern, distribuzioni di variabili, relazioni fra variabili, presenza di outlier, relazioni fra osservazioni o fra statistiche, e di stimare, seppure approssimativamente, il valore delle singole osservazioni.

Un buon grafico è intuitivo, massimizza il rapporto fra informazione e *rumore*, senza indurre un sovraccarico cognitivo; gli aspetti estetici contribuiscono a rappresentare l'informazione

Gli elementi grafici

Un grafico è un oggetto grafico complesso. Gli oggetti grafici complessi sono composti da

- uno spazio grafico (bidimensionale)
- degli oggetti grafici (complessi o semplici)
- delle strutture di riferimento (titoli, legende, assi, etichette, annotazioni), che permettono l'interpretazione e facilitano la lettura dei dati

La definizione è ricorsiva: un oggetto grafico complesso può contenere altri oggetti grafici complessi.

Gli oggetti grafici rappresentano:

- osservazioni
- statistiche
- relazioni (fra osservazioni o fra statistiche)

Gli oggetti grafici semplici sono formati da figure geometriche: punti (0d), linee o curve (1d), rettangoli o aree (2d), da icone o da elementi testuali

Rappresentare le relazioni

Le relazioni (fra osservazioni o statistiche) sono rappresentate attraverso differenti modalità:

- clustering spaziale
- inclusione in un contenitore (categoriale)
- utilizzo di separatori (categoriale)
- allineamento (relazione ordinale)
- collegamento: generalmente le osservazioni sono punti (nodi), i collegamenti linee o curve (archi); possono essere rappresentati varie tipologie di grafi: catene, loop, grafi complessi, alberi

Codifica delle variabili

Gli oggetti grafici codificano l'informazione in base ad alcune dimensioni. Le principali sono

- la posizione sugli assi x ed y
- la forma
- il colore
- la luminosità
- la dimensione

Altre possibili dimensioni sono il tipo e lo spessore dei bordi o delle linee, il pattern in caso di aree.

Le coordinate x e y sono adatte a rappresentare variabili continue, ordinali e categoriali.
La forma (dei punti), il tipo di linea, alcuni pattern di riempimento delle aree, e alcune palette di colore sono adatti a rappresentare un numero limitato di categorie.
Alcune palette di colore, la luminosità degli oggetti e la densità dei pattern sono adatti a rappresentare le variabili ordinali, mentre sono meno adatti a rappresentare le variabili ad intervalli, in quanto la capacità del nostro sistema visivo di stimare gli intervalli è meno precisa.

Il sistema visivo riesce a stimare abbastanza bene la dimensione degli oggetti (rappresentazione grafica dei punti, la lunghezza delle linee, la dimensione delle aree) se si sviluppa in una sola dimensione, mentre ha difficoltà a stimare la dimensione in base all'area.
Anche la stima angolare è più approssimativa (e dunque è più difficile stimare le proporzioni nei grafici con assi radiali: grafici a torta o a ciambella)

È pertanto potenzialmente possibile rappresentare, in un grafico, fino a 5 variabili di ogni osservazione, attraverso le coordinate x e y (a intervalli, ordinali o categoriali), la forma (categorie), il colore (ordinale o categoriale) e la dimensione (ordinale o a rapporti).

In pratica, però, si rischia l'overload cognitivo. È pertanto opportuno limitarsi a rappresentare fino a 3, al massimo 4 variabili.

Fundamentals of Data Visualization

ggplot2

ggplot2 è un pacchetto per creare grafici in R. ggplot2 è molto popolare, in quanto

- crea grafici esteticamente piacevoli
- è molto potente
- utilizza in maniera esplicita l'approccio di *Grammar of Graphics*

La logica

Con ggplot2 la creazione di un grafico avviene attraverso una serie di passi, che possono essere concatenati:

- creare l'oggetto ggplot ed associare il set di dati (generalmente un data.frame) da rappresentare
- calcolare le eventuali statistiche
- identificare la variabile (analisi univariata) o le variabili (rappresentazione bi- o multivariata) da mappare sugli assi x e y o sulle dimensioni grafiche del colore, della dimensione, della forma
- decidere la *geometria*, ovvero il tipo di elementi visuali (punti, linee, aree o oggetti complessi) in base alla tipologia di dati e delle eventuali statistiche da rappresentare
- definire le scale e le eventuali trasformazioni
- creare uno o più livelli grafici (layer) dove collocare
  - gli elementi grafici (semplici e complessi)
  - le strutture di supporto alla lettura ed interpretazione (titoli, assi, scale, griglie)

La sovraimposizione di livelli diversi permette di rappresentare, sullo stesso spazio bidimensionale, osservazioni, statistiche e strutture di supporto

La sintassi

La creazione di un grafico ggplot2 utilizza una sintassi che può essere sintetizzata nel seguente template (cita fonte)

```
grafico <- ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,
     position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION>
```

```
print (grafico)
```

Risorse

- The Grammar of Graphics
- A Comprehensive Guide to the Grammar of Graphics for Effective Visualization of Multi-dimensional…
- The Art of Effective Visualization of Multi-dimensional Data
- Grammar of Graphics in R
- 3 Data visualisation | R for Data Science
- Quick-R: ggplot2 Graphs
- Data visualization with ggplot2
- Before Tufte, there was Bertin – Karl Sluis – Medium
- Beyond Tufte – Karl Sluis – Medium
- Top 50 ggplot2 Visualizations - The Master List (With Full R Code)

Esempi

Per mostrare ggplot2 in azione, carichiamo il dataframe `parole_nonparole_pulito.rds` e carichiamo la libreria `ggplot2` (eventualmente la installiamo se non presente nel nostro sistema).

```
# install.packages("ggplot2")
library(ggplot2)
# partecipanti <- readRDS("https://s3.eu-central-1.amazonaws.com/bussolon/dati/parc
partecipanti <- readRDS("../../dati/parole_nonparole_pulito.rds")
```

Barplot

Creaiamo un grafico a barre con la distribuzione della scolarità. Mappiamo anche la dimensione del sesso.

```
g <- ggplot(partecipanti, aes(scol))
g + geom_bar(aes(fill=sex), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6)) +
  labs(title="Scolarità",
       subtitle="Scolarità, per sesso",
       caption="Parole non parole - barplot")
```

Un secondo esempio di barplot per visualizzare la distribuzione del sesso.

```
ggBar2 <- ggplot(partecipanti, aes(x="", fill=sex)) +
    geom_bar(width = 0.5) +
    #theme(axis.text.x = element_text(angle=65, vjust=0.6)) +
    labs(title="Genere",
       #subtitle="Genere",
             x= "",
```

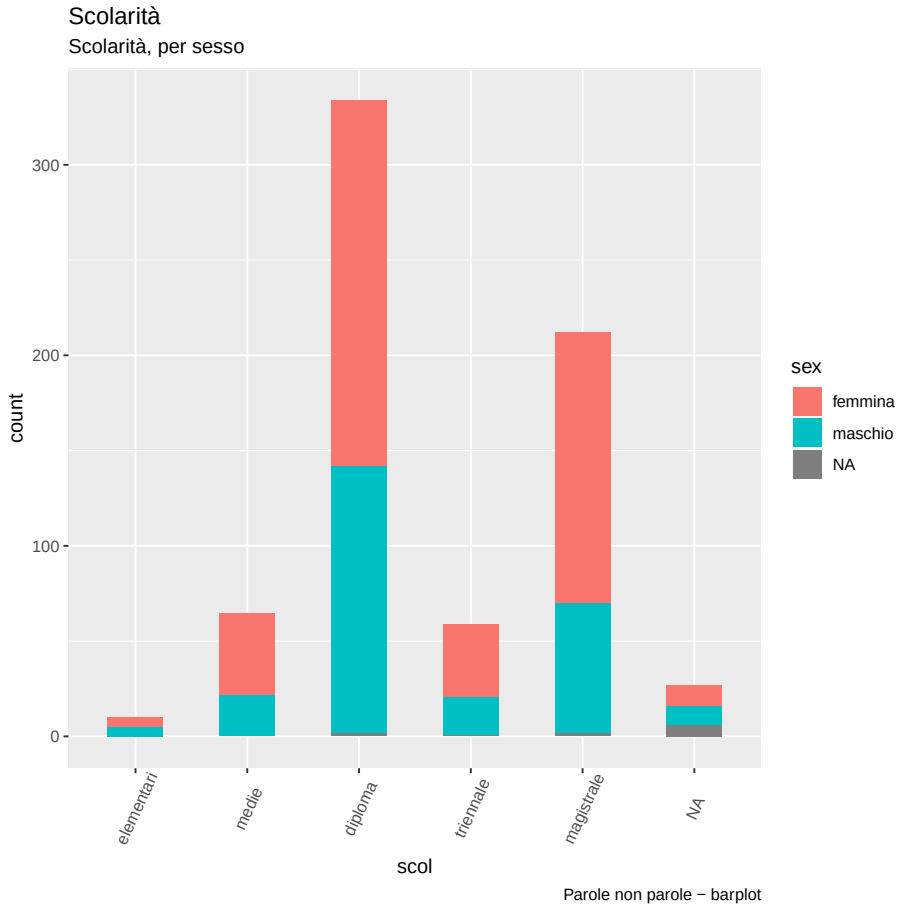


Figura 9.1: Grafico a barre

```
          y= "Frequenza",
       caption="Parole non parole - barplot 2")
plot(ggBar2)
```

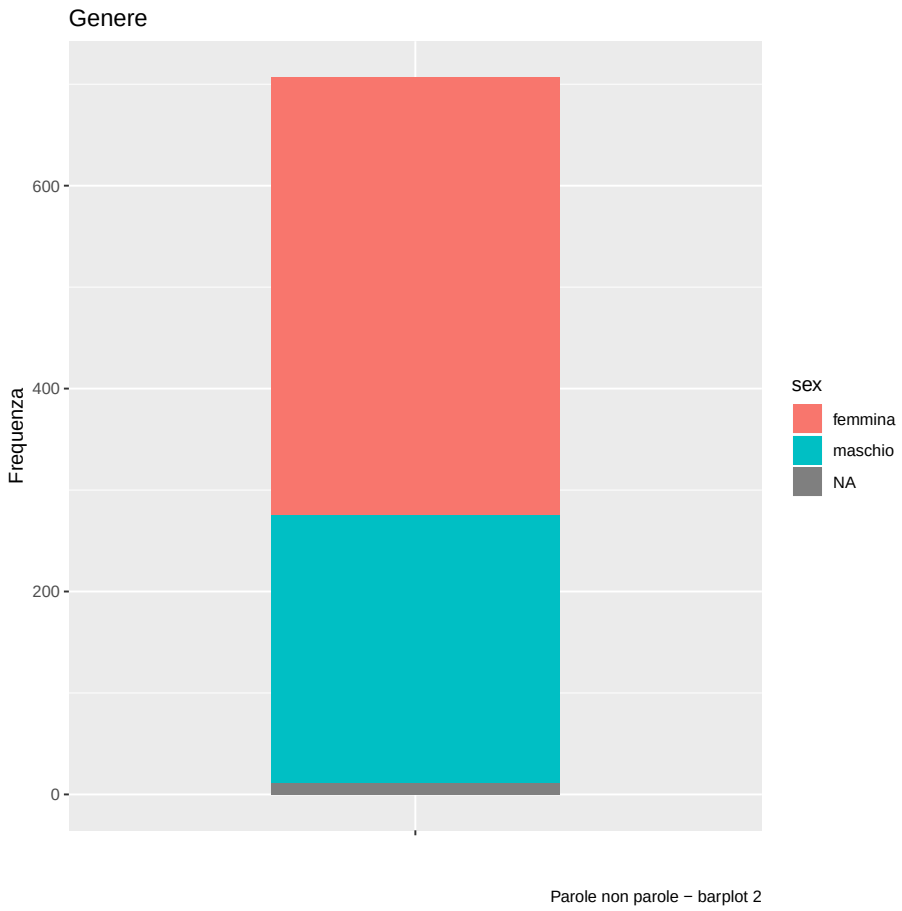


Figura 9.2: Secondo barplot

Grafico a torta

In ggplot2 un grafico a torta è un grafico a barre con coordinate polari.

```
ggPie <- ggBar2 +
    coord_polar("y", start=0)

plot(ggPie)
```

Istogramma

L'istogramma della distribuzione delle risposte corrette

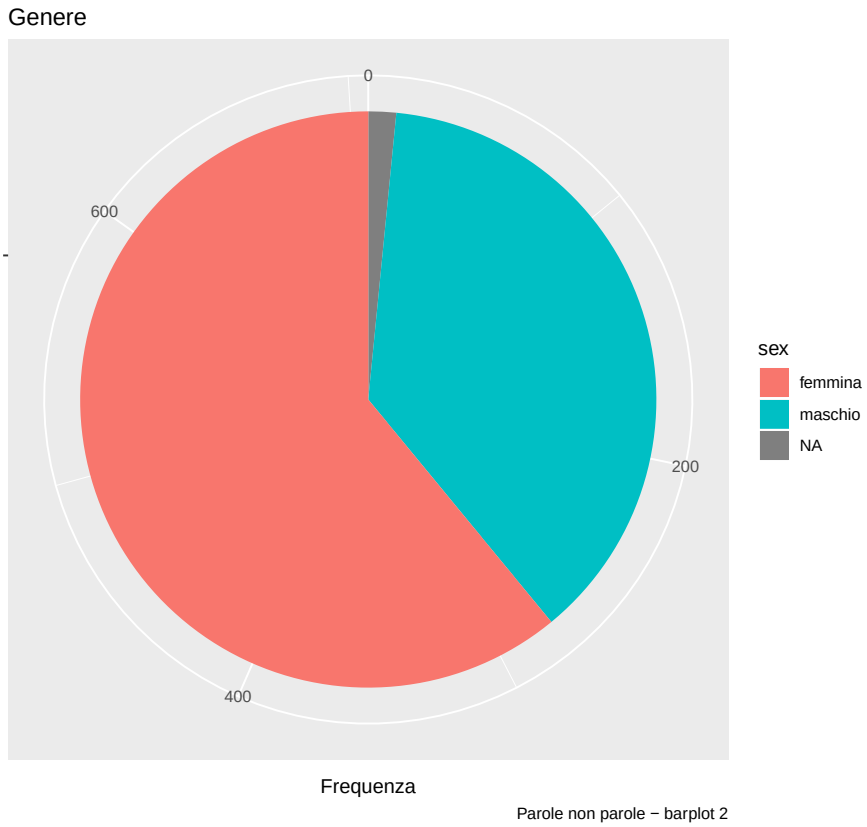


Figura 9.3: Grafico a torta

```
ggHist <- ggplot(data=partecipanti, aes(corrette)) +
    geom_histogram() +
  labs(title="Corrette",
         x="Risposte corrette",
         y="",
       subtitle="Numero di risposte corrette",
       caption="Parole non parole - barplot")

plot(ggHist)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

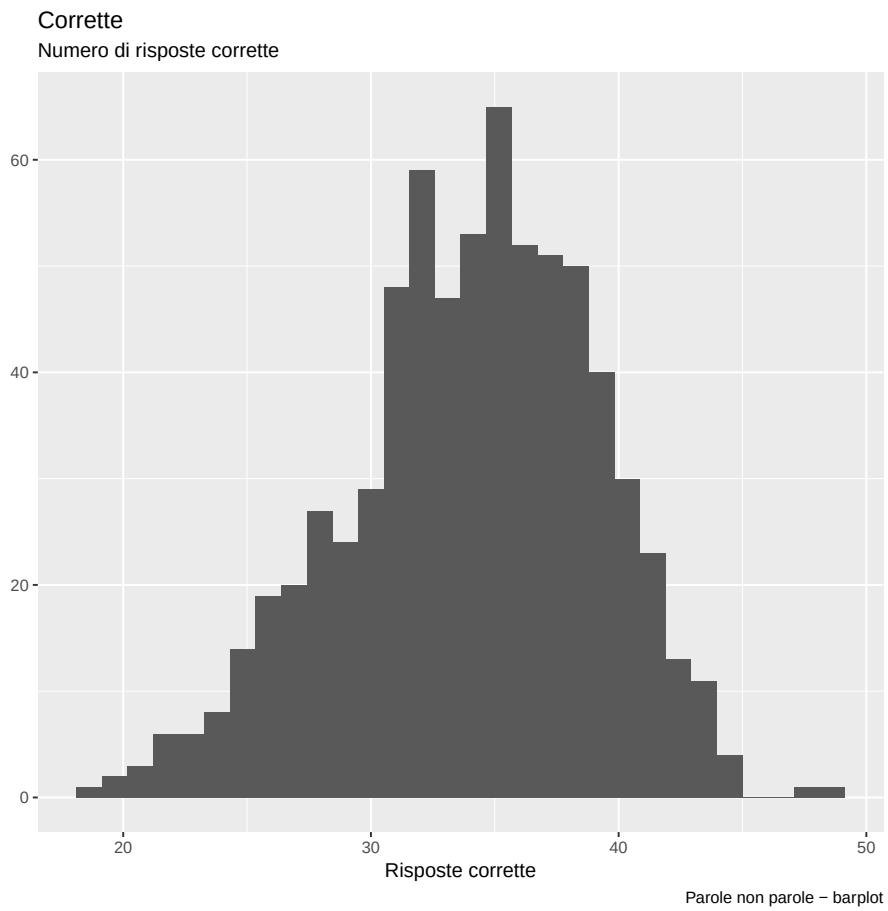


Figura 9.4: Istogramma

Histograms and frequency polygons — geom_freqpoly • ggplot2

Density plot

```
ggDensity <- ggplot(data=partecipanti, aes(corrette)) +
    geom_density() +
```

```
  labs(title="Corrette",
         x="Risposte corrette",
         y="",
       subtitle="Numero di risposte corrette",
       caption="Parole non parole - Density")

plot(ggDensity)
```

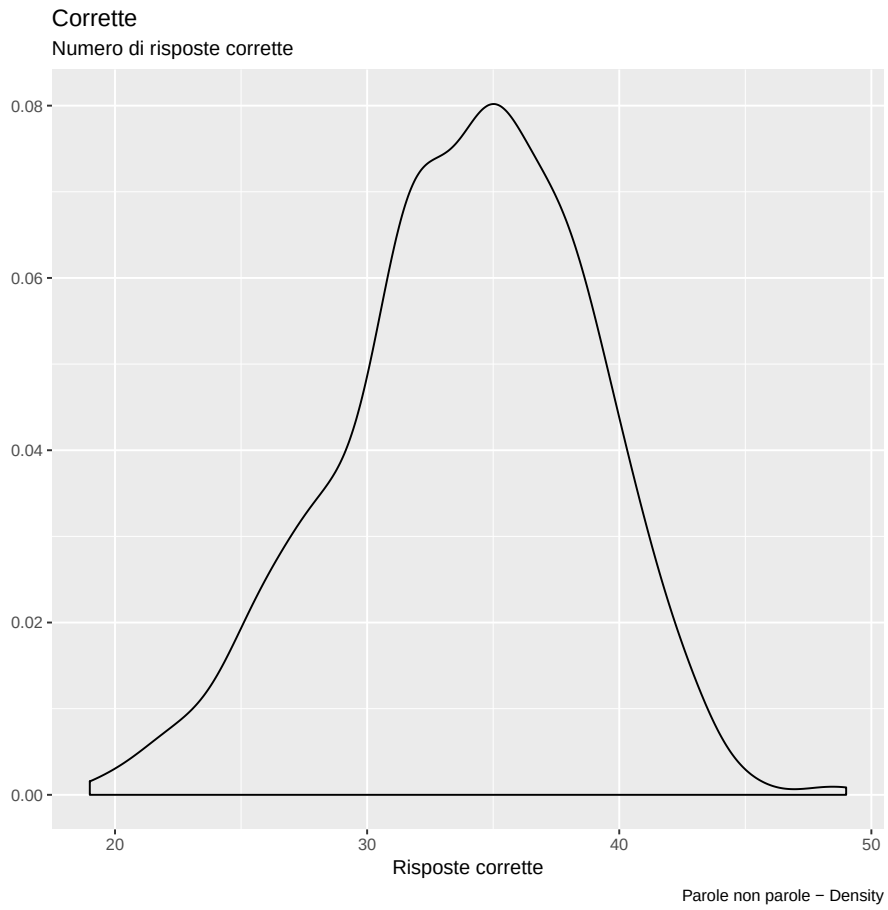


Figura 9.5: Density graph

```
# geom_density(aes(fill=factor(cyl)), alpha=0.8) +
```

Di nuovo il density plot. In questo caso, però, definiamo il colore dell'area (fill) in base alla scolarità. In questo modo otterremo 6 aree sovrapposte.

Va notato che l'altezza delle aree è normalizzata, e non proporzionale alla frequenza delle varie categorie.

```
ggDensity2 <- ggplot(data=partecipanti, aes(corrette)) +
    geom_density(aes(fill=factor(scol)), alpha=0.5) +
  labs(title="Corrette",
         x="Risposte corrette",
```

```
        y="",
      subtitle="Numero di risposte corrette",
      caption="Parole non parole - Density")

plot(ggDensity2)
```

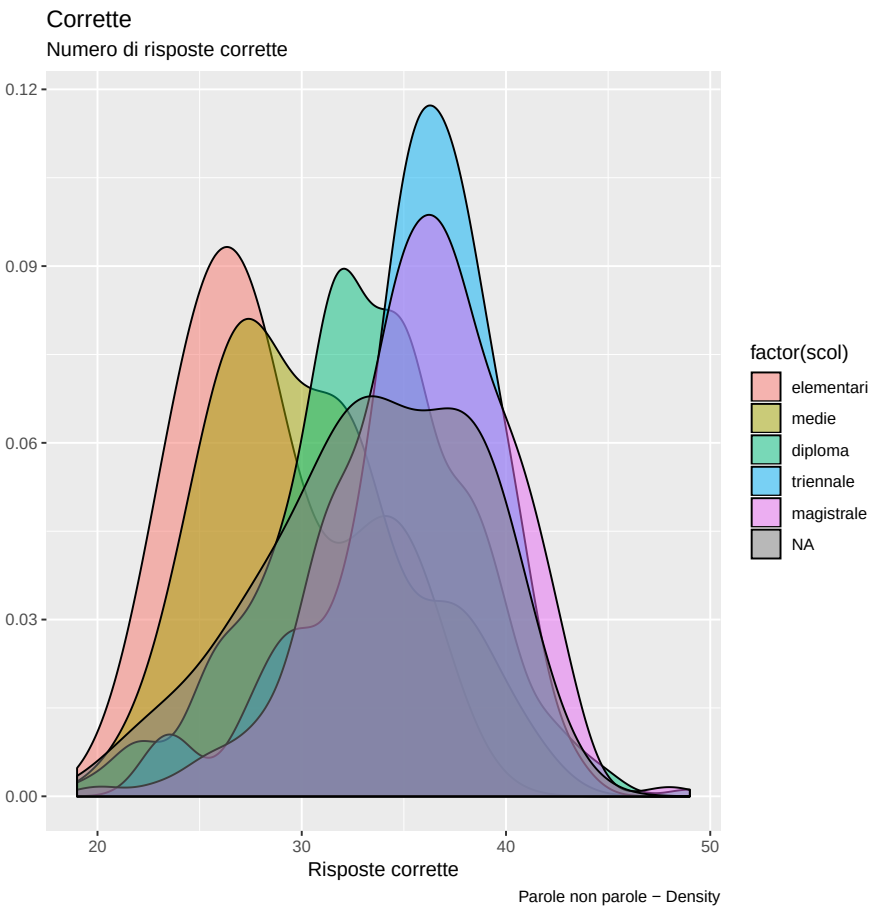


Figura 9.6: Density graph - per scolarità

```
# geom_density(aes(fill=factor(cyl)), alpha=0.8) +
```

Boxplot

Il boxplot della distribuzione delle risposte corrette, distinte per scolarità

```
ggBoxplot <- ggplot(data=partecipanti, aes(scol,corrette)) +
    geom_boxplot(varwidth=T) + # , fill="plum"
  labs(title="Corrette x scolarità",
       x="Risposte corrette",
       y="",
      subtitle="Numero di risposte corrette",
```

```
       caption="Parole non parole - Boxplot")

plot(ggBoxplot)
```

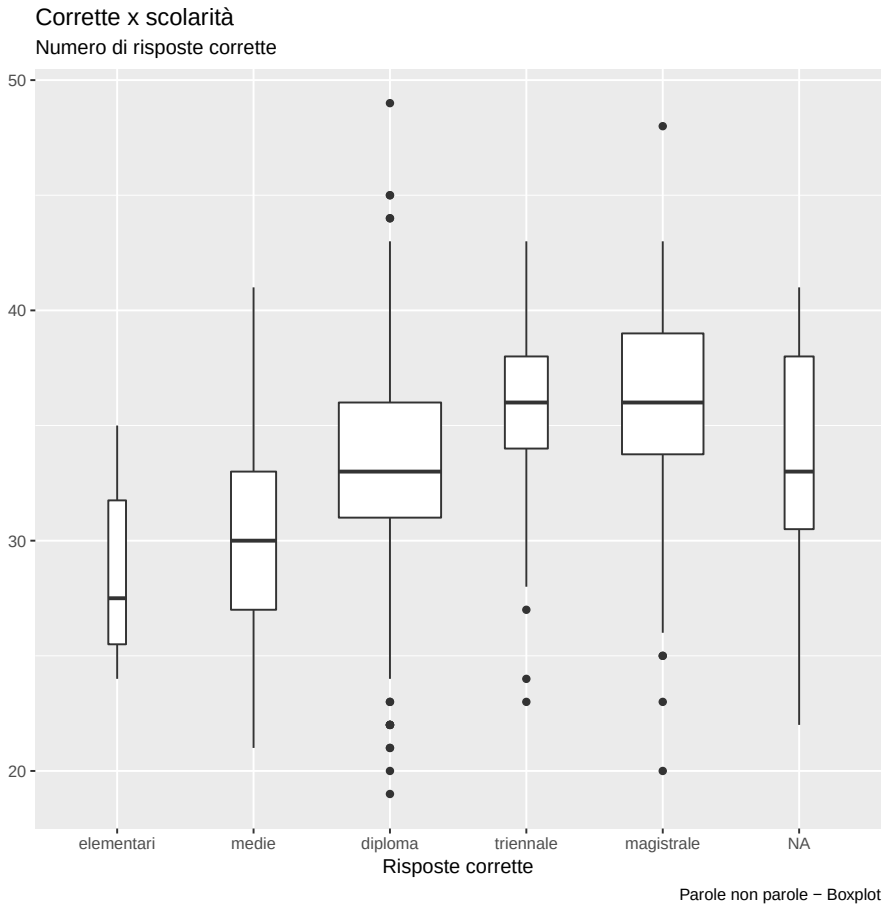


Figura 9.7: Boxplot corrette x scolarità

In questo grafico al boxplot sono sovrapposti (in un layer superiore) i punti corrispondenti alle osservazioni. Viene usata la geometria `jitter`, che

```
ggBoxplot <- ggplot(data=partecipanti, aes(scol,corrette)) +
    geom_boxplot(varwidth=T) + # , fill="plum"
    geom_jitter(width = .2, size=1, color="lightslateblue") +
  labs(title="Corrette x scolarità",
         x="Risposte corrette",
         y="",
       subtitle="Numero di risposte corrette",
       caption="Parole non parole - Boxplot")

plot(ggBoxplot)
```

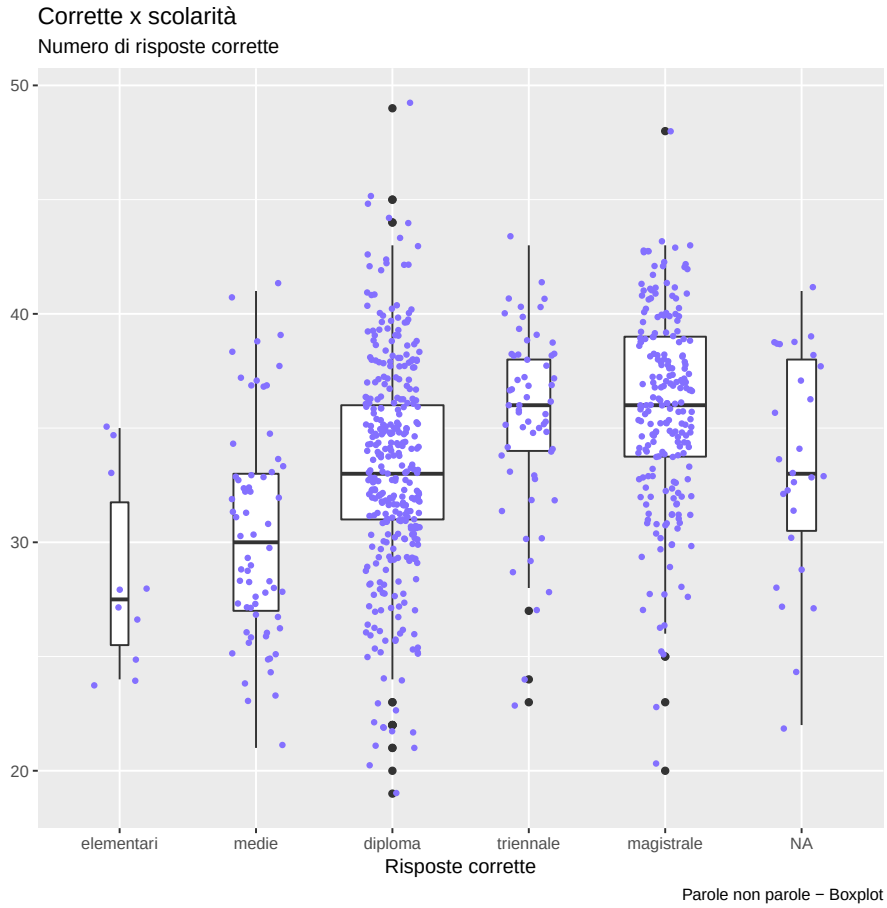


Figura 9.8: Boxplot + Jitter

Violin plot

```
ggViolin <- ggplot(data=partecipanti, aes(scol,corrette)) +
    geom_violin() + # , fill="plum"
  labs(title="Corrette x scolarità",
         x="Risposte corrette",
         y="",
       subtitle="Numero di risposte corrette",
       caption="Parole non parole - Boxplot")

plot(ggViolin)
```

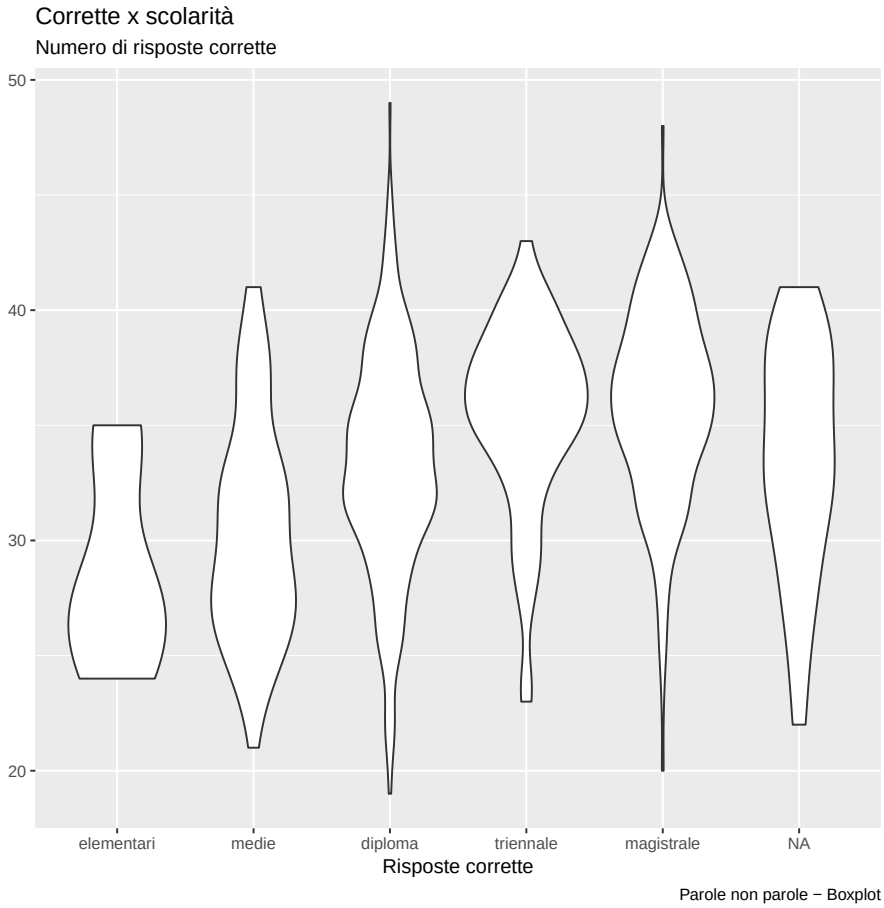


Figura 9.9: Grafico a violino

Scatterplot

```
ggScatter <- ggplot(partecipanti, aes(x=risposte, y=corrette)) +
  geom_point(aes(col=sex, size=as.numeric(scol))) +  # , size=scol
    # geom_jitter(width = .2, size=1) +
```

```
  # geom_smooth(method="loess", se=F) +
  # xlim(c(0, 0.1)) +
  # ylim(c(0, 500000)) +
  labs(
    title="Scatterplot",
    subtitle="Risposte Vs Corrette",
    y="Corrette",
    x="Risposte date",
    caption = "Exp. parole non parole")

plot(ggScatter)
```

```
## Warning: Removed 27 rows containing missing values (geom_point).
```

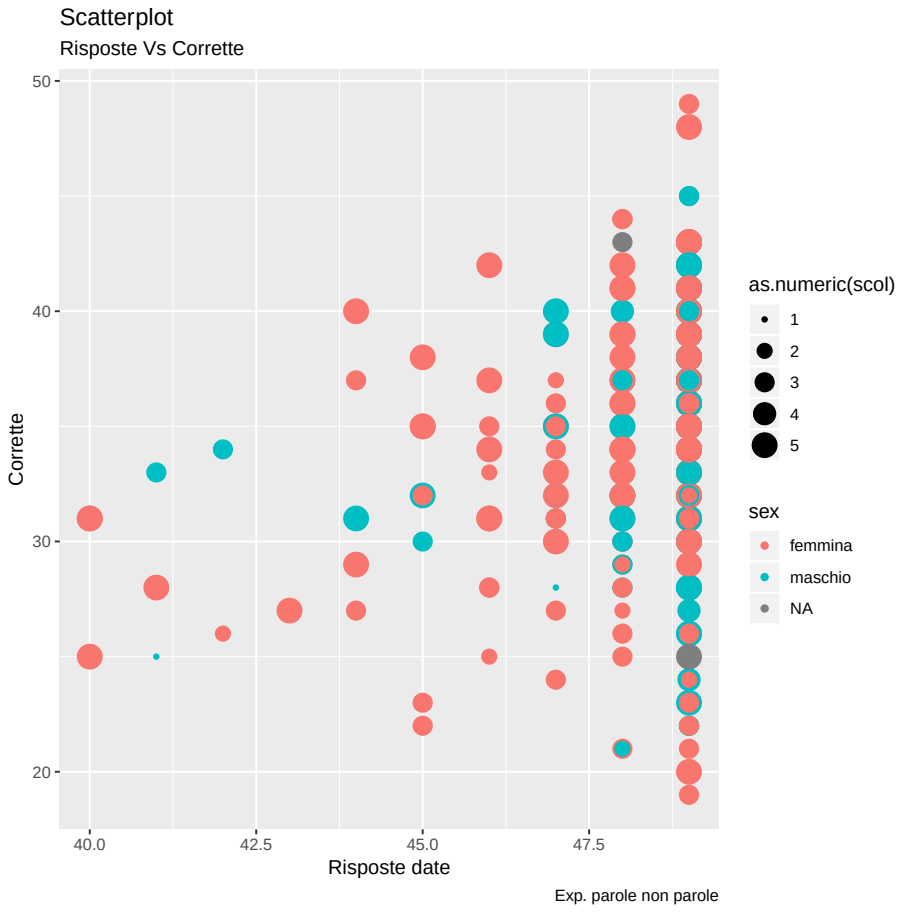


Figura 9.10: Scatterplot

Attenzione a non sovraccaricare il grafico

```
ggJitter <- ggplot(partecipanti, aes(x=risposte, y=corrette)) +
  # geom_point(aes(col=sex, size=as.numeric(scol))) +  # , size=scol
    geom_jitter(aes(col=sex),width = .2, size=1) +
```

```
  labs(
    title="Jitterplot",
    subtitle="Risposte Vs Corrette",
    y="Corrette",
    x="Risposte date",
    caption = "Exp. parole non parole")

plot(ggJitter)
```

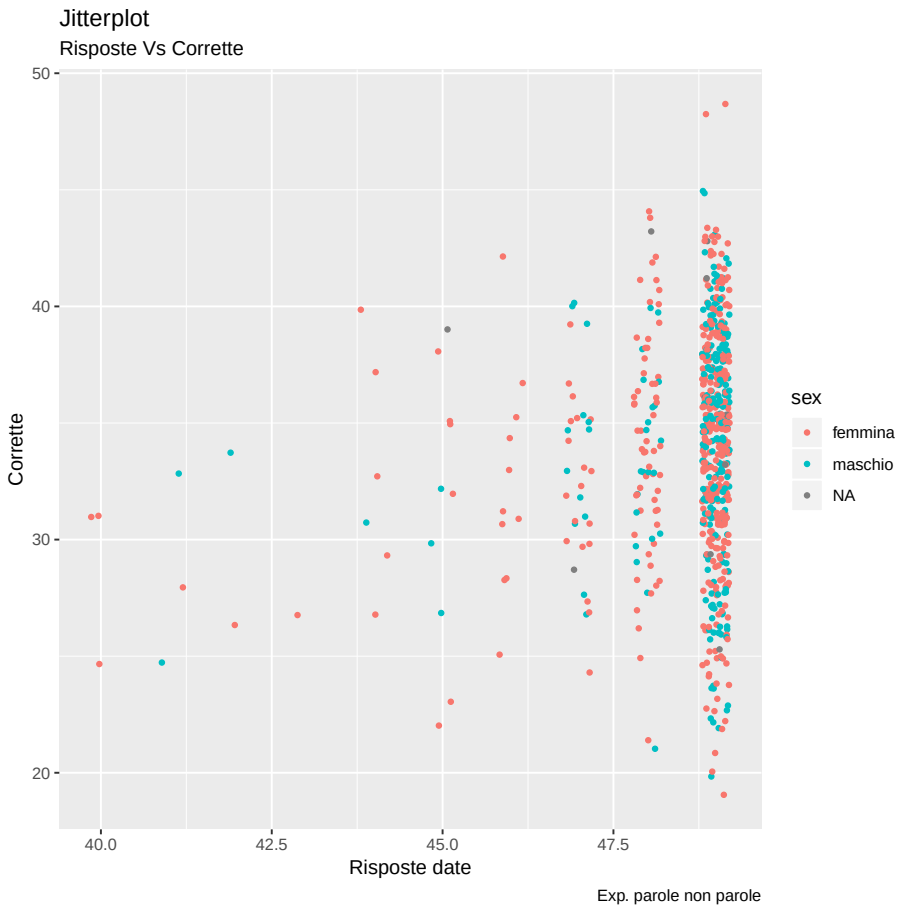


Figura 9.11: Jitter Plot

Parte III

La statistica inferenziale

Capitolo 10

Analisi inferenziale univariata

L'approccio simulativo

Gli errori di campionamento

L'analisi dei dati deve confrontarsi con la gestione degli errori. Se una buona metodologia ed un corretto campionamento possono minimizzare l'impatto degli errori sistematici, gli errori casuali non possono essere eliminati. L'analisi inferenziale permette al ricercatore di stimare l'entità di questi errori, e di capire quanto le misure e le relazioni emerse siano da imputare a tali errori.

L'analisi si basa sul calcolo di alcune statistiche. Nell'analisi univariata si calcolano gli indici di centralità e di dispersione, nelle statistiche bivariate si calcolano delle statistiche capaci di misurare le relazioni fra variabili.

Sia le statistiche uni che bivariate devono tener conto dell'errore di campionamento. Facciamo alcuni esempi.

La media del campione costituisce la migliore stima della media della popolazione (la media è una stima *unbiased*); se dalla stessa popolazione, però, estraggo dieci campioni diversi, otterrò dieci stime differenti.

Un tipico disegno sperimentale consiste nel dividere il campione in 2 gruppi, somministrare un trattamento ad un gruppo (sperimentale), somministrare un diverso trattamento (o un placebo) all'altro gruppo, e misurare l'effetto attraverso una variabile numerica; per valutare l'effetto del trattamento, si misura la differenza fra le medie dei due gruppi. Di nuovo: questa differenza va attribuita al trattamento, o al caso (alla variabilità campionaria)? Infatti, in maniera del tutto paragonabile all'esempio precedente, cosa succederebbe se applicassimo lo stesso trattamento (o nessun trattamento) ai due gruppi? Ci aspettiamo che le medie dei due gruppi siano perfettamente uguali? La risposta è naturalmente no: le medie saranno probabilmente simili, ma non uguali.

Facciamo un terzo esempio: immaginiamo di voler capire se vi è una relazione fra due variabili numeriche. Decidiamo di adottare la statistica della correlazione di Pearson, una misura che si muove nel range $-1 < r < +1$ e dove 0 significa assenza di correlazione. Anche in questo caso, però, nella circostanza di due variabili fra loro indipendenti, non possiamo aspettarci una correlazione esattamente pari a 0.

Distribuzione degli errori

Approccio parametrico

Fortunatamente, gli errori dovuti al caso (e alla varianza campionaria) sono soggetti a delle distribuzioni note (quantomeno per quanto riguarda le statistiche più comuni). La cosiddetta statistica parametrica si basa proprio sul fatto che, se alcuni assunti sono rispettati, la distribuzione dell'errore delle statistiche usate approssima, previo opportuna trasformazione, delle distribuzioni teoriche. Il processo inferenziale utilizza questa proprietà; si calcola la statistica, si opera la trasformazione, e si confronta il risultato con la distribuzione teorica.

Statistiche non parametriche

Lo svantaggio dell'approccio parametrico è che fa delle assunzioni sulle variabili; vi sono delle circostanze in cui queste assunzioni non vengono rispettate. In questi casi, le statistiche parametriche possono essere inaffidabili; a questo punto, diventa opportuno affidarsi a delle famiglie di statistiche non parametriche, il cui vantaggio è quello di fare un minore numero di assunzioni.

Generalmente, l'approccio delle statistiche non parametriche consiste nel trasformare la variabile dipendente, numerica, in una variabile ordinale. La trasformazione consiste nel calcolare il rank, ovvero il valore ordinale della misura.

Approccio simulativo (resampling)

Esiste poi un'altra possibilità: utilizzare il calcolatore per generare la distribuzione dell'errore, e basare il processo inferenziale non sulla distribuzione teorica, ma sulla distribuzione generata.

Questo approccio è relativamente recente, in quanto è computazionalmente *oneroso*, e dunque può essere applicato soltanto con degli strumenti di calcolo potenti. Oggi, però, possono essere applicati agevolmente anche con i comuni computer, e dunque stanno guadagnando crescente popolarità.

L'approccio simulativo ha alcuni vantaggi, il principale dei quali è che fa pochissime assunzioni, e dunque può essere applicato anche nel caso, ad esempio, di distribuzioni che non possono essere ricondotte alle distribuzioni teoriche. Un secondo vantaggio è che l'approccio simulativo è che può essere applicato anche a statistiche non comuni, per le quali non esiste – o non è nota – una distribuzione teorica.

L'approccio simulativo ha infine il vantaggio di essere particolarmente intuitivo, in quanto permette di **mostrare** l'errore di campionamento, la sua distribuzione, e i rispettivi parametri. Questa caratteristica rende l'approccio simulativo particolarmente indicato ai fini didattici, in quanto è possibile simulare la varianza di campionamento, generare la distribuzione campionaria, e confrontarla con la distribuzione teorica. L'approccio computazionale è inoltre un ottimo modo per *giocare* con strumenti come R, prendere confidenza con il linguaggio, e capire cosa succede dietro alle quinte quando si utilizzano le funzioni di testing – parametrici e non parametrici.

Introduzione all'approccio simulativo

Per introdurre l'approccio simulativo, utilizziamo R per fare delle simulazioni che ci permettano di riprodurre, in laboratorio, l'errore di campionamento.

Attraverso la simulazione possiamo creare delle circostanze difficilmente riproducibili nella realtà: possiamo generare una popolazione, generare un numero molto alto di campioni, e valutare qualitativamente (graficamente) e quantitativamente l'errore stocastico di campionamento[^1].

Generare popolazione e campioni

Generare la popolazione

Nel contesto della simulazione, generare una popolazione significa generare un vettore di valori casuali. Se si assume che la distribuzione della popolazione sia normale, è possibile utilizzare la funzione `rnorm` per generare un vettore di numeri distribuiti normalmente intorno ad una media e con una deviazione standard predefinita.

La lunghezza del vettore corrisponde alla numerosità della nostra popolazione virtuale.

Nel nostro esempio, genereremo una popolazione con media teorica 100 e deviazione standard teorica 15 (la scelta di media e deviazione standard è arbitraria).

Generare dei campioni

A partire dal vettore popolazione, è possibile estrarre un vettore campione (di numerosità $m < n$). Per fare questo, R mette a disposizione la funzione `sample(x,m,replace=FALSE)`, dove x è la popolazione e m è la numerosità del campione.

Per visualizzare la distribuzione dell'errore di campionamento, utilizzeremo una popolazione di 10000 valori, e genereremo 200 campioni di numerosità 50.

Dunque n = 10000 (numerosità della popolazione simulata), k = 200 (numero di campioni), m = 50 (osservazioni per campione). Poi, genereremo anche una serie di campioni da 20 osservazioni.

Analisi descrittiva

Una volta generati questi dati, possiamo utilizzare alcune tecniche di analisi univariata per fare delle misurazioni.

In primo luogo possiamo calcolare la media e la deviazione standard della popolazione. Ci aspetteremo che la prima sia prossima a 20 e la seconda a 2. Poi, possiamo visualizzare un istogramma con la distribuzione della popolazione, che ci aspettiamo sia di tipo normale. Per verificarlo, possiamo usare le funzioni qqnorm e qqline.

```
n_pop <- 10000
m50 <- 50
K <- 200
media_teorica <- 100
sd_teorica <- 15
popolazione <- rnorm(n_pop, media_teorica, sd_teorica)
mean(popolazione)
```

```
## [1] 99.90498
```

```
sd(popolazione)
```

```
## [1] 14.90249
```

```
hist(popolazione)
```

Utilizzando qqnorm, valutiamo la normalità della distribuzione

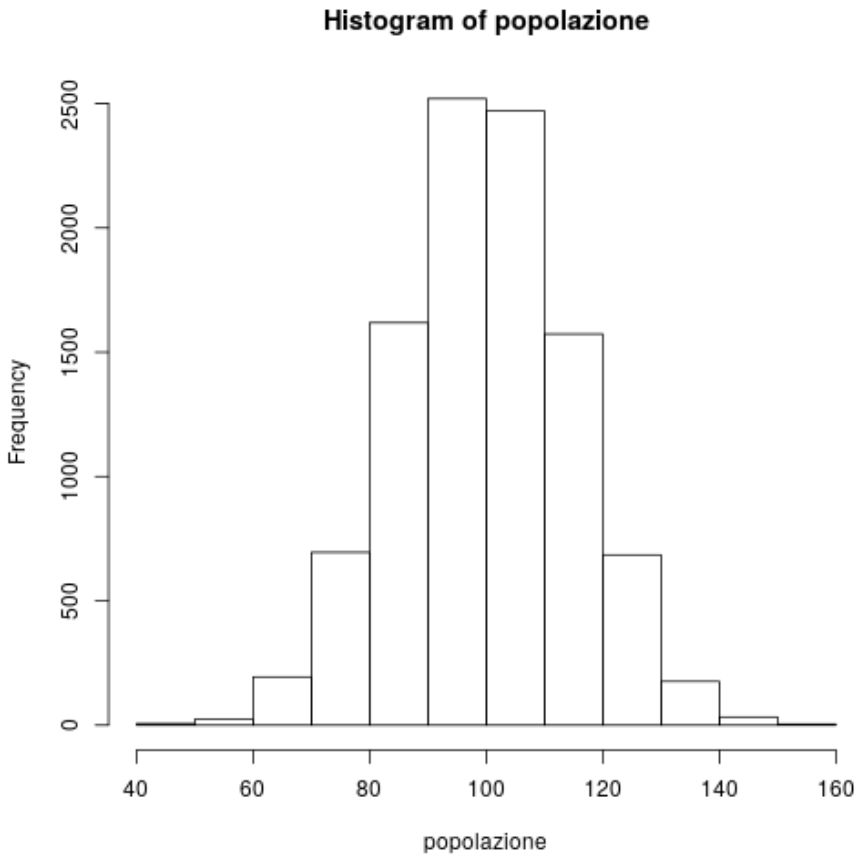


Figura 10.1: plot of chunk boot_popolazione_crea

```
qqnorm(popolazione)
qqline(popolazione, col = 2)
```

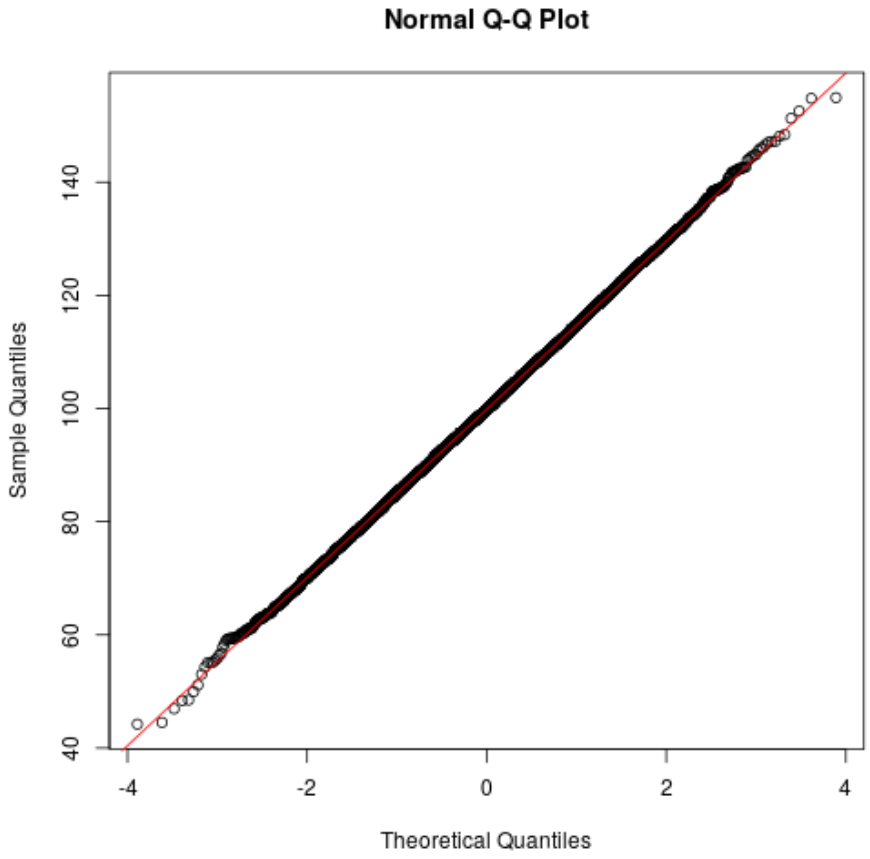


Figura 10.2: plot of chunk boot_qqnorm_pop

Ora, creiamo una matrice 200*50. Ogni riga rappresenta un campione di 50 osservazioni. Popoliamo le righe con la funzione sample, che campiona 50 osservazioni dalla popolazione.

Con `medie_campioni50 <- apply(campioni50, 1, mean)`, calcoliamo la media di ogni campione e la salviamo nel vettore (di lunghezza 200) medie_campioni50. Su questo vettore calcoliamo la media e la deviazione standard (che rappresentano la media delle medie e la deviazione standard delle medie, ovvero l'errore standard.

```
campioni50 <- matrix(nrow = K, ncol = m50)
for (k in 1:K) {
    campioni50[k, ] <- sample(popolazione, m50)
}
medie_campioni50 <- apply(campioni50, 1, mean)
mean(medie_campioni50)
```

```
## [1] 99.63541
```

```
sd(medie_campioni50)
```

```
## [1] 2.043277
```

L'istogramma della distribuzione campionaria

```
hist(medie_campioni50)
```

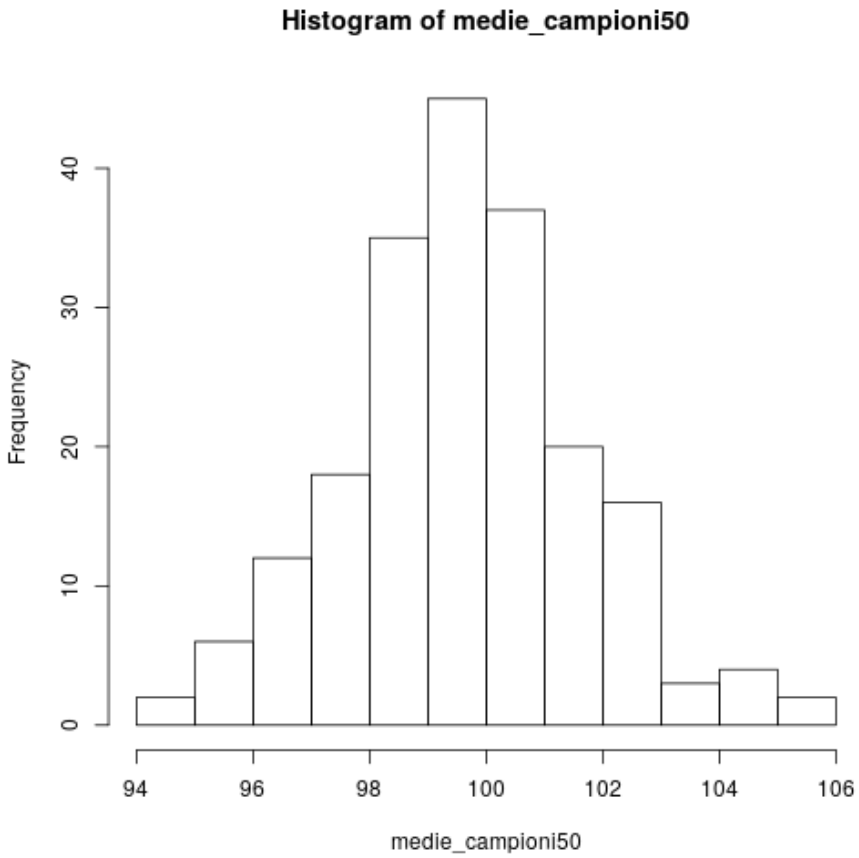


Figura 10.3: plot of chunk boot_hist_medie_50

```
qqnorm(medie_campioni50)
qqline(medie_campioni50, col = 2)
```

Testiamo la normalità della distribuzione delle medie campionarie, usando lo *Shapiro-Wilk normality test.*

```
shapiro.test(medie_campioni50)
```

```
##
##  Shapiro-Wilk normality test
```

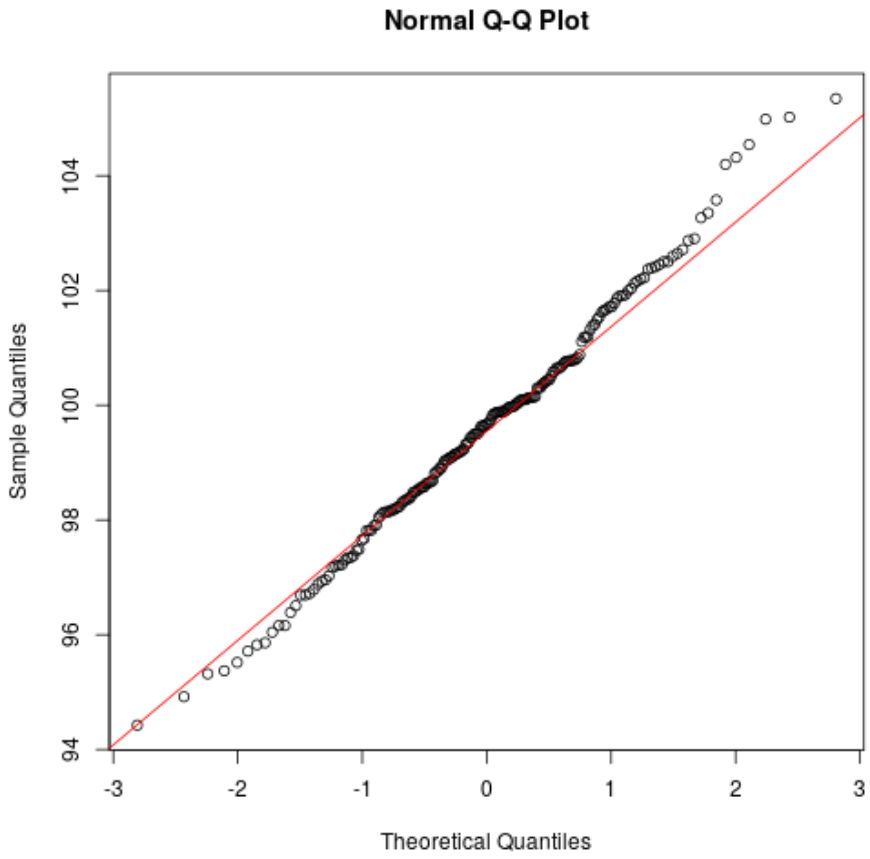


Figura 10.4: plot of chunk boot_qqnorm_c50

```
##
## data:  medie_campioni50
## W = 0.99384, p-value = 0.5766
```

Leggere i risultati

La funzione `shapiro.test` restituisce un :riferimento:. Questo significa che non è rifiutata l'ipotesi di normalità. Dunque, non vi è violazione della normalità della distribuzione.

Campioni di numerosità 20

Ripetiamo la procedura, ma questa volta generiamo campioni di 20 osservazioni. Questo passaggio ci serve per capire se e come cambia la distribuzione campionaria al variare della numerosità del campione.

```
m20 <- 20
campioni20 <- matrix(nrow = K, ncol = m20)
for (k in 1:K) {
    campioni20[k, ] <- sample(popolazione, m20)
}
medie_campioni20 <- apply(campioni20, 1, mean)
mean(medie_campioni20)
```

```
## [1] 100.0424
```

```
sd(medie_campioni20)
```

```
## [1] 3.392602
```

Intervallo di confidenza

A partire da queste simulazioni, possiamo introdurre il concetto di intervallo di confidenza.

Conoscendo la popolazione, possiamo prevedere il valore esatto della media di un campione di numerosità m estratto casualmente? La risposta, abbiamo visto, è negativa. Possiamo però stimare un intervallo entro il quale possiamo prevedere dove questa media verrà a cadere.

Nemmeno l'intervallo, però, può garantirci la sicurezza al 100%, in quanto non possiamo escludere di incorrere in campionamenti particolarmente sbilanciati da una parte o dall'altra.

La cosa più ragionevole da fare è quella di stabilire un livello di rischio percentuale accettabile, e di calcolare l'intervallo in base a questo rischio.

Calcolare il range

Detto in altri termini, possiamo calcolare i valori minimo e massimo, e dunque il range, entro il quale, probabilmente, il (100-rischio)% delle medie dei campioni andrà a cadere.

Se, ad esempio, consideriamo accettabile un rischio del 5%, calcoleremo il range entro il quale si collocano le medie del 95% dei campioni estratti. Questo ci permette di tagliare le code estreme, a destra e a sinistra, della distribuzione.

Per fare questo, tagliamo il 2.5% di campioni con media più bassa e il 2.5% di campioni con media più alta.

La media del campione con media più bassa rimanente, e la media del campione con media più alta rimanente, costituiscono il range che cercavamo, ovvero l'intervallo di confidenza. Per calcolare questi valori, sarà sufficiente estrarre i percentili 2.5 e 97.5 dalla distribuzione delle medie dei campioni.

```
confidenza_campioni50 <- quantile(medie_campioni50, probs = c(0.025, 0.975
confidenza_campioni20 <- quantile(medie_campioni20, probs = c(0.025, 0.975
confidenza_campioni50
```

```
##      2.5%     97.5%
##  95.70891 104.20070
```

```
confidenza_campioni20
```

```
##      2.5%     97.5%
##  93.65156 106.29324
```

```
differenza_campioni50 <- confidenza_campioni50[2] - confidenza_campioni50[
differenza_campioni20 <- confidenza_campioni20[2] - confidenza_campioni20[
```

Come possiamo notare, l'intervallo di confidenza della distribuzione campionaria è diverso, cambiando la numerosità dei campioni. Nel caso di campioni di numerosità 20, il range è di 12.6416796, mentre per i campioni di numerosità 50, è di 8.4917889.

Confrontare le due distribuzioni

Ora, usiamo la funzione density per confrontare le due distribuzioni, quella dei campioni di 50 osservazioni, e quella dei campioni di 20. Abbiamo disegnato anche le due medie (le righe verticali) e gli intervalli di confidenza (le righe orizzontali).

```
density20 <- density(medie_campioni20)
density50 <- density(medie_campioni50)
plot(density20, ylim = c(0, max(density50$y)), col = 2, lty = 2)
abline(v = mean(medie_campioni20), col = 2, lty = 2)
lines(x = confidenza_campioni20, y = c(0.2, 0.2), col = 2, lty = 2)
lines(density50, col = 3, lty = 4)
abline(v = mean(medie_campioni50), col = 3, lty = 4)
lines(x = confidenza_campioni50, y = c(0.3, 0.3), col = 3, lty = 4)
```

Possiamo notare come la distribuzione dei campioni di 20 osservazioni sia più larga di quella da 50. Corrispondentemente, anche i due intervalli di confidenza sono diversi.

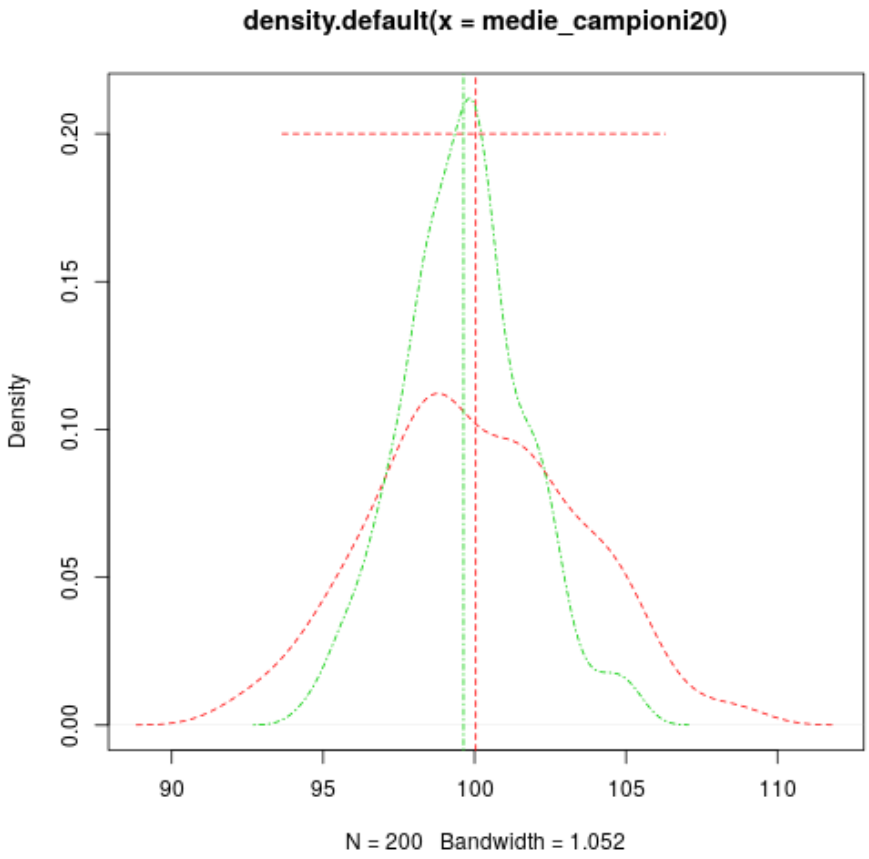


Figura 10.5: plot of chunk boot_density_20_50

Numerosità dei campioni e varianza

La varianza della distribuzione delle medie dei campioni costituisce una stima dell'errore di campionamento: più bassa la varianza (e la sd), più basso l'errore, e viceversa.

Dalle nostre simulazioni, si può intuire che l'entità dell'errore è legato alla numerosità del campione.

Bootstrapping

Generare molti campioni da un campione

La simulazione presentata nei paragrafi precedenti, seppur utile da un punto di vista didattico, è irrealistica: il ricercatore non può lavorare sull'intera popolazione, ma soltanto un campione.

Inoltre, il compito del ricercatore è quello di stimare la media della popolazione partendo dal campione, e non il contrario.

Partiamo da due osservazioni

- Se abbiamo a disposizione soltanto il nostro campione, la miglior stima della media della popolazione è la media del campione stesso.
- La distribuzione del campione dovrebbe "assomigliare" (al netto dell'errore statistico) alla distribuzione della popolazione. E, dunque, la distribuzione del campione è la miglior stima che abbiamo della distribuzione della popolazione.

Se assumiamo che la media della popolazione sia pari a quella del campione, e che anche la distribuzione sia paragonabile, possiamo immaginare di generare numerosi campioni fittizzi a partire dal campione noto.

Bootstrapping

Questa tecnica è nota come bootstrapping, e permette di calcolare l'intervallo di confidenza di un parametro (quale, ad esempio, la media).

Per generare un nuovo campione dal campione esistente, basta estrarre a caso m osservazioni dal campione originale.

Naturalmente, l'estrazione dev'essere *con ripetizione*. In caso contrario, il nuovo campione sarebbe identico. Dunque, alcuni elementi verranno estratti più di una volta, altri nessuna.

Percentili e intervallo di confidenza

In questo modo, possiamo generare dei nuovi campioni dal campione esistente. Anche in questo caso possiamo calcolare la media per ogni nuovo campione, e calcolare la distribuzione delle medie.

A partire da questa distribuzione, possiamo calcolare l'intervallo di confidenza, partendo dai percentili. Useremo i percentili 2.5 e 97.5 per un intervallo di confidenza del 95% (e un errore del 5%).

Per iniziare, prendiamo ora il primo dei 200 campioni generati, ed usiamolo per il bootstrapping. Calcoliamo il vettore delle medie dei bootstrap. Calcoliamo la media delle medie.

```
campioneA <- campioni50[1, ]
mean(campioneA)
```

```
## [1] 100.5788
```

```
bootstraps <- matrix(sample(campioneA, size = 10000, replace = TRUE), nrow = k, nco
medie_bootstraps <- apply(bootstraps, 1, mean)
confidenza_bootstraps <- quantile(medie_bootstraps, probs = c(0.025, 0.975))
mean(medie_bootstraps)
```

```
## [1] 100.5243
```

```
confidenza_bootstraps
```

```
##      2.5%     97.5%
##  96.36295 104.74116
```

Confronto fra le distribuzioni

Nella sezione precedente, abbiamo visto la situazione *ideale ma improbabile*: conoscere l'intera popolazione, estrarre k campioni, calcolare la media per ognuno dei campioni; in questo modo abbiamo la vera distribuzione campionaria, di cui possiamo calcolare media e varianza.

In questa sezione, vediamo una situazione più realistica: abbiamo un campione, e lavoriamo su quello. Attraverso il bootstrapping, generiamo k campioni *virtuali*, e calcoliamo la *distribuzione campionaria virtuale*. Per capire se il secondo algoritmo, realistico ma virtuale, produce risultati *robusti*, confrontiamo le due medie, i due intervalli di confidenza e le due distribuzioni nel grafico **??**.

```
densityboot <- density(medie_bootstraps)
plot(density50, ylim = c(0, max(c(density50$y, densityboot$y))), col = 2, lty = 2)
abline(v = mean(medie_campioni50), col = 2, lty = 2)
lines(x = confidenza_campioni50, y = c(0.2, 0.2), col = 2, lty = 2)
lines(densityboot, col = 3, lty = 4)
abline(v = mean(medie_bootstraps), col = 3, lty = 4)
lines(x = confidenza_bootstraps, y = c(0.3, 0.3), col = 3, lty = 4)
abline(v = mean(popolazione))
```

Le due distribuzioni, seppure non identiche, sono molto simili.\ Anche gli intervalli di confidenza sono paragonabili: gli intervalli calcolati sui 200 campioni sono pari a 19.44196 e 20.46475; gli intervalli calcolati attraverso il bootstrapping sono 19.56849 e 20.50061.

Possiamo dunque intuitivamente affermare che il metodo del bootstrapping riesce a simulare, in maniera piuttosto precisa, la distribuzione campionaria.

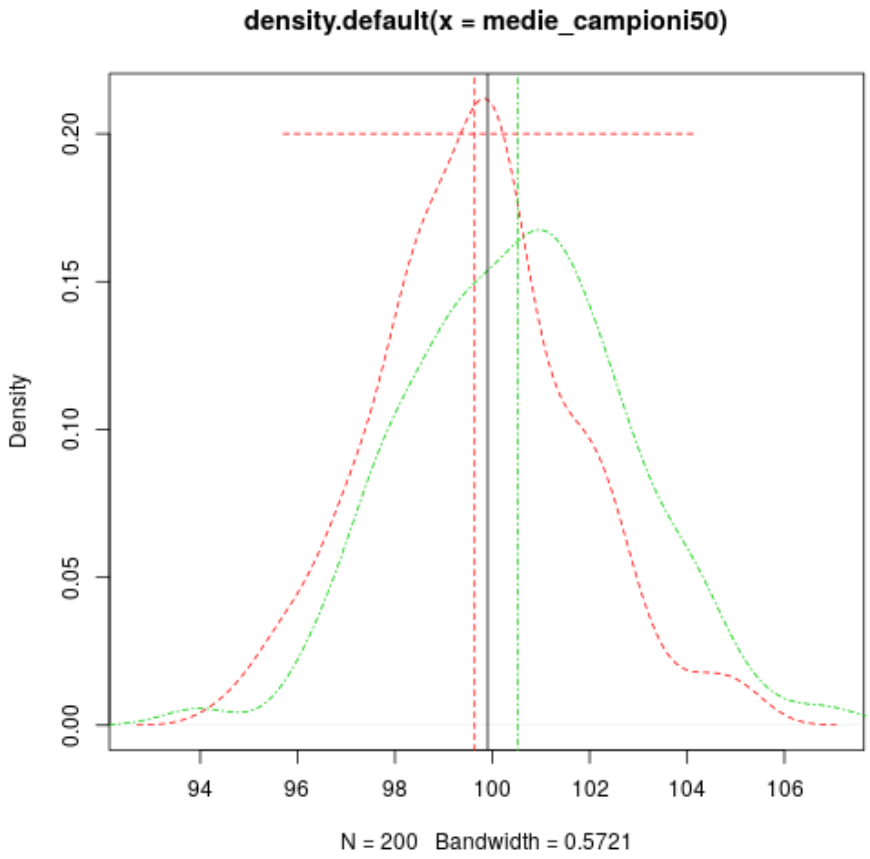


Figura 10.6: plot of chunk boot_density_boot

Usare l'approccio parametrico

Il calcolo parametrico dell'intervallo di confidenza è l'argomento del prossimo capitolo. Qui, ci limitiamo ad anticipare i risultati del test parametrico `t.test`.

```
t_campioneA <- t.test(campioneA)
t_campioneA$conf.int
```

```
## [1]  95.76952 105.38811
## attr(,"conf.level")
## [1] 0.95
```

```
confidenza_campioni50
```

```
##      2.5%     97.5%
##  95.70891 104.20070
```

Il calcolo parametrico, resitiuisce un intervallo di confidenza di 19.47571 e 20.44586. Come vediamo, il metodo parametrico e il metodo che usa il bootstrap non restituiscono risultati uguali, ma molto simili.

```
n <- 10000
m50 <- 50
K <- 200
media_teorica <- 20
sd_teorica <- 2
popolazione <- rnorm(n, media_teorica, sd_teorica)

campioni50 <- matrix(nrow = K, ncol = m50)
for (k in 1:K) {
    campioni50[k, ] <- sample(popolazione, m50)
}
medie_campioni50 <- apply(campioni50, 1, mean)

m20 <- 20
campioni20 <- matrix(nrow = K, ncol = m20)
for (k in 1:K) {
    campioni20[k, ] <- sample(popolazione, m20)
}
medie_campioni20 <- apply(campioni20, 1, mean)
density20 <- density(medie_campioni20)
density50 <- density(medie_campioni50)
campioneA <- campioni50[1, ]
```

Intervallo di confidenza, calcolo parametrico

L'intervallo di confidenza

I concetti di base

Riprendiamo alcuni dei concetti alla base del calcolo dell'intervallo di confidenza:

- il fine del calcolo è di stimare il parametro di una popolazione, partendo da un campione
- la statistica calcolata sul campione viene usata come stima del parametro della popolazione: è una stima puntuale
- le metodologie che stimano l'errore sono finalizzate a stimare l'accuratezza della stima
- l'intervallo di confidenza è il range entro il quale si prevede si collochi il parametro della popolazione

Bias e errori non sistematici

Nella definizione di errore, dobbiamo distinguere fra gli errori sistematici e gli errori non sistematici

- l'errore sistematico è definito anche bias: una inaccuratezza dovuta ad un errore che sistematicamente *alza* o *abbassa* la stima.

- l'errore non sistematico, al contrario, tende ad aumentare la varianza delle osservazioni.

L'accuratezza di un processo di stima è influenzata sia dal bias e dalla varianza

Accuratezza, efficienza

- Per aumentare l'accuratezza, è necessario tentare di ridurre sia il bias che la varianza.
- A parità di bias, minore è la varianza dovuta all'errore e maggiore l'efficienza.
- Un buon processo di stima ha bias nullo e varianza bassa.

La media di un campione, ad esempio, è una stima unbiased, in quanto la distribuzione delle medie si distribuisce normalmente intorno alla media della popolazione.

L'intervallo di confidenza

La percentuale di confidenza si riferisce alla probabilità che il valore del parametro della popolazione cada nell'intervallo identificato in base alla nostra stima.

L'intervallo di confidenza copre, con una determinata probabilità, il parametro della popolazione, non noto.

Come abbiamo visto, l'intervallo può essere calcolato con un metodo non parametrico, il bootstrapping.

L'intervallo di confidenza può essere calcolato anche con dei metodi parametrici.

Assunto di normalità

Poiché questi metodi fanno delle assunzioni sulla distribuzione della popolazione (e del campione), prima di applicarle è necessario verificare questa assunzione.

Detto in altri termini, prima di calcolare l'intervallo di confidenza è necessario verificare che la distribuzione del campione non si discosti dalla distribuzione normale.

Una volta stabilita la normalità del campione e assunta la normalità della popolazione, possiamo procedere con il calcolo.

La simulazione

Riprendiamo la distribuzione di 200 campioni di numerosità 50, generata nel capitolo precedente. Abbiamo visto che la distribuzione ha una forma che si approssima a quella normale, con media che si approssima alla media della popolazione. Formalmente: $\mu_{\bar{X}} \approx \mu$

Varianza della distribuzione delle medie

Come abbiamo osservato, la varianza della distribuzione delle medie dei campioni cambia a seconda della numerosità del campione. Più in particolare, la varianza della distribuzione delle medie tende ad essere pari alla varianza della popolazione / la numerosità delle osservazioni dei campioni:

$$\sigma^2_{\bar{X}} = \frac{\sigma^2}{m}, \sigma_{\bar{X}} = \frac{\sigma}{\sqrt{m}} \tag{10.1}$$

Questa misura viene definita errore standard.

Proviamo a verificare 10.1 con le nostre simulazioni.

```
var(popolazione)/var(medie_campioni50)
```

```
## [1] 56.1371
```

```
var(popolazione)/var(medie_campioni20)
```

```
## [1] 22.40829
```

Possiamo notare che il rapporto fra la varianza della popolazione e la varianza campionaria è ≈ 50 nel primo gruppo (campioni con numerosità 50), e ≈ 20 nel secondo, dove i campioni sono di numerosità 20.

```
errore_standard <- sd(popolazione)/sqrt(m50)
plot_range <- seq(-3,3,by=.05)
plot_range <- plot_range * errore_standard + mean(popolazione)
prob_dist <- dnorm(plot_range,mean=mean(popolazione),sd=errore_standard)
plot(plot_range,prob_dist,type="l")
lines(density50,col=3)

score <- qnorm (0.025,mean=mean(popolazione),sd=errore_standard)
xx <-c(plot_range[plot_range<score],score,score,plot_range[1])
yy <- c(dnorm(xx[1:(length(xx)-2)],mean=mean(popolazione),sd=errore_standa
polygon(xx, yy, col="gray", border = "red")

score <- qnorm (0.975,mean=mean(popolazione),sd=errore_standard)
xx <-c(score,score,plot_range[plot_range>score],plot_range[length(plot_ran
yy <- c(0,prob_dist[(length(plot_range)-length(xx)+3):length(plot_range)],
polygon(xx, yy, col="gray", border = "red")
```

Dalla simulazione alla stima

Nella circostanza della simulazione, conosciamo la popolazione, ne conosciamo la distribuzione, la media, la varianza. Grazie all'equazione 10.1 possiamo stimare la distribuzione campionaria. Il passaggio logico dei prossimi paragrafi sarà il seguente:

1. stimiamo la distribuzione campionaria conoscendo media e varianza della popolazione;
2. stimiamo la distribuzione campionaria stimando la media, conoscendo la varianza;

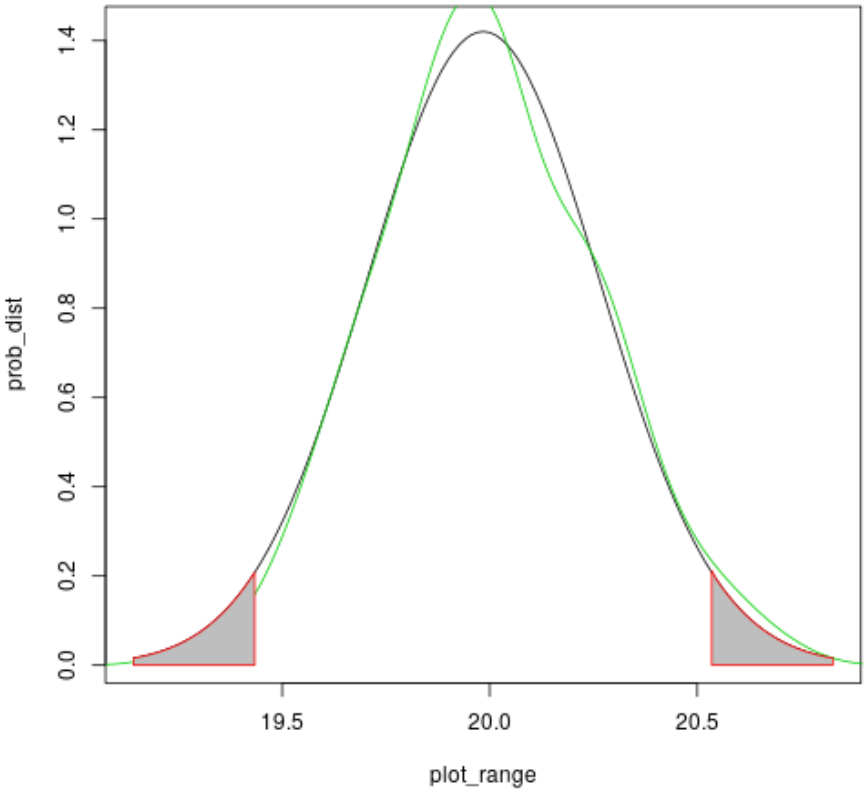


Figura 10.7: plot of chunk unipara_3

3. infine, la situazione più realistica: stimiamo la distribuzione campionaria stimando media e varianza.

Media e varianza nota

Assumiamo, per ora, di conoscere media e varianza della popolazione.

Conoscendo la media della popolazione e la sua varianza, possiamo ricostruire la distribuzione delle medie dei campioni, che sarà una distribuzione (teorica) normale con media $\mu_{\bar{X}} = \mu$ e deviazione standard $\sigma_{\bar{X}} = \frac{\sigma}{\sqrt{m}}$ (errore standard)

Sovrapporre le distribuzioni

Per verificare graficamente la corrispondenza, sovrapponiamo il grafico della distribuzione teorica appena calcolata sulla distribuzione campionaria della simulazione (fig. ??).

:todo: manca la figura?

A partire da questa distribuzione, possiamo calcolare i percentili 2.5 e 97.5, che corrispondono all'intervallo di confidenza del 95%.

```
media_pop <- mean(popolazione)
confidenza1 <- c(qnorm(0.025, media_pop, errore_standard), qnorm(0.975, me
confidenza1
```

```
## [1] 19.43377 20.53491
```

Ricordiamo che `qnorm(p,m,sd)` calcola il valore che, data media `m` e deviazione standard `sd`, copre un'area pari a `p`.

Varianza nota, media ignota

Nella realtà, però, noi non conosciamo né la media né la varianza della popolazione.

Assumiamo, per ora, di conoscere ancora la varianza della popolazione, ma non la sua media. A questo punto, l'intervallo di confidenza avrà la stessa ampiezza di quello calcolato prima, ma sarà centrato attorno non alla media della popolazione (che non conosciamo) ma alla media del campione: usiamo $\bar{X}$ come stima di μ.

R: calcolo dell'intervallo

A questo punto, il calcolo dell'intervallo di confidenza sarà il seguente:

```
media_campione <- mean(campioneA)
confidenza2 <- c(qnorm(0.025, media_campione, errore_standard), qnorm(0.97
confidenza2
```

```
## [1] 19.59141 20.69255
```

Varianza e media ignota

Arriviamo, ora, all'ipotesi più realistica: conosciamo media e deviazione standard del campione, ma non quelle della popolazione.

Il passaggio più logico parrebbe quello di usare $s_{\bar{X}}$, la deviazione standard del campione, come stima di σ, la deviazione standard della popolazione. In realtà, la deviazione standard del campione è più bassa di quella della popolazione: se utilizziamo la prima al posto della seconda otteniamo un range irrealisticamente troppo stretto.

Per correggere questo bias (è un errore sistematico) si utilizza, al posto della distribuzione normale, la distribuzione t di Student.

La distribuzione t di Student

La t di Student è una classe di distribuzioni, che si basano sui gradi di libertà. Nel caso del nostro intervallo di confidenza, i gradi di libertà sono pari a m-1. [1]

R: calcolo dal t di Student

Calcoliamo l'errore standard stimato a partire dalla deviazione standard del campione. Usiamo poi la funzione `qt` per calcolare i quantili 0.025 e 0.975 della distribuzione t con 50-1 gradi di libertà. Il risultato, sarà l'intervallo di confidenza.

```
errore_standard_stimato <- sd(campioneA)/sqrt(m50)
confidenza3 <- c(qt(0.025, df = 49) * errore_standard_stimato + mean(campioneA), qt
confidenza3
```

```
## [1] 19.59116 20.69279
```

L'uso della funzione `t.test` era stato anticipato in **??**. Il risultato – in termini di intervallo di confidenza – del test è quello calcolato con il codice appena mostrato.

Confronto fra un campione ed una popolazione

Abbiamo appena visto la statistica parametrica per calcolare l'intervallo di confidenza della stima del parametro della media di una popolazione, a partire da un campione. Nel capitolo precedente abbiamo usato il bootstrapping, in questo il t test. Il test t di Student, però, può essere usato anche per stimare se un campione appartiene ad una popolazione la cui media è nota.

In questo caso, si tratta di stimare se il campione è stato estratto da una popolazione con media μ oppure no.

In termini inferenziali, abbiamo le due ipotesi:

- ipotesi nulla, H_0: non vi è differenza significativa fra la media del campione, $\bar{X}$ e la media della popolazione, μ;

[1] quando df>120, la distribuzione t di Student tende ad approssimarsi alla distribuzione normale.

- ipotesi alternativa, H_A: la differenza fra le due medie è significativa, e dunque il campione non appartiene alla popolazione.

Per giungere alla nostra decisione inferenziale, ci viene in soccorso proprio l'intervallo di confidenza: se la media della popolazione cade all'interno dell'intervallo, non possiamo rifiutare l'ipotesi nulla. In caso contrario, rifiutiamo l'ipotesi nulla e accettiamo l'ipotesi alternativa.

Il p-value

Vi è una possibilità complementare: calcolare il p-value. In termini inferenziali, il p-value ci dice la probabilità di incorrere in un errore di tipo I nel caso di rifiuto dell'ipotesi nulla.

Di fatto, quello che calcoliamo è la probabilità che il nostro campione possa essere stato estratto da una popolazione la cui media è pari ad un valore predefinito.

Decidiamo per un errore di I tipo pari ad $\alpha = 0.05$ e, nella nostra simulazione, assumiamo un'ipotesi a due code.

Il primo passaggio, è quello di calcolare la differenza, in termini assoluti, fra la media del campione e quella della popolazione. Il secondo passaggio è di trasformare questa distanza in punti t, attraverso la formula `distanza / errore standard`.

Infine, confrontiamo questo punteggio con la distribuzione t di Student, con gradi di libertà pari a m-1.

Primo esempio

Come primo esempio, calcoliamo il p-value della differenza fra la media della popolazione e quella del campione campioneA. Poiché il campione è stato estratto dalla popolazione, ci aspettiamo che il p-value sia alto (superiore ad α).

Calcolo con R

```
distanza1 <- abs(mean(campioneA) - mean(popolazione))
t1 <- distanza1/errore_standard_stimato
p_value1 <- (1 - pt(t1, df = 49)) * 2
t1
```

```
## [1] 0.5751085
```

```
p_value1
```

```
## [1] 0.5678495
```

R: uso del t.test

Dopo aver calcolato manualmente il p-value, ci affidiamo alla funzione `t.test`.

```
t.test(campioneA, mu = mean(popolazione))
```

```
##
##  One Sample t-test
##
## data:  campioneA
## t = 0.57511, df = 49, p-value = 0.5678
## alternative hypothesis: true mean is not equal to 19.98434
## 95 percent confidence interval:
##  19.59116 20.69279
## sample estimates:
## mean of x
##  20.14198
```

L'algoritmo usato dalla funzione `t.test` è leggermente diverso: non viene usato il valore assoluto della differenza, e il punteggio t in questo caso è negativo. Il principio rimane comunque lo stesso – e il risutato anche.

Leggere l'output

La funzione `t.test` ci restituisce tutte le informazioni di cui abbiamo bisogno: la statistica calcolata: *One Sample t-test*; i gradi di libertà: `df = 49`; il p-value: `p-value = 0.9105`; l'intervallo di confidenza al 95%. Poiché $p > \alpha = 0.05$, non rifiutiamo l'ipotesi H_0.

Secondo esempio

Proviamo ora a confrontare il nostro campione con una media più alta: 20.8. In questo caso, sapendo che 20.8 è esterno all'intervallo di confidenza, ci aspettiamo un p-value inferiore ad $\alpha = 0.05$.

```
distanza2 <- abs(mean(campioneA) - 20.8)
t2 <- distanza2/errore_standard_stimato
p_value2 <- (1 - pt(t2, df = 49)) * 2
t2
```

```
## [1] 2.400719
```

```
p_value2
```

```
## [1] 0.02020456
```

R: uso del t.test

Dopo aver calcolato manualmente il p-value, ci affidiamo alla funzione `t.test`.

```
t.test(campioneA, mu = 20.8)
```

```
##
##  One Sample t-test
##
## data:  campioneA
## t = -2.4007, df = 49, p-value = 0.0202
```

```
## alternative hypothesis: true mean is not equal to 20.8
## 95 percent confidence interval:
##  19.59116 20.69279
## sample estimates:
## mean of x
##  20.14198
```

Poiché, in questo caso, $p - value = 0.001 < \alpha = 0.05$, rifiutiamo l'ipotesi H_0 e accettiamo l'ipotesi alternativa H_A.

Capitolo 11

Confronto fra variabili nominali

Variabili nominali

Statistiche sulle variabili nominali

A partire da una variabile nominale è possibile ottenere un numero limitato di statistiche descrittive, univariate. Quello che è possibile fare, in pratica, è creare una tabella di contingenza unidimensionale, ovvero un vettore la cui lunghezza è pari al numero di livelli, e dove il valore di ogni cella è pari al numero di osservazioni che appartengono alla corrispondente categoria.

A partire da questa rappresentazione, è possibile calcolare l'indice centrale della moda.

Distribuzioni categoriali

Da un punto di vista inferenziale, è possibile utilizzare la distribuzione delle osservazioni del campione come stima della frequenza di ogni categoria nella popolazione.

Confronto di una distribuzione campionaria con una distribuzione teorica

Vi sono circostanze, in cui l'ipotesi di ricerca è finalizzata a confrontare la distribuzione categoriale di un campione, rispetto ad una distribuzione teorica riguardante una popolazione.

Un esempio: distribuzione occupati

Facciamo un esempio. Assumiamo che, in Italia, il 17% della popolazione occupata lavori nell'agricoltura, il 51% nell'industria, e il rimanente 32% nel terziario (le cifre sono assolutamente inventate).

Potremmo chiederci se, in una determinata provincia, la distribuzione categoriale si discosta o meno dalla distribuzione ipotizzata.

Per fare questo, possiamo estrarre un campione dalla popolazione provinciale degli occupati, creare la relativa tabella di contingenza, e capire se la distribuzione del campione rispetta le cosiddette frequenze attese.

Calcolo frequenze attese

Il calcolo delle frequenze attese è semplice: basta moltiplicare la probabilità attesa (ovvero la probabilità che, data la popolazione di riferimento, venga estratta una osservazione appartenente ad una determinata categoria) per la numerosità del campione.

Se immaginiamo di utilizzare un campione di 100 persone, la frequenza attesa sarà di 17 persone che lavorano nell'agricoltura, 51 nell'industria, 33 nel terziario.

Stima dell'errore

Errore di campionamento

A causa dell'errore non sistematico di campionamento, però, difficilmente le frequenze osservate saranno uguali alle frequenze attese.

Il compito della statistica inferenziale, in questo caso, sarà quello di stabilire se la differenza fra le frequenze osservate e quelle attese sono da attribuire o meno al caso. Nella prima ipotesi, non si rifiuta l'ipotesi nulla, secondo cui non vi è differenza significativa fra le frequenze osservate e quelle attese, e dunque si assume che non vi sia differenza fra la distribuzione della provincia in esame e quella nazionale, di riferimento.

Nella seconda ipotesi, si rifiuterà l'ipotesi nulla, e di conseguenza si assumerà che vi è una differenza significativa fra la distribuzione di frequenza del campione e quella della popolazione.

Nel nostro esempio, si assumerà che una differenza fra il nostro campione e la popolazione nazionale di riferimento, e pertanto che la distribuzione di frequenza nelle tre categorie è, nella popolazione provinciale, diversa da quella nazionale.

Stimare la differenza fra le fequenze

Per fare questo, abbiamo bisogno di una misura che ci permetta di calcolare la differenza fra due tabelle.

Informalmente, abbiamo bisogno di una misura che stimi la distanza fra due tabelle, e dunque che sia pari a zero se le due tabelle sono uguali, che sia positiva se le due tabelle sono differenti, e che cresca al crescere delle differenze.

In letteratura, vengono citate tre possibili misure.

- il χ^2 di Pearson :

$$\chi^2 = \sum_i^r \sum_j^c \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{11.1}$$

- Il *likelihood ratio chi square* :

$$G^2 = 2 \sum_i^r \sum_j^c (O_{ij} \times log(\frac{O_{ij}}{E_{ij}})) \tag{11.2}$$

- il test di Fisher.

La misura più nota, che utilizzeremo, è il χ^2 di Pearson 11.1.

Numerosità delle frequenze attese

Com'è intuibile, sia l'equazione 11.1 che la 11.2 sono poco adatte a circostanze in cui la frequenza attesa di una delle celle è molto bassa (generalmente, si assume che 5 sia la frequenza attesa minima).

La simulazione

**** spiegare

Per introdurre la stima della probabilità che la distribuzione di frequenze di un campione sia significativamente diverso dalla distribuzione attesa, useremo di nuovo il metodo della simulazione.

Più in particolare, andiamo a generare k campioni di una variabile categoriale con una determinata frequenza attesa, e per ogni campione misuriamo, utilizzando l'equazione 11.1, la distanza fra la distribuzione osservata e quella attesa.

La distribuzione delle distanze dei k campioni ci permette di stimare la distribuzione dell'errore di campionamento, di stabilire i valori critici che corrispondono ad un errore α, e al calcolo del p-value di una distribuzione.

Generiamo un'urna

Generiamo un'urna, di 100 valori, da cui estrarre i campioni.

```
atteso <- c(rep(1,17),rep(2,51),rep(3,32))
length(atteso)
```

```
## [1] 100
```

```
t_atteso <- table(atteso)
prob_attesa <- t_atteso/length(atteso)
```

Di fatto, in questa simulazione utilizziamo il metodo delle permutazioni .

Generiamo i campioni

Generiamo 10.000 tabelle da 100 elementi, estratti (con ripetizione) dall'urna sopra creata. Per ogni tabella, calcoliamo la distanza dalla tabella di riferimento (t_atteso) usando la formula del χ^2, e la salviamo in un vettore, distanza_oss (distenza osservata).

R: generazione dei campioni

```
distanza_oss <-vector(mode = "numeric", length = 10000) # distanze osserva

for (ciclo in 1:length(distanza_oss)) {
    campione <- sample(atteso,100,replace=TRUE)
    t_campione <- table(campione)
    chi_quadro <- sum(((t_campione-t_atteso)^2)/t_atteso)
    distanza_oss [ciclo] <- chi_quadro
}

hist(distanza_oss,freq=FALSE,breaks=20)
```

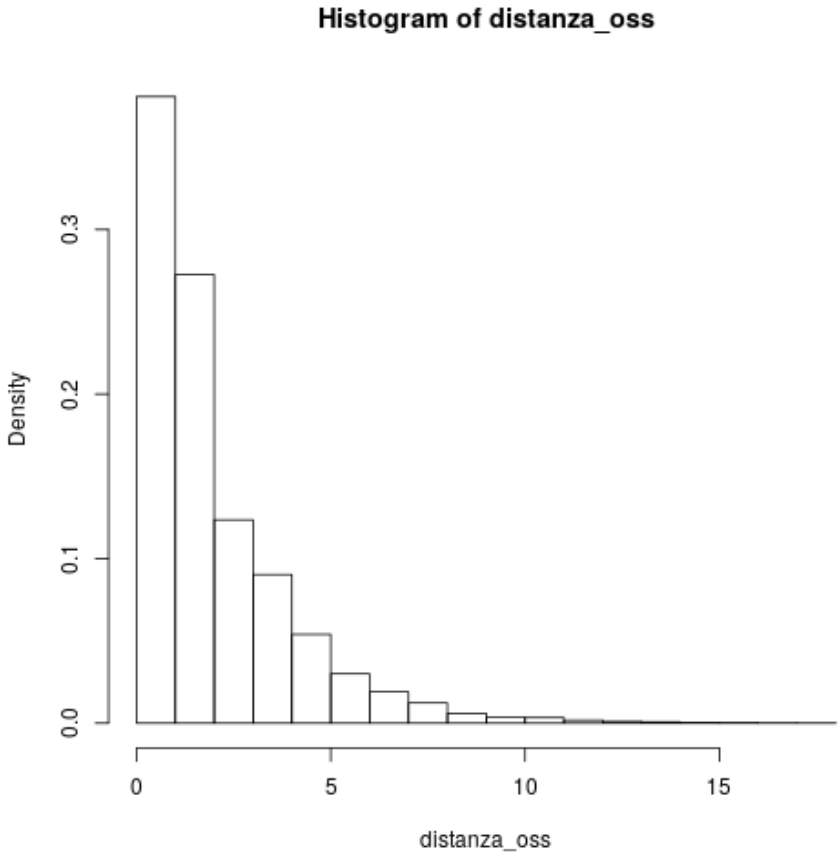


Figura 11.1: plot of chunk unnamed-chunk-3

R: Valori critici

calcoliamo i valori critici per $\alpha = 0.05$ e 0.01, ovvero il 95^o ed il 99^o percentile.

```
quantile(distanza_oss, probs = c(0.95,0.99))
```

```
##      95%      99%
## 5.884191 9.184926
```

Valori critici e inferenza

Questo calcolo ci permette di inferire che possiamo rifiutare l'ipotesi nulla quando la distanza della tabella osservata da quella attesa (misurata con la formula 11.1) è > 5.88 (con $\alpha = 0.05$) o > 9.47 (con $\alpha = 0.01$)

Possiamo dunque generare un nuovo campione, calcolare la distanza dalla distribuzione attesa, e confrontarla con i valori critici.

R: Generazione di un nuovo campione}

Generiamo un nuovo campione, calcoliamo la tabella di contingenza, e calcoliamo il χ^2. Poiché il campione è generato a partire dall'urna, ci aspettiamo che la statistica calcolata non sia significativa.

```
campione <- sample(atteso,100,replace=TRUE)
t_campione <- table(campione)
chi_quadro <- sum(((t_campione-t_atteso)^2)/t_atteso)
chi_quadro
```

```
## [1] 3.766544
```

Stima del p-value

Inoltre, possiamo stimare il p-value, ovvero la probabilità di compiere un errore di tipo I rifiutando l'ipotesi nulla. Per fare questo, basta aggiungere la distanza del nuovo campione al vettore delle 10.000 distanze, e calcolare la posizione della distanza rispetto a tutte le altre (usando la funzione rank()).

Calcolo della posizione

```
posizione <- rank (c(chi_quadro,distanza_oss))[1]
p_value <- 1-posizione/length(distanza_oss)
p_value
```

```
## [1] 0.145
```

Non rifiuto dell'ipotesi nulla

In questo caso, dunque, non rifiutiamo l'ipotesi nulla, in quanto

- il valore del χ^2 è inferiore al valore critico con $\alpha = 0.05$: 0.09007353xxxxxx < 5.88
- il p-value è pari a 0.94675xxxxxxxxxx, ovvero ben sopra ad $\alpha = 0.05$.

Calcolo su nuovo campione

Proviamo ora a fare la stessa verifica, ma questa volta partendo da un campione da noi generato, in cui le frequenze osservate sono pari a 22, 35, 43.

È facile intuire che questa distribuzione è molto diversa da quella attesa: 17 51 33.

Il calcolo inferenziale, però, ci permette una stima più precisa.

R: Calcolo su nuovo campione

```
t_campione2 <- c (22, 35,43)
chi_quadro <- sum(((t_campione2-t_atteso)^2)/t_atteso)
chi_quadro
```

```
## [1] 10.27145
```

```
posizione <- rank (c(chi_quadro,distanza_oss))[1]
p_value <- 1-posizione/length(distanza_oss)
p_value
```

```
## [1] 0.00745
```

Rifiuto dell'ipotesi nulla

In questo caso, dunque, **rifiutiamo** l'ipotesi nulla, in quanto

- il valore del χ^2 è superiore ad entrambi i valori critici: con $\alpha = 0.01$: 10.27 > 9.47
- conseguentemente, il p-value (pari a 0.00705), è inferiore ad $\alpha = 0.01$.

La distribuzione χ^2

Nella sezione precedente, abbiamo calcolato valori critici e p-value basandoci sulla distribuzione dell'errore di campionamento generata dalla simulazione.

La distribuzione che abbiamo ottenuto, applicando la formula del χ^2, è una distribuzione nota con il nome *distribuzione χ^2.

L'ambiente R mette a disposizione delle funzioni che, similmente alle distribuzioni normale e t di Student, permette di calcolare alcuni valori legati alla distribuzione χ^2.

χ^2: funzioni in R

Con `rchisq` è possibile generare dei valori casuali, con distribuzione χ^2. Con `dchisq` possiamo ottenere la densità della distribuzione per un determinato valore.

Così come per la distribuzione t di Student, anche la χ^2 è una famiglia di distribuzioni, che differiscono fra loro in base ai gradi di libertà (df, degree of freedom).

Pertanto, le funzioni legate al χ^2 si attendono, fra gli argomenti, anche i gradi di libertà.

Gradi di libertà

Nell'esempio precedente, la tabella delle distribuzioni aveva un rango pari a 3 (ovvero, avevamo 3 categorie: agricoltura, industria, terziario).

In una tabella unidimensionale, i gradi di libertà sono pari a r-1. Nel nostro caso, dunque df=2.

Nel prossimo grafico, visualizziamo nuovamente l'istogramma delle distribuzioni della distanza dai campioni generati alle frequenze attese (figura **??**). All'istogramma sovrapponiamo la distribuzione χ^2 con 2 gradi di libertà.

Sovrapposizione fra distribuzione osservata e teorica

```
plot_range <- seq(0,15,by=.25)
prob_dist <- dchisq(plot_range,2)
hist(distanza_oss,freq=FALSE,breaks=20)
lines(plot_range,prob_dist,type="l", col=2)
```

Utilizzo della distribuzione

Come vediamo, la distribuzione della nostra simulazione si sovrappone quasi perfettamente alla distribuzione χ^2. Appurata questa sovrapposizione, possiamo sfruttare la distribuzione χ^2 per calcolare valori critici e p-value.

Ad esempio, grazie alla funzione `pchisq` possiamo calcolare il p-value dei due campioni che abbiamo utilizzato negli esperimenti precedenti.

R: Calcolo del p-value usando pchisq

la funzione `pchisq`, analogamente a `pnorm`, ci permette di calcolare l'area della distribuzione *a destra* di un determinato valore. Con `1 - pchisq` calcoliamo l'area rimanente, che corrisponde al p-value.

```
1-pchisq(0.2034314, df=2) ## non capisco i valori
```

```
## [1] 0.9032863
```

```
1-pchisq(10.27145, df=2)
```

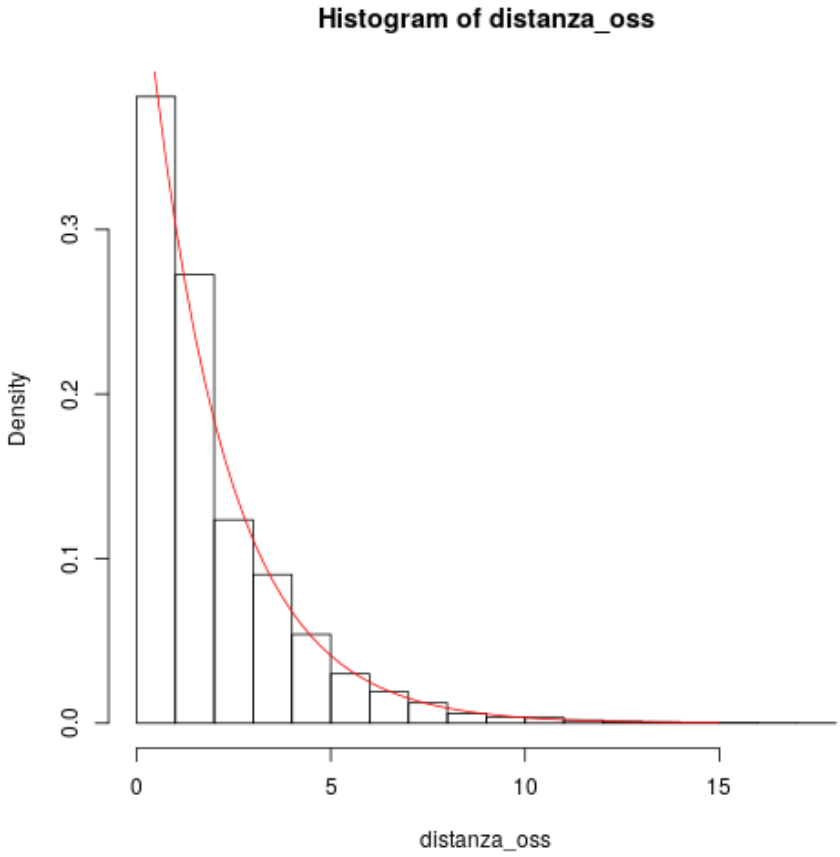


Figura 11.2: plot of chunk unnamed-chunk-8

```
## [1] 0.005882785
```

Approssimazione dei risultati

Come possiamo notare, i p-value non sono identici a quelli ottenuti partendo dalla simulazione, in quanto la sovrapposizione fra la distribuzione ottenuta dalla simulazione non è perfettamente identica alla distribuzione teorica.

Ciononostante, i valori sono molto simili, e ci portano a trarre le stesse conclusioni: non rifiuto di H_0 nel primo caso, rifiuto nel secondo.

La funzione `chisq.test`

Il nostro excursus ha, naturalmente, una finalità esclusivamente didattica. In pratica, per calcolare il test del χ^2, R ci mette a disposizione la funzione *chisq.test}, che ci restituisce il calcolo del χ^2, i gradi di libertà, e il p-value.

Applichiamo la funzione al primo gruppo.

```
chisq.test (x=t_campione,p=prob_attesa)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  t_campione
## X-squared = 3.7665, df = 2, p-value = 0.1521
```

La funzione ci restituisce il calcolo del χ^2, i gradi di libertà (df), il p-value.

chisq.test, secondo gruppo

Applichiamo la stessa funzione al secondo gruppo.

```
chisq.test (x=t_campione2,p=prob_attesa)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  t_campione2
## X-squared = 10.271, df = 2, p-value = 0.005883
```

Confronto fra due variabili nominali

Statistica bivariata

Nella sezione precedente abbiamo analizzato il caso di del confronto fra una distribuzione osservata ed una attesa.

La procedura, però, può essere utilizzata anche per valutare delle ipotesi relative al rapporto che intercorre fra due variabili nominali, ovvero nel contesto di una statistica bivariata.

Esempio: categoria lavorativa e genere

Per introdurre questa statistica, modifichiamo l'esempio precedente. Immaginiamo di voler capire se, nella distribuzione della forza lavoro fra agricoltura, industria e terziario, vi sono differenze di genere.

Per fare questo, raccogliamo un campione di persone attive, e per ognuno di loro identifichiamo il genere e la categoria lavorativa (agricoltura, industria, terziario).

Il processo

- A partire da questi dati, possiamo creare una tabella di contingenza a doppia entrata, di dimensione r * c, dove r è pari al rango della prima variabile, e c a quello della seconda.
- In secondo luogo, calcoliamo la tabella delle frequenze attese.
- Calcoliamo, attraverso la formula 11.1, la *distanza} fra le frequenze attese e quelle osservate.
- Calcoliamo il p-value, attraverso la funzione $1-pchisq(\chi^2, df = (r-1)(c-1))$.

Calcolare le frequenze attese

L'unica novità di rilievo, rispetto all'algoritmo relativo ad una sola variabile, è il calcolo delle frequenze attese è più complicato (ma non troppo)

Nuovamente, le frequenze attese si basano sull'ipotesi nulla, ovvero che non vi sia alcun *legame} fra le due variabili.

In termini di probabilità condizionale, si assume che la probabilità che un individuo appartenga ad una delle categorie della seconda variabile non cambi a seconda che l'individuo appartenga ad una categoria della prima, e viceversa.

L'ipotesi di indipendenza

Nel nostro esempio, l'ipotesi di indipendenza assume che il fatto di essere maschio (o femmina) non influisca sulla probabilità di essere occupato nell'agricoltura, nell'industria o nel terziario, e viceversa: il fatto di essere impiegato nel terziario non incide sulla probabilità di essere maschio o femmina.

In base a questa assunzione, la frequenza attesa, nella categoria agricoltura, maschio è pari alla probabilità associata alla categoria agricoltura, moltiplicata per la probabilità associata alla categoria maschio, moltiplicata per la numerosità del campione.

Frequenze attese

Continuiamo ad assumere probabilità pari a .17, .51 e .32 per la variabile tipo di occupazione, e assumiamo che la popolazione attiva sia per il 54% maschile e per il 46% femminile (di nuovo, sono percentuali inventate). Decidendo per un campione di 100 persone, la frequenza attesa, per la casella agricoltore maschio, sarà pari a .17 * .54 * 100 = 9.18

% *** controlla la sintassi

$$f_{e[i,j]} = \frac{f_i f_j}{n} \tag{11.3}$$

$$f_{e[i,j]} = p_i p_j n \tag{11.4}$$

R: Generare il data-frame

Creiamo un data.frame con due colonne: il genere e l'occupazione. In primo luogo creiamo un'urna, con probabilità .54 e .46, da cui estrarre il genere. Poi, usando l'urna precedente (relativa all'occupazione), creiamo un vettore di 100 osservazioni. Creiamo un secondo vettore, nuovamente di 100 osservazioni, relative al genere. Creiamo infine il data.frame con i due vettori, `campione_genere` e `campione_occupazione`.

```
genere_atteso <- c(rep(1,54),rep(2,46))
length(genere_atteso)
```

```
## [1] 100
```

```
t_genere_atteso <- table(genere_atteso)
prob_genere_attesa <- t_genere_atteso/length(genere_atteso)

campione_occupazione <- sample(atteso,100,replace=FALSE)
campione_genere <- sample(genere_atteso,100,replace=FALSE)
campione <- data.frame (campione_genere,campione_occupazione)
t_campione <- table(campione)
```

R: calcolo delle probabilità attese

Calcoliamo le probabilità attese. Calcoliamo la somma marginale delle righe e delle colonne. Calcoliamo le probabilità attese, pari a `(marginale riga/numerosità campione) * (marginale colonna/numerosità campione)`.

```
marginali_riga <- apply(t_campione,1,sum)
marginali_colonna <- apply(t_campione,2,sum)
prob_attese <-(marginali_riga/100) %*% t(marginali_colonna/100)
# prob_attese ?
```

Frequenze attese e osservate

Le frequenze attese sono pari alle probabilità attese, moltiplicate per il numero di osservazioni.

```
t_atteso <- prob_attese * 100
t_atteso
```

```
##         1     2     3
## [1,] 9.18 27.54 17.28
## [2,] 7.82 23.46 14.72
```

```
t_campione
```

```
##                 campione_occupazione
## campione_genere  1  2  3
##               1  9 27 18
##               2  8 24 14
```

R: calcolo di χ^2 e p-value

Calcoliamo il valore della statistica χ^2. Calcoliamo poi il `p-value`, utilizzando la funzione `pchisq`.

```
chi_quadro <- sum(((t_campione-t_atteso)^2)/t_atteso)
p_value <- 1-pchisq(chi_quadro, df=2)
chi_quadro
```

```
## [1] 0.09590793
```

```
p_value
```

```
## [1] 0.9531777
```

R: uso di chisq.test

Naturalmente, lo stesso calcolo può essere eseguito – più agevolmente – usando la funzione `chisq.test`.

```
chisq.test (t_campione)
```

```
##
##  Pearson's Chi-squared test
##
## data:  t_campione
## X-squared = 0.095908, df = 2, p-value = 0.9532
```

Leggere l'output

La funzione `chisq.test` ci restituisce il nome del test: *Pearson's Chi-squared test*; il valore della statistica: X-squared = 2.5602}; i gradi di libertà: `df = 2`; il `p-value = 0.278xxxxxxxxx`.

Non rifiuto dell'ipotesi nulla

Come prevedibile – considerata la modalità con cui abbiamo generato il campione – dal calcolo del χ^2 non possiamo rifiutare l'ipotesi nulla, in quanto $p - value = 0.278xxxxxx > \alpha = 0.05$.

Le frequenze osservate nel data frame generato scegliendo le due variabili in maniera indipendente non si discostano significativamente dalle frequenze attese.

Capitolo 12

T test: confronto fra medie di due campioni

T test: confronto fra medie di due campioni

Introduzione

Confronto fra due medie

Nel capitolo [ch:chiquadro], abbiamo introdotto il confronto fra variabili di tipo categoriale. In questo capitolo, affronteremo la statistica che permette di valutare la relazione fra una variabile di tipo categoriale ed una numerica (ad intervalli o rapporti). Nel caso specifico, ci limitiamo alla circostanza in cui la variabile indipendente, categoriale, ha due sole categorie.

In questa circostanza, le osservazioni sulla variabile dipendente, numerica, vengono divise in due insiemi. Il questito inferenziale che ci si pone è di valutare se i valori della variabile dipendente, di tipo numerico, differiscono significativamente da un gruppo all'altro.

Nel confronto fra due campioni si può adottare l'approccio parametrico, utilizzando il t-test, oppure un approccio non parametrico. In questo capitolo, verrà introdotto prima l'approccio non parametrico, in quanto più intuitivo, e dopo l'approccio parametrico. In entrambi i casi, l'argomento verrà affrontato prima con un approccio simulativo, e poi utilizzando la distribuzione teorica di riferimento. Infine, verrà utilizzata la corrispondente funzione di R e ne verranno letti i risultati.

Calcolo non parametrico

Approccio intuitivo

Intuitivamente, possiamo dire che i due gruppi differiscono se le misurazioni di un gruppo sono, *in genere*, sistematicamente più alte (o più basse) delle misurazioni sull'altro gruppo.

Se tutte le osservazioni di un gruppo (chiamiamolo gruppo B) sono più elevate di tutte le osservazioni dell'altro (gruppo A), possiamo inferire che, relativamente alla variabile misurata, vi è una differenza significativa fra il gruppo B ed il gruppo A.

Seguendo questa intuizione, un metodo per valutare se la variabile indipendente, categoriale, ha una relazione sulla variabile dipendente (numerica), è quello di confrontare ogni elemento del gruppo A con ogni elemento del gruppo B.

Confronto fra elementi

Sempre facendo ricorso all'intuizione, appare chiaro che se il numero di confronti vinti da un gruppo è molto superiore al numero di confronti vinti dall'altro, la differenza è significativa.

In questo conteggio, possiamo arrivare a due condizioni estreme: nella prima, gli elementi di A vincono tutti i confronti nei confronti di tutti gli elementi di B; nella seconda, sono gli elementi di B a vincere tutti i confronti.

L'ipotesi nulla, H_0, assume la parità fra il numero di confronti vinti da A e vinti da B.

La simulazione

Errore di campionamento

Sappiamo però che è molto improbabile che il confronto fra i due gruppi sia perfettamente pari. Come sappiamo, infatti, anche nella circostanza in cui i due campioni sono estratti dalla stessa popolazione ed assegnati ad una o all'altra categoria a caso, emergeranno delle differenze dovute all'errore di campionamento.

Generare la distribuzione dell'errore

Per stimare l'entità di questo errore, e misurare la probabilità che una differenza sia attribuibile o meno ad esso, possiamo usare la stessa metodologia vista nei capitoli precedenti:

- generare k coppie di campioni, estratte ed assegnate casualmente;
- calcolare il numero di confronti vinti dall'uno e dall'altro gruppo;
- salvare questi valori in un vettore, che costituisce la distribuzione dell'errore di campionamento.

Confrontare una coppia di campioni

Potendo disporre della distribuzione dell'errore di campionamento, data una coppia di campioni possiamo calcolare il numero di vittorie dell'uno e dell'altro gruppo, e valutare dove si collocano nella distribuzione.

Se il risultato di questi confronti si colloca sulle code della distribuzione, possiamo rifiutare l'ipotesi nulla ed accettare l'ipotesi alternativa, ovvero che vi è una differenza significativa fra i due gruppi.

In caso contrario, non si rifiuta l'ipotesi nulla.

R: genero la popolazione, due campioni

Generiamo una popolazione di 10.000 unità, con media 20, sd 2 e distribuzione normale, usando `rnorm`.

> n <- 10000 > m <- 100 > k <- 10000 > mediaTeorica <- 20 > sdTeorica <- 2 > popolazione <- rnorm(n, mediaTeorica, sdTeorica)

```
n <- 10000 # numerosità della popolazione
m <- 100   # numerosità di ognuno dei 2 campioni
k <- 10000   # numero di coppie di campioni generati

media_teorica <- 20 # arbitrario
sd_teorica <- 2 # arbitrario

popolazione <- rnorm (n, media_teorica, sd_teorica)
```

R: calcoliamo i confronti fra i due campioni

Estraiamo ora 2 campioni dalla popolazione. Instanziamo due contatori, `sumA` e `sumB`. Con un ciclo for annidato, confrontiamo ogni valore del campione1 con ogni valore del campione2 [1]. Incrementiamo il contatore `sumA` quando a vincere è l'unità del campione1, incrementiamo `sumB` quando vince campione2.

```
campione1 <- sample(popolazione,m,replace=FALSE)
campione2 <- sample(popolazione,m,replace=FALSE)
sumA <- 0; sumB <- 0
for (x in 1:m) {
    oss1 <- campione1[x]
    for (y in 1:m) {
        oss2 <- campione2[y]
        colore <- 2
        if (oss2>oss1) {
            sumA <- sumA + 1
        } else {
            sumB <- sumB + 1
        }
    }
}
```

R: risultati

Non dovrebbe sorprenderci il fatto che la somma dei due valori è pari a m*m, ovvero il numero dei confronti.

```
c(sumA, sumB)
```

```
## [1] 4566 5434
```

Questa statistica viene chiamata *Mann-Whitney-Wilcoxon U*

Calcolo della posizione ordinale

È però possibile ottenere lo stesso risultato con un calcolo diverso: l'insieme di tutte le osservazioni viene ordinato, e ad ogni osservazione viene assegnato un punteggio pari alla sua posizione;

- per ognuno dei due gruppi, si somma il punteggio di ogni osservazione;
- a questi valori, si sottrae quello che è il minimo valore possibile, ovvero m*(m+1)/2.

Il vantaggio di questo algoritmo è che può essere esteso a confronti fra più di due gruppi.

R: il calcolo del ranking

```
due_rank <- matrix(rank(c(campione1,campione2)),nrow=2,ncol=m, byrow=TRUE)
due_somma_rank <- apply (due_rank,1,sum)
```

[1] Per semplicità, ssumiamo che non vi siano *pareggi* fra i confronti.

```
rank_atteso <- m*(m+1)/2
wilcoxon <- due_somma_rank-rank_atteso
wilcoxon
```

```
## [1] 5433.5 4566.5
```

La simulazione

Introdotta la statistica, usiamo la simulazione. Generiamo 2 vettori. Nel primo, `distribuzione`, inseriremo le coppie di valori che calcoleremo usando la statistica di Wilcoxon. Nel secondo, `differenze`, salviamo una seconda statistica: la differenza delle medie fra i due campioni. La prima distribuzione ci serve in questa sezione, la seconda nella sezione dedicata al calcolo parametrico.

```
distribuzione <- vector(mode = "numeric", length = 20000)
differenze <-    vector(mode = "numeric", length = length(distribuzione)/2)
```

A questo punto, usando un ciclo for, possiamo generare k=10000 coppie di campioni, calcolare per ognuno le due statistiche, e salvarla nei due vettori.

R: la generazione delle coppie di campioni

```
for (i in 1:length(distribuzione)/2) {
    due_campioni <- matrix(sample(popolazione,2*m,replace=FALSE),nrow=2,ncol=m, byr
    campione1 <- due_campioni[1,]
    campione2 <- due_campioni[2,]
    due_rank <- matrix(rank(c(campione1,campione2)),nrow=2,ncol=m, byrow=TRUE)
    due_somma_rank <- apply (due_rank,1,sum)
    rank_atteso <- m*(m+1)/2
    wilcoxon <- due_somma_rank-rank_atteso
    distribuzione[i*2-1] <- wilcoxon[1]
    distribuzione[i*2] <- wilcoxon[2]
    differenze[i]<-mean(campione1)-mean(campione2)
}
```

R: la distribuzione U

```
par(mfrow=c(1,2))
hist(distribuzione)
qqnorm(distribuzione)
qqline(distribuzione)
```

R: valori critici

Possiamo calcolare i valori critici, ad esempio per α = 0.01 e 0.05 bidirezionale

```
quantile(distribuzione, probs = c(0.005,0.025,0.975,0.995))
```

```
##     0.5%     2.5%    97.5%    99.5%
## 3936.995 4194.975 5810.000 6059.000
```

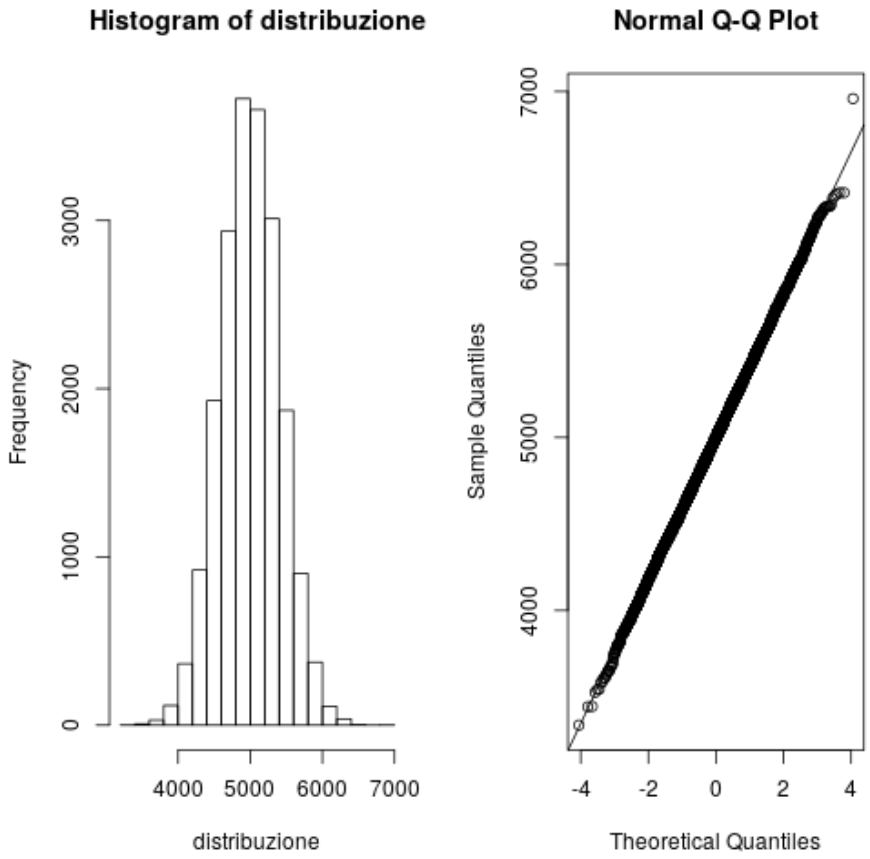


Figura 12.1: plot of chunk ttest_plot_distribuzione

Calcolare il p-value dalla distribuzione

Ora, generiamo due nuovi campioni, calcoliamo la statistica U, e vediamo dove si colloca rispetto alla distribuzione.

```
due_campioni <- matrix(sample(popolazione,2*m,replace=FALSE),nrow=2,ncol=m, byrow=T
campione1 <- due_campioni[1,]
campione2 <- due_campioni[2,]
due_rank <- matrix(rank(c(campione1,campione2)),nrow=2,ncol=m, byrow=TRUE)
due_somma_rank <- apply (due_rank,1,sum)
rank_atteso <- m*(m+1)/2
```

Calcolare il p-value dalla distribuzione

```
(wilcoxon <- due_somma_rank-rank_atteso)
```

```
## [1] 3984 6016
```

```
(p_value_simulazione <- rank(c(wilcoxon,distribuzione))[1:2]/length(distribuzione))
```

```
## [1] 0.007150 0.993425
```

La distribuzione U Mann-Whitney-Wilcoxon

La distribuzione di errore che abbiamo ottenuto grazie al confronto di 10000 coppie di campioni, è nota come distribuzione U Mann-Whitney-Wilcoxon [@Whitley-Ball:2002].

R mette a disposizione il gruppo di funzioni per calcolare la densità, la probabilità, per generare dei numeri secondo la distribuzione. Inoltre, mette a disposizione la funzione wilcox.test per calcolare, automaticamente, la statistica ed il p-value.

R: le funzioni per la distribuzione U

R mette a disposizione le consuete funzioni per calcolare densità, probabilità, generare numeri casuali e calcolare il test.

- dwilcox(x, m, n) calcola la densità di x * pwilcox(q, m, n) calcola la probabilità * rwilcox(nn, m, n) genera nn numeri casuali. * wilcox.test(gruppoA,gruppoB) calcola il test corrispondente

m e n sono la numerosità del primo e del secondo campione (che, nel nostro esempio, sono uguali – m)

R: calcolo del p-value con pwilcox

Utilizzando `pwilcox` calcoliamo i due p-value. Il valore interessante è quello più basso. Se, come nell'esempio, l'ipotesi è a due vie, dobbiamo raddoppiare il p-value

```
(p_value_Wilcoxon <-pwilcox(wilcoxon, 100,100))
```

```
## [1] 0.006436703 0.993607899
```

```
# se il test è a due code, il p-value va moltiplicato per 2
p_value_simulazione*2
```

```
## [1] 0.01430 1.98685
```

```
p_value_Wilcoxon*2
```

```
## [1] 0.01287341 1.98721580
```

```
wilcox.test(campione1,campione2,exact = TRUE)
```

```
##
##  Wilcoxon rank sum test
##
## data:  campione1 and campione2
## W = 3984, p-value = 0.01287
## alternative hypothesis: true location shift is not equal to 0
```

Come sempre, i risultati della distribuzione generata non sono uguali, ma simili, a quelli della distribuzione teorica.

R: uso di wilcox.test

Vediamo ora il calcolo effettuando la funzione di R `wilcox.test`

```
wilcox.test(campione1,campione2,exact = TRUE)
```

```
##
##  Wilcoxon rank sum test
##
## data:  campione1 and campione2
## W = 3984, p-value = 0.01287
## alternative hypothesis: true location shift is not equal to 0
```

La funzione calcola la statistica, xxx, e il p-value.

Approccio parametrico

La differenza delle medie

L'algoritmo utilizzato nelle sezioni precedenti costituisce l'approccio non parametrico al confronto fra due campioni.

Come abbiamo visto, il calcolo non parametrico non prende in considerazione né la media né la deviazione standard e nemmeno la distribuzione dei campioni.

Il vantaggio del calcolo non parametrico è che fa pochissime assunzioni:

- le m+n osservazioni devono essere indipendenti;
- m e n devono avere una numerosità di almeno 5 elementi.

Assunzioni

Per applicare il test parametrico, al contrario, è necessario non solo che le osservazioni siano indipendenti (e la numerosità adeguata). È altresì necesario che:

- la distribuzione dei due campioni sia normale;
- la varianza dei due gruppi non sia diversa.

R permette il calcolo del t test anche in caso di varianze differenti (attraverso l'approssimazione di Welch).

La differenza fra le medie

Il test parametrico si basa sulla differenza fra le medie dei due campioni. Questo spiega l'assunto di normalità dei campioni.

In pratica, il test calcola il valore assoluto della differenza fra le due medie, e la confronta con la distribuzione t di Student.

Naturalmente, anche in questo caso possiamo calcolare il p-value ignorando la distribuzione teorica, ma basandoci sulla distribuzione dell'errore di campionamento generata dal confronto delle nostre 10.000 coppie di campioni.

Nel ciclo for che usammo per generare la statistica U, popolammo anche un vettore, differenze, con il codice `differenze[i]<-mean(campione1)-mean(campione2)`. Possiamo ora usare quel vettore di distribuzioni dell'errore.

R: p-value usando la distribuzione

Calcoliamo il p-value confrontando la distanza delle due medie con la distribuzione dell'errore.

```
distanza<-abs(mean(campione1)-mean(campione2))
(p_value_differenze_simulazione <- 1-rank(c(distanza,differenze))[1]/length(differe
```

```
## [1] 0.0067
```

```
# se test a due code, va raddoppiato
p_value_differenze_simulazione*2
```

```
## [1] 0.0134
```

Distribuzione dell'errore

Usiamo la funzione `density` per visualizzare la distribuzione dell'errore, e `qqnorm-qqline` per testarne la normalità.

```
par(mfrow=c(1,2))
plot(density(differenze))
qqnorm(differenze)
qqline(differenze,col="red")
```

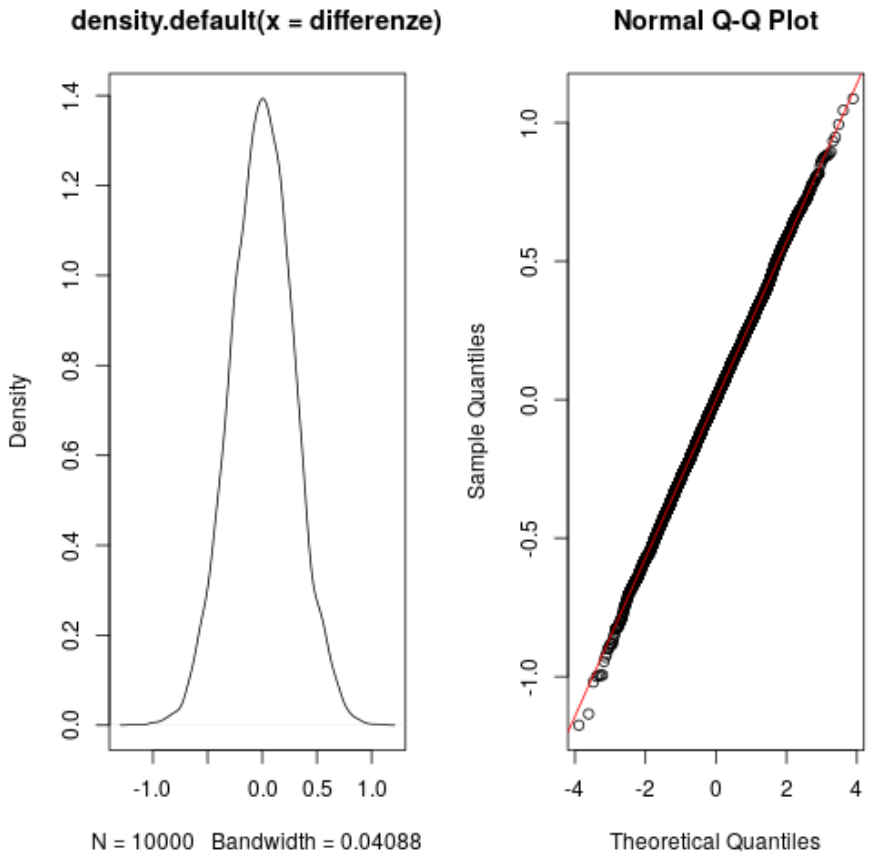


Figura 12.2: plot of chunk ttest_plot_differenze

Varianza dell'errore

Possiamo osservare dal grafico qqnorm che la distribuzione approssima la distribuzione normale.

In realtà, la varianza della distribuzione dell'errore è legata alla numerosità dei due campioni. Più precisamente, la varianza stimata è pari a

$$S^2_{\bar{x_1}-\bar{x_2}} = \frac{s_1^2(m-1)+s_2^2(n-1)}{m+n-2}(\frac{1}{m}+\frac{1}{n})$$

Dunque, l'errore standard della differenza fra le medie può essere calcolato, con R, usando

```
errore_standard <- sqrt((var(campione1)*(m-1)+var(campione2)*(m-1))/(m-1+m-1)*(1/m+
```

Calcolo di t

Calcoliamo t, e calcoliamo il p-value (che raddoppiamo, se l'ipotesi è bidirezionale).

```
t <- distanza/errore_standard
(p_value_differenze_t <- (1 - pt(t, df = (m-1+m-1))))
```

```
## [1] 0.006525759
```

```
# se test a due code, va raddoppiato
p_value_differenze_t*2
```

```
## [1] 0.01305152
```

Uso della funzione `t.test`

Infine, utilizziamo la funzione `t.test`

```
t.test(campione1,campione2)
```

```
##
##  Welch Two Sample t-test
##
## data:  campione1 and campione2
## t = -2.505, df = 197.32, p-value = 0.01305
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.2580755 -0.1497596
## sample estimates:
## mean of x mean of y
##  19.73332  20.43724
```

Leggere l'output

La funzione `t.test(campione1, campione2)` restituisce `xxx`, i gradi di libertà stimati `xxx`, il `p-value = xxx`, e l'intervallo di confidenza della differenza fra le medie: `xxx`. Quando i due termini dell'intervallo di confidenza hanno segni opposti, la differenza non è significativa.

Conclusioni

Capitolo 13

Correlazione e regressione lineare

Correlazione e regressione lineare

Introduzione

Confronto fra variabili quantitative

Dopo aver visto il confronto fra due variabili categoriali ed il confronto fra una variabile categoriale ed una variabile a intervalli (limitatamente al caso del confronto fra due soli gruppi), analizziamo ora il confronto fra due variabili quantitative.

Anche in questo caso, la statistica inferenziale si propone di valutare se esiste una relazione fra le variabili, e se questa relazione è significativa.

Andamento congiunto

Nei casi visti in precedenza, nei capitoli [ch:chiquadro] e [ch:t_test], ci si chiedeva se l'appartenenza di una osservazione ad una categoria od un'altra della variabile A influiva sulla distribuzione della variabile B.

Sia nel caso del test del χ^2 che del test t di Student, che (come vedremo) nell'analisi della varianza, ci si concentra sulla significatività delle differenze.

Nel caso della relazione fra due variabili numeriche, invece, ci si chiede se le due variabili si *muovono assieme*: se al crescere di una cresce anche l'altra (correlazione positiva), o se al crescere di una l'altra cala (correlazione negativa), e ci si chiede se questo *andamento congiunto* sia significativo o sia dovuto al caso.

Ipotesi nulla e ipotesi *estrema*

Anche in questo caso, è opportuno partire dall'ipotesi nulla, ovvero dall'ipotesi che non vi sia alcun legame fra le due variabili.

Anche in questo caso, è necessario identificare una statistica, ovvero una misura dell'aspetto rilevante.

Da un punto di vista didattico, però, può essere utile focalizzarci sulla situazione estrema di una totale correlazione fra le due variabili.

Nell'esempio che segue, ci limiteremo ad analizzare la circostanza di una correlazione positiva, ma il principio può essere generalizzato alle correlazioni negative.

Esempio: misurare le precipitazioni

Immaginiamo che un osservatorio metereologico debba misurare le precipitazioni atmosferiche (pioggia) nel corso dell'anno. Per farlo, viene raccolto in un bacino di un metro quadrato l'acqua piovana, e ad ogni pioggia l'acqua raccolta viene misurata.

Immaginiamo che il responsabile della misurazione sia molto pignolo, e che decida di misurare sia il volume dell'acqua in litri, che il suo peso in kilogrammi.

Rapporto fra litri e kg

```
x <- abs(rnorm (50,8,5))+0.5
plot(x,x,xlab="litri",ylab="kg", main="Correlazione perfetta")
```

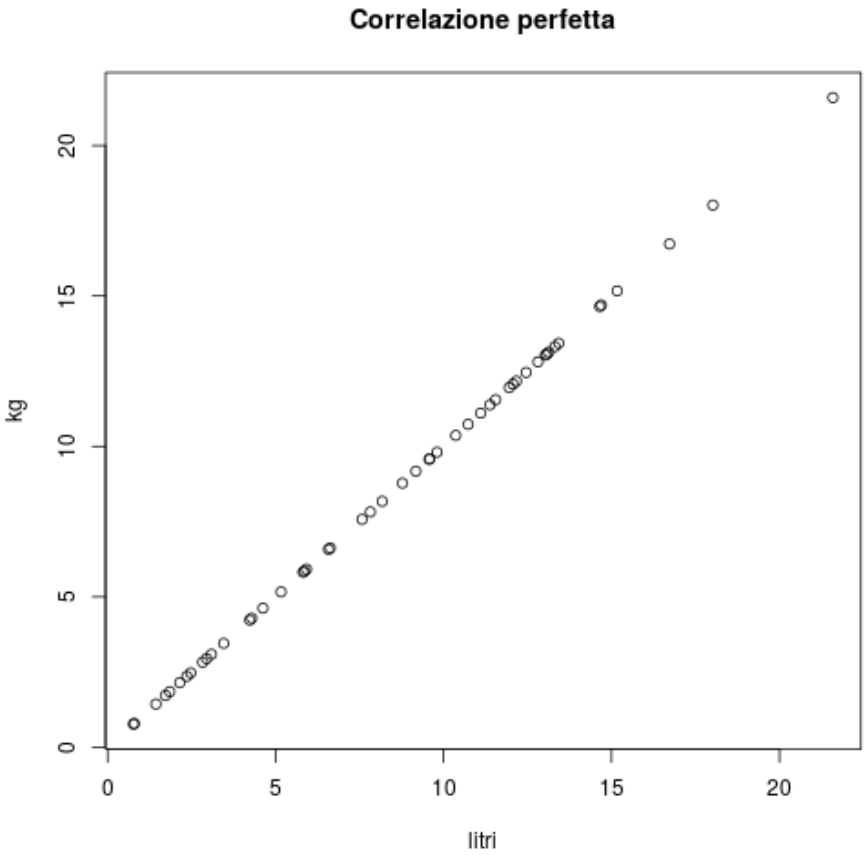


Figura 13.1: plot of chunk unnamed-chunk-1

La linea retta

L'aspetto più saliente del grafico [fig:corrlitrikg] è che le misure relative a peso e a volume si dispongono lungo una linea retta.

Il secondo aspetto è che, grazie alla linea, conoscendo il valore in litri, possiamo dedurre il peso in kg, e viceversa.

Diverse distribuzioni di esempio

```
sequenza <- seq(0,100, by=20)
k <- length(sequenza)
ycoeffs <- vector(mode = "numeric", length = k)
```

```
xcoeffs <- vector(mode = "numeric", length = k)
estimates <- vector(mode = "numeric", length = k)

x1 <- vector(mode = "numeric", length = 100)
y1 <- vector(mode = "numeric", length = 100)
conta <- 1
par(mfrow = c(2,3))
for (b in sequenza) {
    a <- 100-b
    for (i in 1:100) {
        uni1 <- runif (100,0,1)
        x1[i] <- sum (uni1)
        if (b>0 && b<100) {
            uni2 <- runif (b,0,1)
            y1[i] <- sum (uni1[1:a],uni2)
        }
        if (b==100) {
            uni2 <- runif (b,0,1)
            y1[i] <- sum (uni2)
        }
        if (b==0) {
            y1[i] <- sum (uni1)
        }

    }

#   x1 <- (x1-mean(x1))/sd(x1)
#   y1 <- (y1-mean(y1))/sd(y1)

    linmod <- lm (y1~x1)
    ycoeff <- linmod$coefficients[2]
    xcoeff <- lm (x1~y1)$coefficients[2]
    cortest <- cor.test (x1,y1)
    estimate <- cortest$estimate
    pvalue <- cortest$p.value
    ycoeffs[conta] <- ycoeff
    xcoeffs[conta] <- xcoeff
    estimates [conta] <- estimate
    conta <- conta + 1
    testo <-paste("corr.=",round(estimate,3),"p=",round(pvalue,3)) # "a=",
    testo
    plot (x1,y1, main=testo)
    abline(linmod)
    abline(h=mean(y1))
    abline(v=mean(x1))
}
```

Più realistica la simulazione della figura [fig:corrvarie]. In questo caso, la posizione dei punti sull'asse x è dato dalla somma di 100 valori casuali, generati su una distribuzione uniforme. Nel primo grafico, in alto a sinistra, i valori sull'asse y sono dati dalla

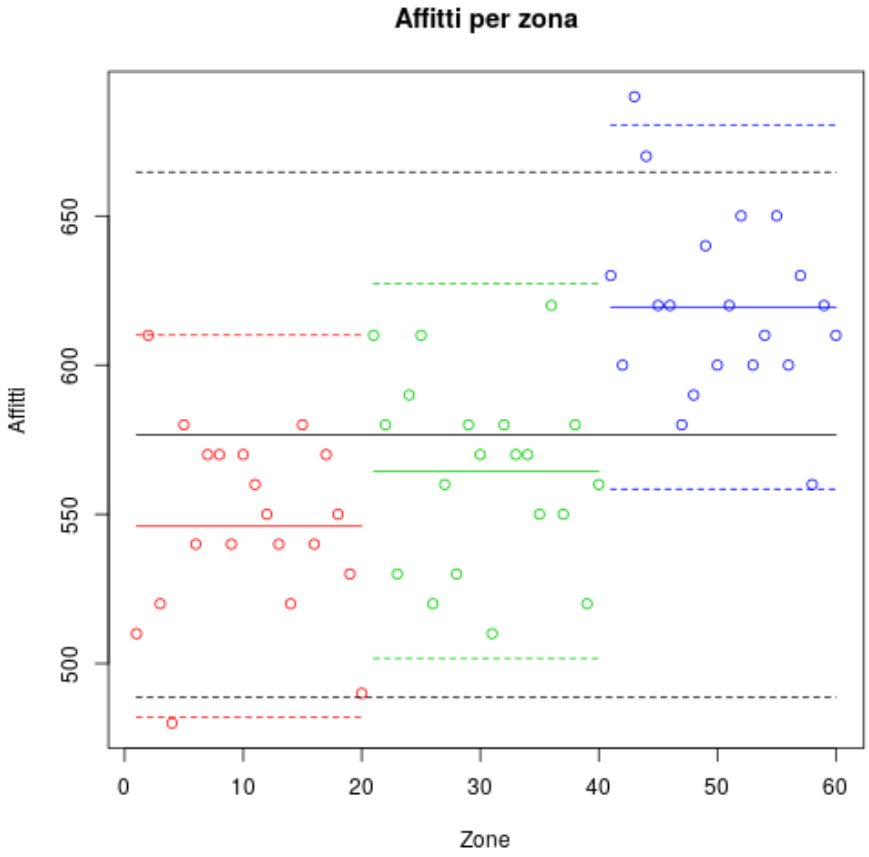


Figura 13.2: plot of chunk unnamed-chunk-2

somma degli stessi 100 valori, e dunque i punti si collocano perfettamente sulla linea di regressione.

Nel secondo grafico, i valori y sono il risultato di 80 dei valori casuali di x, e 20 valori casuali diversi. Negli altri grafici, la proporzione è di 60-40, 40-60 e così via. Nell'ultimo grafico, tutti e 100 i valori di y sono indipendenti, e dunque non vi è alcuna correlazione fra le due variabili.

La retta di regressione

In questi grafici abbiamo implicitamente introdotto la retta di regressione, che verrà discussa nelle prossime slides. Per ora, ci basti sapere che la retta di regressione ci permette di fare una stima del valore di y conoscendo il valore di x.

Nel primo grafico, proprio come nell'esempio precedente, conoscere il valore di x ci permette di inferire perfettamente il valore di y, in quanto le osservazioni di y cadono perfettamente sulla retta.

Già nel secondo grafico, però, questa previsione non è più perfetta: conoscendo x, possiamo soltanto stimare il valore di y.

Mano a mano che il legame diventa meno importante, la pendenza della linea di regressione diminuisce, fino a quasi sovrapporsi alla linea della media di y nell'ultimo grafico, dove le due variabili sono indipendenti.

Ciò che questo andamento ci dice è che conoscere il valore di x contribuisce sempre meno alla nostra conoscenza di y.

Dai grafici della seconda figura possiamo notare che la linea di regressione si incrocia, in tutti i casi, nella linea della media di x e nella linea della media di y nello stesso punto.

Questo significa che il valore stimato di y quando x è pari a $\bar{x}$ è $\bar{y}$.

Analisi grafica

Come abbiamo visto nelle analisi univariate e nelle altre statistiche bivariate, la visualizzazione dei dati e l'analisi visiva, qualitativa delle distribuzioni è parte integrante del processo descrittivo e inferenziale.

Il grafico utilizzato nella visualizzazione di due variabili quantitative è il grafico di dispersione, *scatterplot*. In R, si ottiene usando la funzione `plot (x,y)`.

Grafico di dispersione: cosa guardare

Quali sono gli aspetti più salienti che bisogna osservare in un grafico a dispersione?

- La forma generale della distribuzione.
- La direzione dell'associazione: è positiva o negativa?
- La forma della la relazione? È una linea retta oppure no? È una curva?
- La forza dell'associazione: i punti osservati sono vicini o lontani dalla linea di regressione?

- La presenza di outliers: vi sono osservazioni molto lontane dalla retta di regressione? È possibile che queste osservazioni insolite siano dovute ad errori?
- È ipotizzabile che l'andamento del grafico lasci intendere l'influenza di una variabile terza?

Analisi inferenziale

Coefficiente di correlazione

La correlazione è una relazione lineare fra due variabili a intervallo o a rapporti.

È importante sottolineare che la correlazione non distingue fra variabile indipendente e dipendente, e tratta le due variabili simmetricamente: la correlazione fra Y e X equivale alla correlazione fra X e Y.

L'analisi inferenziale è finalizzata a calcolare se esiste una relazione fra le due variabili, e se la relazione è statisticamente significativa.

Correlazione: cautele

Prima di applicare la statistica, è necessario tener conto di alcuni possibili problemi:

- La correlazione misura relazioni di tipo lineare. Se la relazione non è di tipo lineare, la correlazione non è appropriata.
- Soprattutto se l'insieme di osservazioni ha una bassa numerosità, è possibile che degli outliers condizionino fortemente il risultato.
- Se una terza variabile, anche categoriale, ha una influenza significativa su una o entrambe le variabili misurate, è possibile che, non tenendone conto, si calcolino delle correlazioni non appropriate.

Correlazione e causazione

La correlazione non implica causazione. La correlazione, infatti, può essere attribuibile a

- Causazione diretta: A causa B.
- Causa comune: C causa sia A che B.
- Fattore confounding: l'andamento della variabile dipendente può essere condizionato da un fattore esterno che non ha nulla a che fare con la variabile indipendente.
- Semplice coincidenza.

Requisiti per inferire causazione

Affinché si possa inferire causazione è necessario che:

- L'associazione sia abbastanza forte.

- Vi sia la possibilità di manipolare la variabile indipendente, e che il valore della variabile dipendente cambi di conseguenza.
- Vi sia un chiaro rapporto temporale: la causa deve precedere l'effetto.
- Le misure devono essere consistenti, e dunque replicabili.
- I risultati siano teoreticamente plausibili e coerenti con altre evidenze empiriche.
- L'associazione sia specifica (e dunque non possa essere attribuita a cause comuni o altri confounding).

Modelli Lineari Generalizzati

Sia l'analisi della varianza che la regressione lineare sono casi particolari della metodologia nota come Modelli Lineari Generalizzati.

La differenza più importante fra la regressione e l'ANOVA è che in un caso la variabile indipendente è a intervalli, nell'altro categoriale. Nell'ANOVA, dunque, non si fanno assuzioni sulla linearità della relazione.

Approccio intuitivo

Per misurare la correlazione lineare, abbiamo bisogno di una statistica che abbia alcune caratteristiche:

- sia pari a 0 in assenza di correlazione
- sia positiva quando la correlazione è positiva, e negativa quando è negativa
- che abbia un valore assoluto massimo, che identifica la circostanza in cui la correlazione è perfetta
- che sia standardizzata, ovvero che non dipenda dai valori assoluti delle variabili.

In termini formali:

$$-1 \leq r \leq 1$$

Linea di regressione e medie

Come abbiamo osservato, la linea di regressione si incrocia sempre con le due medie. Usando le linee che identificano le medie di x e di y possiamo dividere il grafico di dispersione in 4 quadranti. Se le osservazioni si distribuiscono principalmente nel quadrante in alto a destra e in quello in basso a sinistra, la correlazione è positiva. Se sono più frequenti nei quadranti in alto a destra o in basso a sinistra, la correlazione è negativa.

Per ottenere una misura standardizzata del rapporto fra le due variabili, possiamo decidere di trasformare sia la variabile x che la y in punteggi zeta.

Trasformazione in punti z

Con la trasformazione, otteniamo due variabili con media pari a zero e deviazione standard pari ad uno. La retta di regressione, a questo punto, incrocia le due medie nella posizione 0,0 del grafico.

- Il quadrante in basso a sinistra raccoglie le osservazioni in cui sia x che y sono negative.
- Il quadrante in alto a destra raccoglie le osservazioni in cui sia x che y sono positive.
- Il quadrante in basso a destra raccoglie le osservazioni in cui x è positiva e y è negativa.
- Il quadrante in alto a sinistra raccoglie le osservazioni in cui y è positiva e x è negativa.

A questo punto appare evidente che, moltiplicando x per y, otterrò valori positivi nei due quadranti alto sinistra e basso destra, e valori negativi nei quadranti basso destra, alto sinistra.

Cosa succede se sommo la moltiplicazione x * y di tutte le osservazioni, e divido per il numero di osservazioni (per la precisione, per n-1)? Otterrò un valore che sarà positivo in caso di correlazione positiva, negativo in caso di correlazione negativa, e sarà prossimo allo zero in caso di assenza di correlazione.

Inoltre, vedremo che il valore più alto che questa misura può raggiungere è pari ad 1, e di conseguenza il valore più basso è pari a -1.

Correlazione lineare: la formula

$$r = \frac{1}{n-1}\sum_{i=1}^{n}[(\frac{x_i - \bar{x}}{\sigma_x})(\frac{y_i - \bar{y}}{\sigma_y})]$$

Ricordando che $\bar{x}$ e $\bar{y}$ sono le medie di X e Y e che σ_X e σ_Y sono le deviazioni standard di X e Y, e ricordando che il calcolo del punteggio z è pari a

$$z = \frac{x_i - \bar{x}}{\sigma_x}$$

possiamo riscrivere r come

$$r = \frac{1}{n-1}\sum_{i=1}^{n}(z(x_i)z(x_i))$$

Correlazione lineare: assunti

- Poiché il calcolo della correlazione utilizza la trasformazione dei punteggi grezzi in punti zeta, si assume che entrambe le variabili siano almeno a livello di scala di intervallo e abbiano una distribuzione normale.

- Le osservazioni su entrambe le variabili devono essere stocasticamente indipendenti (ovvero, il valore di di una osservazione non deve condizionare il valore di un'altra osservazione)
- Infine, il rapporto fra le due variabili sia di tipo lineare. Vedremo nelle prossime sezioni quali alternative esistono in caso di non linearità del rapporto.

Distribuzione dell'errore

Come consuetudine, utilizziamo la consueta sequenza logica

- identificazione di una misura della relazione
- identificazione di una popolazione virtuale
- estrazione casuale di k campioni
- calcolo del vettore delle k misure
- osservazione della distribuzione delle misure
- calcolo del p-value attraverso il confronto con il vettore delle misure
- identificazione di una distribuzione teorica che, previo opportuna trasformaizone, mappa quella osservata
- calcolo del p-value utilizzando la probabilità della distribuzione teorica identificata
- utilizzo della funzione di R

La simulazione

- La misura della relazione è il coefficiente r identificato sopra.
- Attraverso la funzione `rnorm ()` possiamo estrarre n campioni di osservazioni casuali da una popolazione con specifica media e deviazione standard. Poiché siamo interessati a misurare la relazione fra due variabili, per ogni misura estraiamo due campioni
- Utilizziamo la funzione `scale()` per trasformare le osservazioni in punti z.
- Calcoliamo la statistica r, e la salviamo nel vettore delle misure.

R: la simulazione

```
m <- 100   # numerosità dei campioni
k <- 10000   # numero di campioni generati
relazioni <- vector(mode = "numeric", length = k)
for (i in 1:length(relazioni)) {
    x1 <- rnorm (m,20,2)
    x2 <- rnorm (m,50,6)
    x1 <- scale(x1)
    x2 <- scale (x2)
    erre <- sum(x1*x2)/(length(x1)-1)
```

```
    relazioni[i] <- erre
}
```

Grafico della distribuzione dell'errore

Visualizziamo la distribuzione dell'errore di r nella figura

```
hist (relazioni)
```

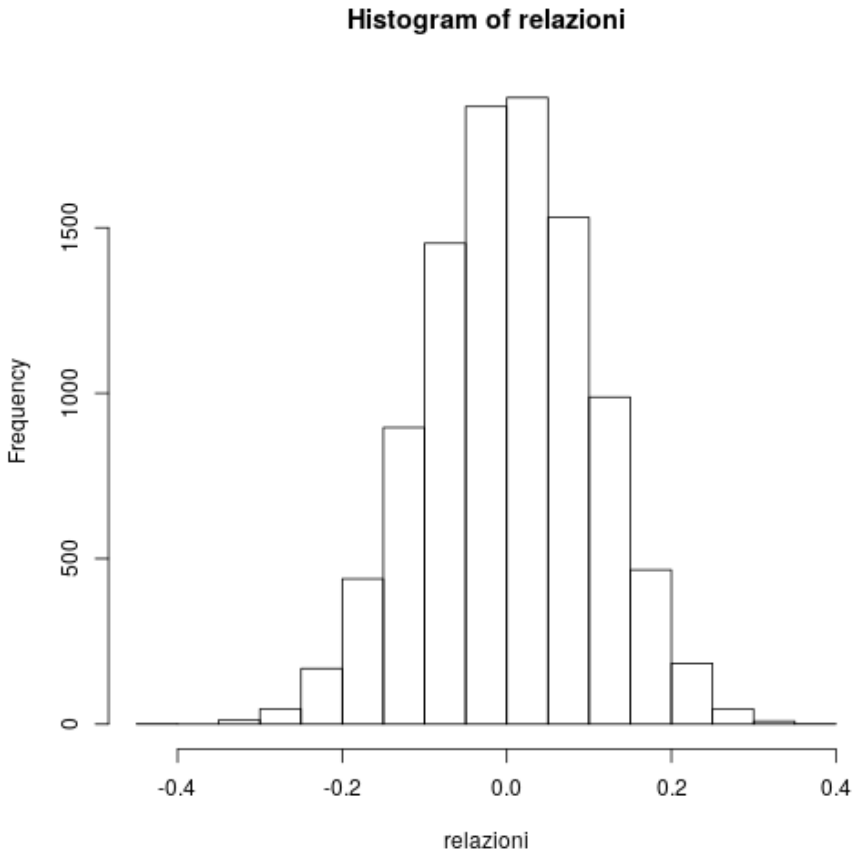


Figura 13.3: plot of chunk corr_hist_relazioni

Rapporto fra distribuzione dell'errore e t

Poiché il valore possibilie di r varia nel range $-1 \leq r \leq 1$, anche la distribuzione dell'errore varia nello stesso range, e si concentra attorno ai valori -0.1, 0.1.

La forma della distribuzione approssima la t di Student, previa opportuna trasformazione. Per arrivare alla distribuzione t va applicata la trasformazione

$$t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}}$$

P-value calcolato sulla distribuzione t

Per calcolare il p-value basandoci sulla distribuzione t, dovremmo dunque trasformare r in t, e poi calcolare la probabilità di t.

Alcuni esempi

Facciamo ora alcuni esempi, generando differenti casi di variabili bivariate.

Primo esempio: variabili indipendenti

In questo caso, generiamo due variabili casuali indipendenti. Ci aspettiamo un *r* prossimo allo 0.

```
x1 <- rnorm (100,20,2)
y1 <- rnorm (100,50,3)
sx1 <- scale(x1)
sy1 <- scale (y1)
erre1 <- sum(sx1*sy1)/(length(sx1)-1)
erre1
```

```
## [1] -0.06094029
```

La statistica *r*, dunque, è pari a -0.0609403.

```
colori <- c(2,2,1)
colore <- colori[sign(sx1*sy1)+2]
plot (sx1,sy1,col=colore)
abline (a=0,b=erre1)
abline (v=0);
abline(h=0)
```

Calcolo p-value su simulazione

Calcoliamo il p-value usando la distribuzione osservata. Poiché la nostra ipotesi è a due code, moltiplichiamo p per 2. Poiché le due variabili sono indipendenti, la nostra previsione è che p sia superiore a 0.05.

```
p_value_simulazione1 <- rank (c(-abs(erre1),relazioni))[1]/(length(relazio
p_value_simulazione1 * 2
```

```
## [1] 0.5329467
```

Il p-value è dunque pari a 0.5329467.

Secondo esempio: variabili correlate

In questo secondo esempio, la variabile y è creata in modo da correlare con x. Ci aspettiamo un *r* relativamente alto, e un p basso.

```
y2 <- x1 + rnorm (100,0,2)
sy2 <- scale (y2)
```

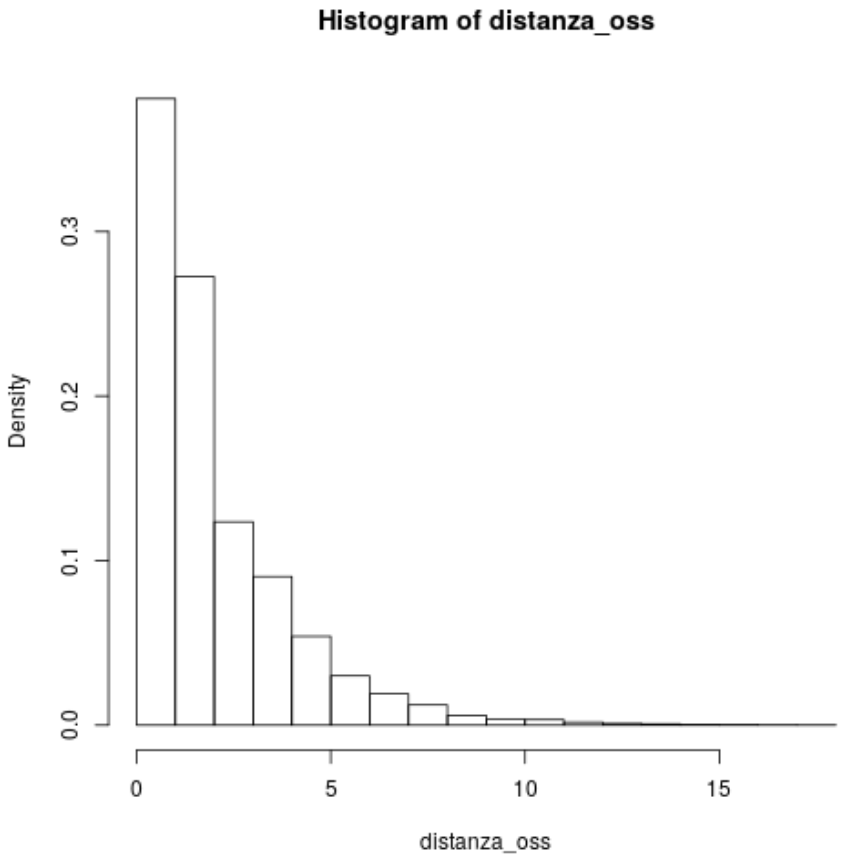


Figura 13.4: plot of chunk unnamed-chunk-3

```
erre2 <- sum(sx1*sy2)/(length(sx1)-1)
erre2
```

```
## [1] 0.6580456
```

Disegnamo il grafico.

```
colori <- c(2,2,1)
colore <- colori[sign(sx1*sy2)+2]
plot (sx1,sy2,col=colore)
abline (a=0,b=erre2)
abline (v=0); abline(h=0)
```

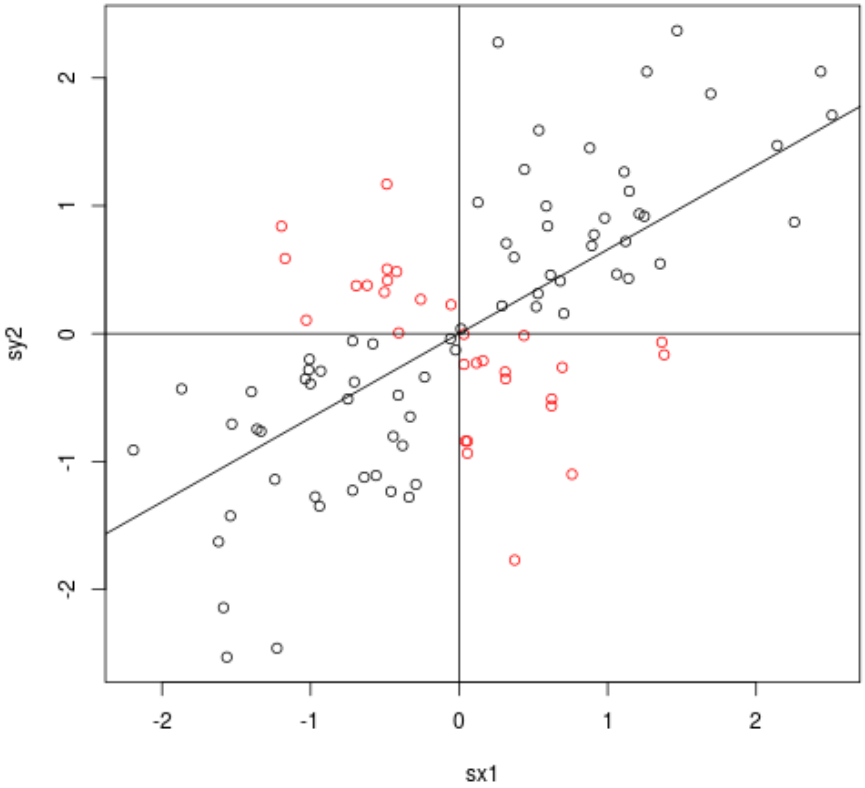


Figura 13.5: plot of chunk corr_plot2

```
p_value_simulazione2 <- rank (c(-abs(erre2),relazioni))[1]/(length(relazio
p_value_simulazione2 * 2 # due code
```

```
## [1] 0.00019998
```

Come previsto, il p-value – calcolato sulla distribuzione osservata dalla simulazione – è basso: 1.9998 × 10-4.

Uso della distribuzione teorica

Dopo aver visto il calcolo del p-value usando la distribuzione della simulazione, utilizziamo la probabilità calcolata a partire dalla distribuzione teorica, *t*.

In primo luogo confrontiamo la distribuzione generata dalla simulazione con la distribuzione teorica, sovrapponendo le due curve.

```
relazioni_t <- relazioni / (sqrt((1-relazioni^2)/(100-2) ))
x <- seq.int(-4, 4, by=0.05)
y <- dt(x, 100-2)
plot (density (relazioni_t),col=4, main="Sovrapposizione delle distribuzioni")
lines (x,y, type="l",col=3)
```

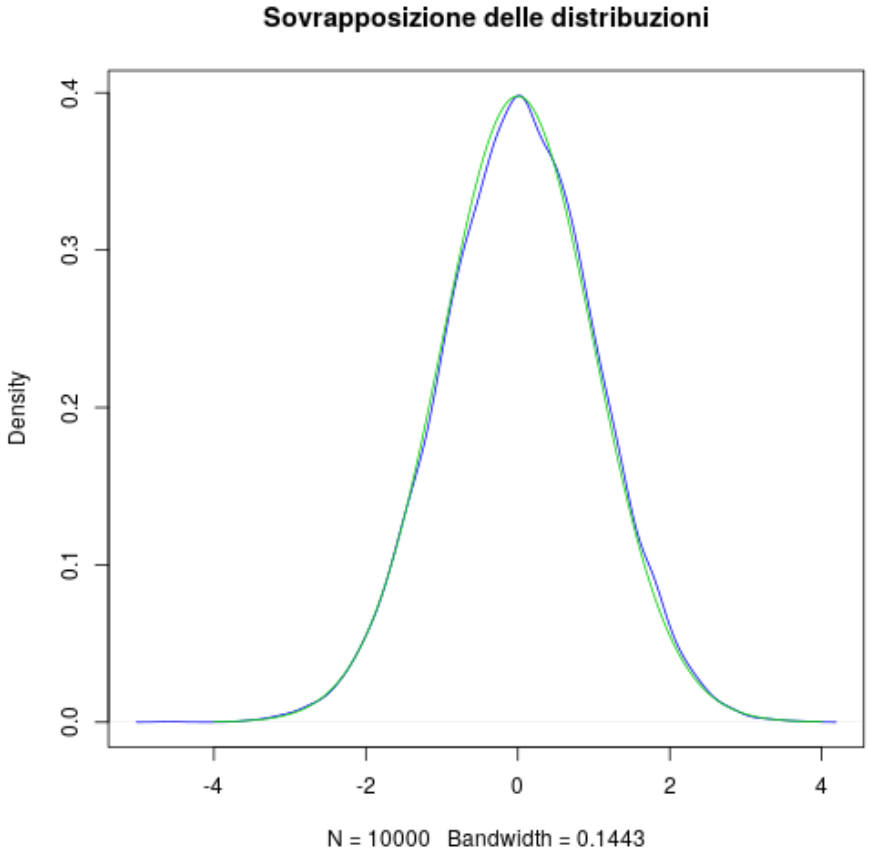


Figura 13.6: plot of chunk corr_sovrapposta

Calcoliamo il p-value usando la distribuzione *t*

Constatato empiricamente che le due distribuzioni sono estremamente simili, decidiamo di calcolare il punteggio t, usando la funzione [eq:rstimat]. Poi, calcoliamo il p-value usando la funzione `pt`

Primo esempio

Mostriamo il calcolo, ed il risultato, del primo esempio.

```
t1 <- abs(erre1) / (sqrt((1-erre1^2)/(100-2) ))
t1
```

```
## [1] 0.6044014
```

```
p_value_t1 <- (1 - pt(t1, df = (100-2)))
p_value_t1 * 2
```

```
## [1] 0.5469736
```

Dunque, t è pari a 0.6044014, p= 0.5469736.

Secondo esempio

Ripetiamo il calcolo, con il secondo esempio.

```
t2 <- abs(erre2)/(sqrt((1 - erre2^2)/(100 - 2)))
p_value_t2 <- (1 - pt(t2, df = (100 - 2)))
p_value_t2 * 2
```

```
## [1] 1.012523e-13
```

t è pari a 8.6514055, p= 1.0125234 × 10-13.

R: uso di `cor.test`

E come di consueto, terminiamo mostrando l'uso della funzione `cor.test (x,y)`. Iniziamo con il primo esempio.

```
correlazione1 <- cor.test (x1,y1)
correlazione1
```

```
##
##  Pearson's product-moment correlation
##
## data:  x1 and y1
## t = -0.6044, df = 98, p-value = 0.547
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2543143  0.1371191
## sample estimates:
##         cor
## -0.06094029
```

```
str(correlazione1)
```

```
## List of 9
##  $ statistic  : Named num -0.604
##   ..- attr(*, "names")= chr "t"
##  $ parameter  : Named int 98
##   ..- attr(*, "names")= chr "df"
##  $ p.value    : num 0.547
##  $ estimate   : Named num -0.0609
##   ..- attr(*, "names")= chr "cor"
##  $ null.value : Named num 0
##   ..- attr(*, "names")= chr "correlation"
##  $ alternative: chr "two.sided"
## $ method    : chr "Pearson's product-moment correlation"
##  $ data.name  : chr "x1 and y1"
##  $ conf.int   : atomic [1:2] -0.254 0.137
##   ..- attr(*, "conf.level")= num 0.95
##  - attr(*, "class")= chr "htest"
```

Leggere i risultati

La funzione ci dice che ha applicato la statistica Pearson's product-moment correlation. Che i gradi di libertà sono 98 (n-2). Il valore della statistica è -0.6044014 ed il p-value è 0.547. La correlazione è pari a -0.0609403. Come prevedibile il p-value è superiore a `0.05`, e dunque non si può rifiutare l'ipotesi nulla di indipendenza fra le due variabili.

Secondo esempio

```
correlazione2 <- cor.test (x1,y2)
correlazione2
```

```
##
##  Pearson's product-moment correlation
##
## data:  x1 and y2
## t = 8.6514, df = 98, p-value = 1.014e-13
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5301505 0.7566634
## sample estimates:
##       cor
## 0.6580456
```

In questo caso, l'ipotesi nulla va rifiutata, in quanto $p - value < 1.01e - 13$.

Regressione lineare

Precipitazioni: dal volume al peso

Torniamo all'esempio, banale, delle precipitazioni. Grazie alla correlazione perfetta fra volume e peso, data una osservazione, conoscendo il volume, possiamo calcolare il peso.

Se guardiamo al grafico, possiamo notare che il peso stimato incrocia il volume proprio lungo la linea di regressione.

La linea di regressione, dunque, stima il valore di y a partire da x (e viceversa, il valore di x conoscendo y).

Esempio 2A: noleggio automobili

Immaginiamo che una agenzia di noleggio auto applichi, per un modello, la seguente tariffa:

- importo fisso di 40 euro al giorno
- 0.08 euro a km percorso

Conoscendo questi due valori, possiamo prevedere esattamente quanto spenderemo. Ad esempio, se ho percorso 140km, spenderò 40 + 0.08*140 = 51.20 euro.

Esempio 2B: noleggio automobili

Immaginiamo che un'altra agenzia applichi, invece, il fisso di 40 euro più il costo della benzina consumata. Immaginiamo che l'auto in questione consumi, in media, 0.06 euro a km.

In questo caso, quando spenderemo, dopo aver fatto 140 km? Il calcolo è lo stesso: 40 + 0.06*140 = 48.40. In questo caso, però, questo valore è solo una *stima* di quello che ci aspettiamo di spendere, in quanto non possiamo essere sicuri che la benzina consumata sia esattamente pari a 8.40 euro.

I valori di 0.06 euro a km, infatti, sono una statistica, calcolata in seguito ad una serie di osservazioni, dove si sono misurati i km effettuati e la benzina consumata.

Il numero di km fatti è il miglior predittore della benzina consumata, ma non è l'unico. Il tipo di percorso e il tipo di guida, fra gli altri, influenzano il consumo.

Quando pagheremo per il noleggio della seconda agenzia, noi possiamo aspettarci un conto di circa 48 euro e 40, ma sappiamo che in quella stima ci sarà un errore, legato a quei fattori che incidono sul consumo ma che non sono annoverati nel calcolo.

Regressione lineare: il modello

Generalizzando dall'esempio precedente, nel modello di regressione lineare bivariato (X,Y), la variabile Y può essere rappresentata tramite la relazione lineare

$$Y = \beta_0 + \beta_1 X + \epsilon$$

ed il valore

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i (i = 1...n)$$

Infine,

$$\hat{y_i} = \beta_0 + \beta_1 x_i (i = 1...n)$$

dove $\hat{y_i}$ è il valore stimato di y.
Spesso si usa la forma $\alpha + \beta x_i$.

Le componenti

- β_0 rappresenta l'intercetta, ovvero il valore di y quando x = 0.
- β_1 (o, nella regressione bivariata, semplicemente β) rappresenta la pendenza della linea, ed è pari alla differenza fra $\hat{y} = f(x)$ e $\hat{y} = f(x + 1)$. Nell'esempio, rappresenta il costo supplementare per ogni km percorso.
- ϵ rappresenta la variabile di errore, ovvero quella varianza in y che non può essere spiegata da x.

Per massimizzare la predittività della regressione è dunque necessario scegliere due parametri β_0 e β_1 capaci di minimizzare l'errore e, possibilmente, di escludere dei bias (errori sistematici).

Somma dei quadrati degli errori

Nella regressione lineare semplice, la misura dell'errore ϵ si basa sulla somma dei quadrati degli errori (in inglese *sum of squared residuals, SSR*):

$$SSR = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (\hat{y_i} - y_i)^2 = \sum_{i=1}^{n} ((\beta_0 + \beta_1 x_i) - y_i)^2$$

dove $\hat{y_i}$ è il valore stimato di y.

Si tratta dunque di identificare i parametri β_0 e β_1 capaci di minimizzare SSR.

Stime di β_0 e β_1

Le due stime che minimizzano la somma dei quadrati degli errori (SSR) sono

$$\hat{\beta_1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta_0} = \bar{y} - \hat{\beta_1}\bar{x}$$

Proporzione di varianza spiegata e residua

[sec:reglinspiegata] La varianza di Y può essere divisa fra la varianza spiegata da $\beta_0 + \beta_1 X$ e la varianza residua, o di errore. La proporzione di varianza spiegata è pari a $R^2 = r^2$.

$$R^2 = 1 - \frac{SSR/(n-1)}{var(Y)} = 1 - \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n}(y_i - \bar{y})_i^2}$$

$0 \leq R^2 \leq 1$. R^2 è pari al quadrato di r.

Va notato che quanto SSR è 0 R^2 è 1, e quando $SSR = var(Y)$ $R^2 = 0$.

Assunti della regressione lineare

- La relazione fra le variabili dev'essere lineare
- L'errore è una variabile casuale con media zero e distribuzione normale
- Gli errori non sono fra loro correlati
- La varianza dell'errore è costante

Distribuzione di y_i

Tenuto conto che

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i (i = 1...n)$$

se ϵ_i ha distribuzione normale, media 0, varianza σ^2, la distribuzione di y_i sarà

$$y_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2)(\forall i = 1...n)$$

dove $N()$ è la distribuzione normale.

R: la funzione `lm ()`

In R, la regressione lineare si calcola utilizzando la funzione `lm()`. La sintassi usata è $lm(y \sim x)$ dove y è la variabile dipendente e x la variabile indipendente.

Per disegnare la retta di regressione, si passa il risultato di lm() alla funzione `abline`, che disegna una linea con parametri a e b.

```
modello1 <- lm (y1~x1)
summary (modello1)
```

```
##
## Call:
## lm(formula = y1 ~ x1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1996 -1.9044 -0.1224  2.1952  7.6711
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  51.9550     2.8837  18.017   <2e-16 ***
## x1           -0.0885     0.1464  -0.604    0.547
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.952 on 98 degrees of freedom
## Multiple R-squared:  0.003714,   Adjusted R-squared:  -
0.006452
## F-statistic: 0.3653 on 1 and 98 DF,  p-value: 0.547
```

`summary(lm())` ci restituisce molte informazioni. `Intercept`, ad esempio, ci dice se l'intercetta è significativamente diversa da 0 (informazione generalmente poco interessante).

Molto più importante la seconda linea, che calcola t e il p-value di β_1. R-squared è R^2. Il p-value è calcolato anche attraverso la statistica F (che non abbiamo affrontato). Il risultato, nel caso di correlazione bivariata, è lo stesso: $p - value : 0.5469736$.

Il grafico e la retta di regressione

Abbiamo già introdotto la retta di regressione, `abline`, che utilizza proprio il risultato del modello lineare `lm()`.

```
plot (x1,y1)
abline (modello1)
```

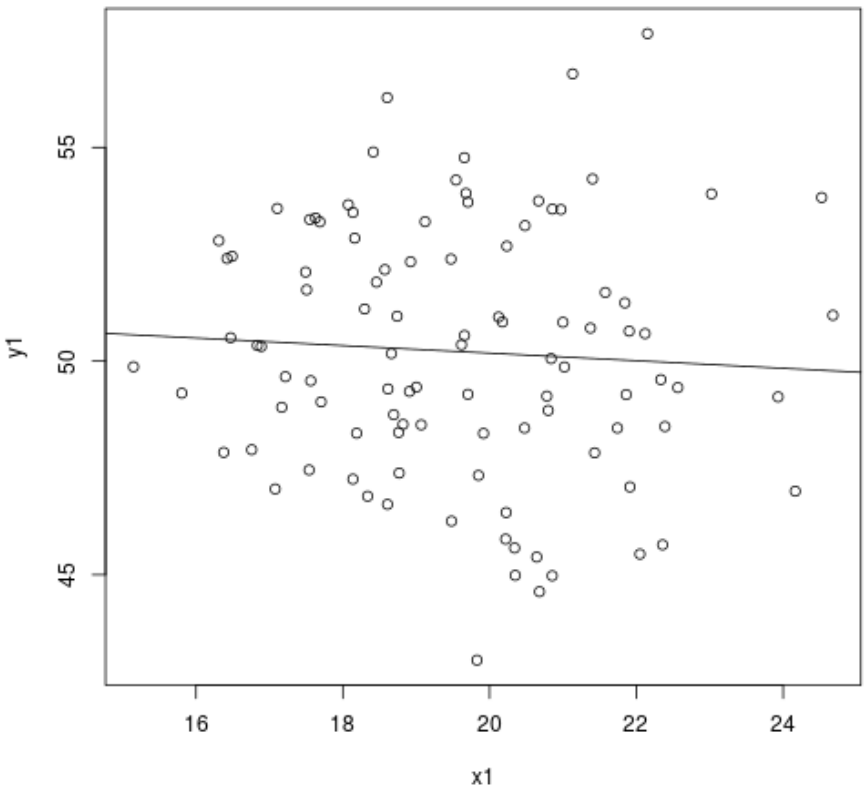


Figura 13.7: plot of chunk corr_plot_modello1

Varianza dei residui, R^2

Nel paragrafo [sec:reglinspiegata] abbiamo introdotto il concetto di rapporto fra varianza spiegata e varianza residua. la funzione `lm()` restituisce, fra le altre cose, i residui: `$residuals`. Sappiamo che la varianza totale è pari alla varianza spiegata più la varianza residua. Sappiamo dunque la varianza spiegata è pari a $R^2 = 1 - \frac{var(residui)}{var(Y)}$. Nelle prossime righe di R calcoliamo R^2, e lo confrontiamo con il r^2, per mostrare che sono uguali.

```
residui1 <- modello1$residuals
R2_1 <- 1 - var(residui1) / var (y1)
R2_1
```

```
## [1] 0.003713718
```

```
erre1^2
```

```
## [1] 0.003713718
```

Come vediamo, R^2 è pari a 0.0037137, e r^2 è pari a 0.0037137.

Grafico dei residui

Il grafico dei residui ci permette di visualizzare la distribuzione dell'errore. È importante per verificare gli assunti del modello.

```
plot (modello1$fitted.values, modello1$residuals)
abline (lm(modello1$residuals~modello1$fitted.values))
lines(smooth.spline(modello1$fitted.values, modello1$residuals), col = "re
```

Il secondo esempio

Usiamo `lm()` sulle variabili del secondo esempio.

```
modello2 <- lm (y2~x1)
summary (modello2)
```

```
##
## Call:
## lm(formula = y2 ~ x1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.2075 -1.6113  0.1272  1.3910  6.4781
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2821     2.2749   0.124    0.902
## x1            0.9993     0.1155   8.651 1.01e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.329 on 98 degrees of freedom
```

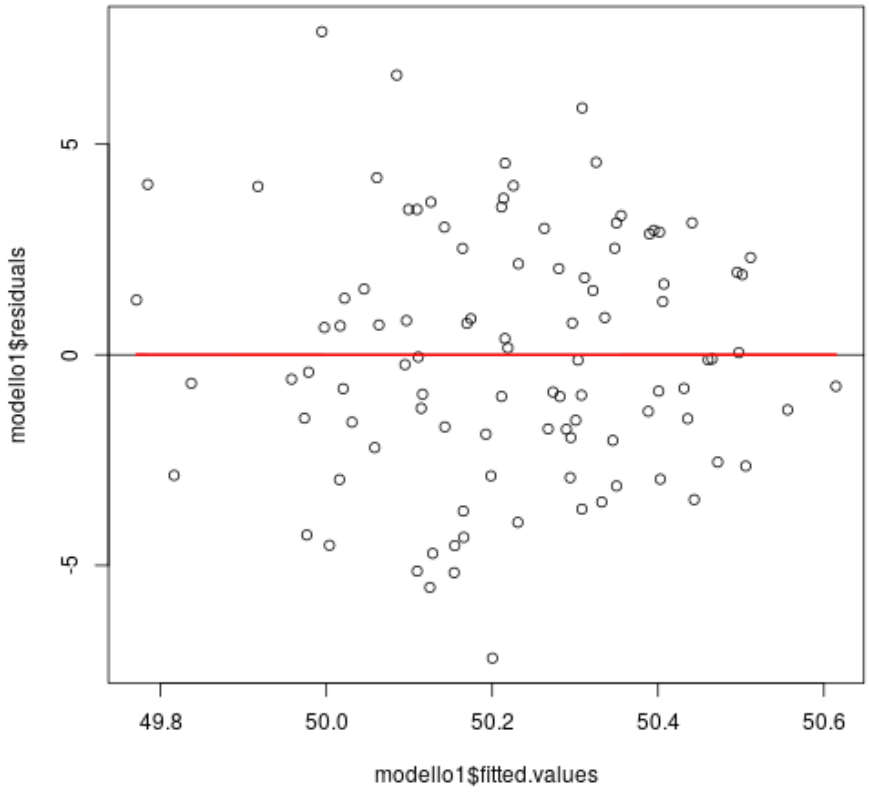


Figura 13.8: plot of chunk corr_plot_residui1

```
## Multiple R-squared:  0.433,  Adjusted R-squared:  0.4272
## F-statistic: 74.85 on 1 and 98 DF,  p-value: 1.014e-13
```

In questo caso, il modello è significativo, in quanto il p-value di x1 è 1.014187 × 10-13.

Visualizziamo il grafico.

```
plot (x1,y2)
abline (modello2)
```

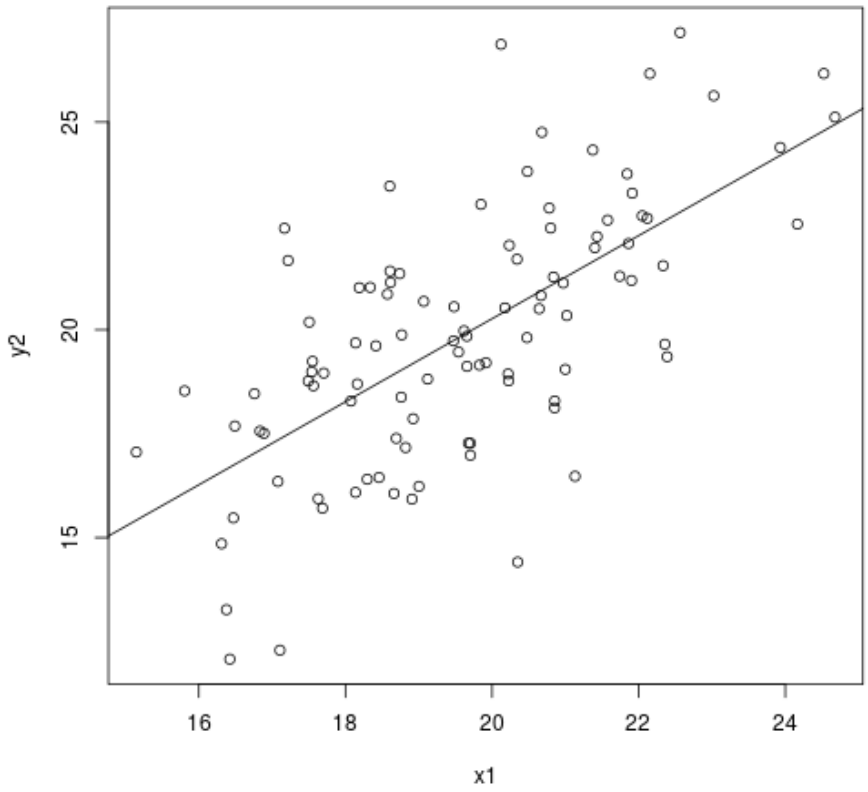


Figura 13.9: plot of chunk corr_plot_modello2

Usiamo i residui, calcoliamo R^2.

```
residui2 <- modello2$residuals
R2_2 <- 1 - var(residui2) / var (y2)
R2_2
```

```
## [1] 0.433024
```

```
erre2^2
```

```
## [1] 0.433024
```

Residui, R^2

Visualizziamo il grafico dei residui su x.

```
plot (modello2$fitted.values, modello2$residuals)
abline (lm(modello2$residuals~modello2$fitted.values))
lines(smooth.spline(modello2$fitted.values, modello2$residuals), col = "red", lwd =
```

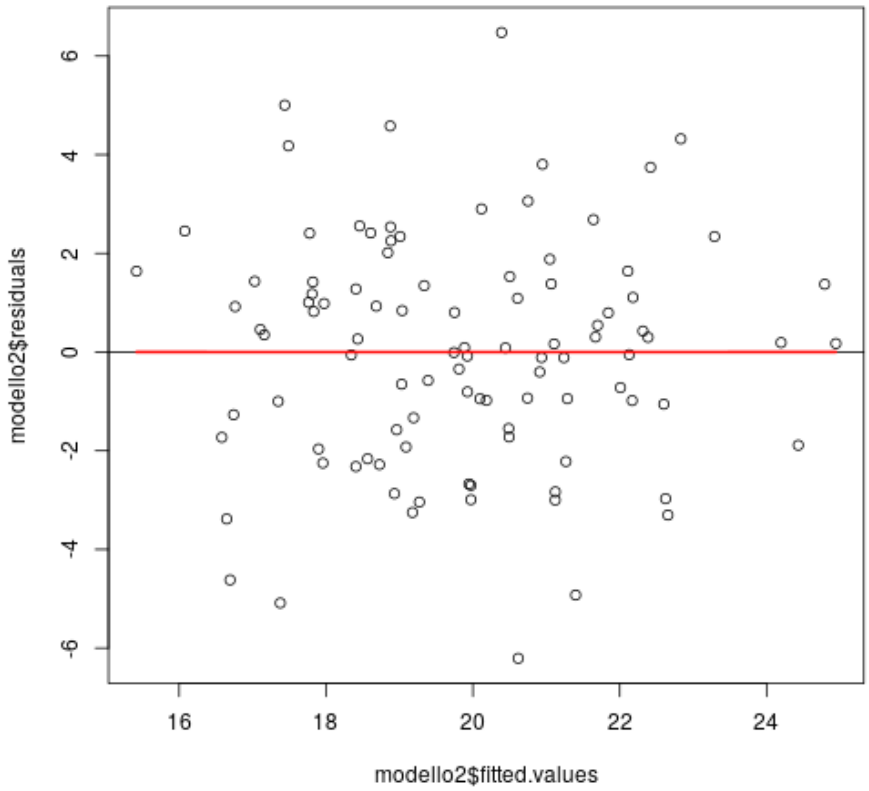


Figura 13.10: plot of chunk corr_plot_residui2

In questo caso, alla linea di regressione abbiamo aggiunto anche una `smooth.spline`, ovvero una curva che segue l'andamento della relazione fra punteggi stimati e residui. Questa curva ci può aiutare a capire se l'assunto di linearità è violato.

Violazione degli assunti

Gli assunti della regressione lineare

Ricordiamo gli assunti della regressione lineare:

- La relazione fra le variabili dev'essere lineare

- L'errore è una variabile casuale con media zero e distribuzione normale
- Gli errori non sono fra loro correlati
- La varianza dell'errore è costante (omoskedasticità)

Quali sono le conseguenze della violazione degli assunti?

La violazione di linearità è il caso più importante, di cui discuteremo estesamente nelle prossime slide.

Correlazione fra gli errori

La correlazione fra gli errori è sintomo di non indipendenza fra le misure.

La non indipendenza fra le misure è un problema di cui va tenuto conto nelle misure ripetute.

Nel caso di misure su di un campione estratto casualmente, la non indipendenza delle misure e dell'errore è meno probabile.

La violazione di questo assunto può essere diagnosticata attraverso un test di autocorrelazione dei residui

Omoskedasticità: varianza dell'errore costante

Se la varianza dell'errore non è costante, l'intervallo di confidenza della distribuzione di Y non sarà correttamente predittivo, in quanto sovrastimato nella parte del grafico in cui la varianza dell'errore è minore, e sottostimato nelle parti dove è maggiore.

La violazione dell'omoskedasticità può essere diagnosticata plottando i residui sui valori attesi: se la dispersione degli errori non è omogenea, possiamo sospettare una violazione della costanza della varianza dell'errore.

Normalità dell'errore

La violazione di questo assunto comporta la compromissione della stima sia dei coefficienti β_1 e β_0 che dei valori di confidenza della distribuzione di Y su X.

Per verificare graficamente la normalità della distribuzione dell'errore, si può utilizzare il grafico qqnorm e qqline.

Per verificarla inferenzialmente, si possono usare il test di *Kolmogorov-Smirnov*, `ks.test`, o il test di normalità *Shapiro-Wilk*: `shapiro.test`

Violazione della linearità

È uno degli aspetti più delicati, in quanto può indurre ad inferenze scorrette.

La non linearità può essere diagnosticata attraverso la visualizzazione del grafico di dispersione delle due variabili, il grafico di dispersione dei residui sui valori attesi, o sulla variabile X.

Da un punto di vista inferenziale, è possibile applicare la statistica *Harvey-Collier*: `harvtest` (va caricata la libreria `lmtest`: `library(lmtest::harvtest)`.

In alcune circostanze, è possibile applicare una trasformazione non lineare ad una o entrambe le variabili, per rendere lineare la relazione.

Secondo esempio: testiamo gli assunti

Focalizziamoci sul secondo esempio, relativo a due variabili correlate, e testiamo gli assunti di normalità e di linearità.

Verifico la normalità della distribuzione dell'errore

Uso il test di Shapiro-Wilk per testare la normalità della distribuzione dell'errore.

```
shapiro2 <- shapiro.test(modello2$residuals)
shapiro2
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modello2$residuals
## W = 0.99467, p-value = 0.9657
```

Poiché $ p= 0.9656781 $, non rifiuto l'ipotesi nulla di normalità del modello. L'assunto, dunque, non è violato.

Valuto la linearità del modello

Utilizziamo ora il test *Harvey-Collier* per testare la linearità del modello. Attenzione: per usare la funzione `harvtest` è necessario, prima, importare la libreria *lmtest*, con il comando `library(lmtest)`.

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
harvtest2 <- harvtest(y2 ~ x1, order.by = ~ x1)
harvtest2
```

```
##
##  Harvey-Collier test
##
## data:  y2 ~ x1
## HC = 0.38184, df = 97, p-value = 0.7034
```

Poiché $p = 0.7034153$, non rifiuto l'ipotesi nulla di linearità del modello. L'assunto, dunque, non è violato.

Terzo esempio, non lineare

Infine, un ultimo esempio.

```
x3 <- runif (100,-2,6)
y3 <- x3^2+ rnorm(100,0,1)
x3 <- x3 + 10
modello3 <- lm (y3~x3)
summary (modello3)
```

```
##
## Call:
## lm(formula = y3 ~ x3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.452 -3.882 -1.962  3.089 11.942
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -35.7243     2.7999  -12.76   <2e-16 ***
## x3            3.7059     0.2299   16.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.053 on 98 degrees of freedom
## Multiple R-squared:  0.7261, Adjusted R-squared:  0.7233
## F-statistic: 259.7 on 1 and 98 DF,  p-value: < 2.2e-16
```

Come vediamo dal codice, la relazione fra y3 e x3, al netto della varianza non spiegata, è data da $x3^2$. Applichiamo il modello lineare, e leggiamo i risultati.

Il grafico

Analizziamo, ora, il grafico.

```
par(mfrow=c(1,2))
plot (x3,y3,main="grafico di dispersione")
abline (modello3)
plot (modello3$fitted.values,modello3$residuals,main="punteggi attesi vs r
abline (lm(modello3$residuals~modello3$fitted.values))
lines(smooth.spline(modello3$fitted.values,modello3$residuals), col='red',
```

Risulta evidente, dal grafico a sinistra, che la relazione fra x3 e y3 non è lineare. La non linearità è ancor più evidente nel grafico dei residui, a destra.

Usiamo il test *Harvey-Collier* per valutare la linearità.

```
harvtest3 <- harvtest(y3 ~ x3, order.by = ~ x3)
harvtest3
```

```
##
##  Harvey-Collier test
##
```

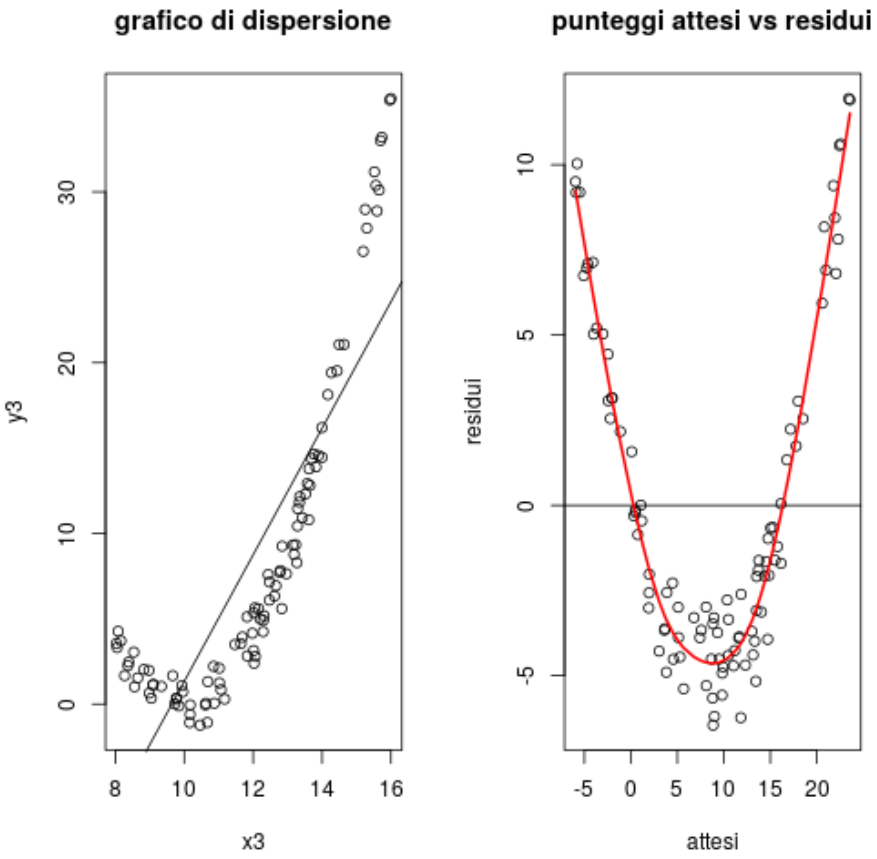


Figura 13.11: plot of chunk corr_plot_modello3

```
## data:  y3 ~ x3
## HC = 9.5248, df = 97, p-value = 1.425e-15
```

$p = 1.4248535 * 10 < sup > -15 < /sup >< 0.05$, e dunque rifiuto l'ipotesi nulla di linearità del modello. L'assunto di linearità del modello è violato, come previsto dall'analisi qualitativa del grafico.

I 4 data-set di Anscombe

In letteratura, sono noti i 4 insiemi di dati pubblicati da Anscombe. I quattro insieme di dati sono particolari: le quattro y hanno la stessa media (7.5), deviazione standard (4.12), correlazione (0.816) e linea di regressione. Com'è possibile notare dal grafico, però, i quattro insiemi sono *qualitativamente* molto diversi. L'esempio, è finalizzato a ricordarci che calcolare il modello lineare non basta, e che una attenta analisi dei grafici di dispersione è indispensabile, per evitare di trarre conclusioni inferenziali indebite.

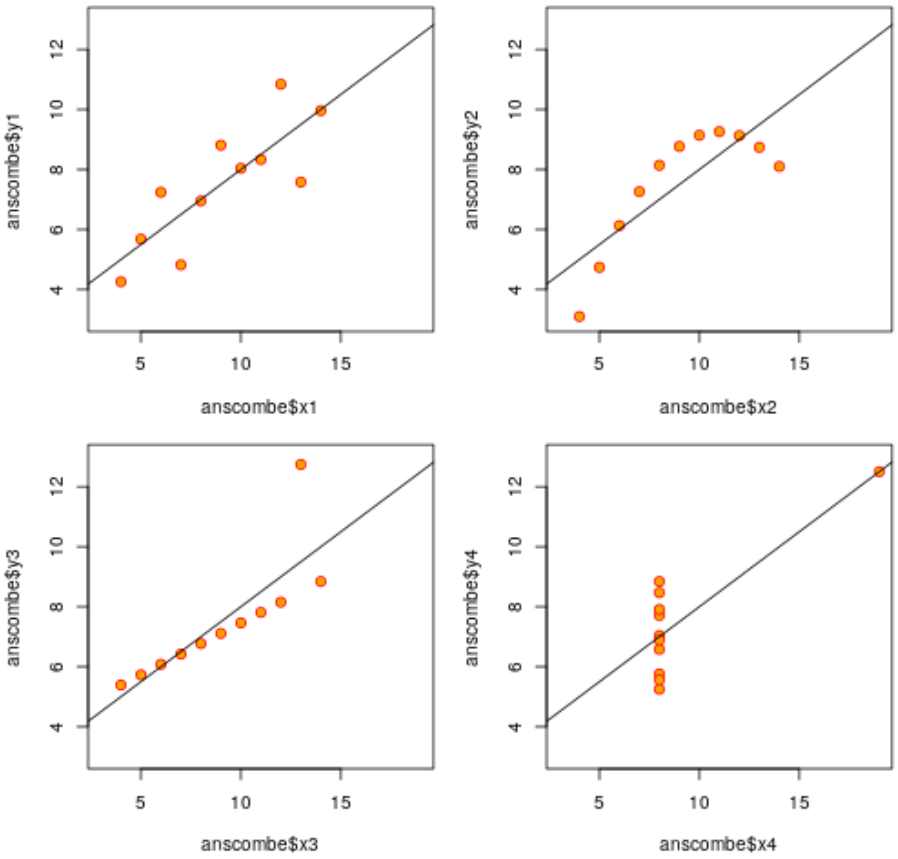


Figura 13.12: plot of chunk corr_anscombe

Coefficiente di Spearman

Dipendenza monotona

Nelle circostanze in cui la relazione fra le due variabili non sia lineare, ma tenda ad essere comunque monotona, è possibile utilizzare il modello non-parametrico della correlazione: ρ di Spearman.

In questo modello, il calcolo della relazione si effettua non sui valori delle due variabili, ma sulla loro posizione ordinale.

Questa statistica, pertanto, può essere applicata anche nella circostanza in cui una o entrambe le variabili siano di tipo ordinale.

Assunti del modello di Spearman

Gli assunti sono i seguenti:

- le due variabili devono essere almeno ordinali, non necessariamente ad intervalli;
- su entrambe le variabili, le diverse osservazioni devono essere fra loro indipendenti;
- si assume che vi sia, fra le variabili, una relazione di tipo monotono;

Quarto esempio, sigmoide

Introduciamo un quarto esempio, dove la curva fra x4 e y4 è una sigmoide.

```
x4 <- rnorm (200,0,4)
y4 <- (1 / (1 + exp(-x4)))*10+rnorm(200,0,0.3)
x4 <- x4+8+rnorm(200,0,0.3)
```

Il grafico

Prima di ogni calcolo, mostriamo il grafico.

```
par(mfrow=c(1,1))
plot (x4,y4)
abline(lm(y4~x4))
lines(smooth.spline(x4,y4), col='red', lwd=2)
```

Dal grafico appare chiaro che una relazione fra x4 e y4 esiste, ma che la relazione non è lineare. Il modello lineare riesce comunque a cogliere la relazione, ma il modello predittivo risulta sostanzialmente scorretto.

Calcolo di r

Calcoliamo, comunque, r, usando `cor.test`. Negli esempi precedenti, l'argomento `method='pearson'` era stato omesso, in quanto costituisce il default del metodo. Adesso, al contrario, lo rendiamo esplicito.

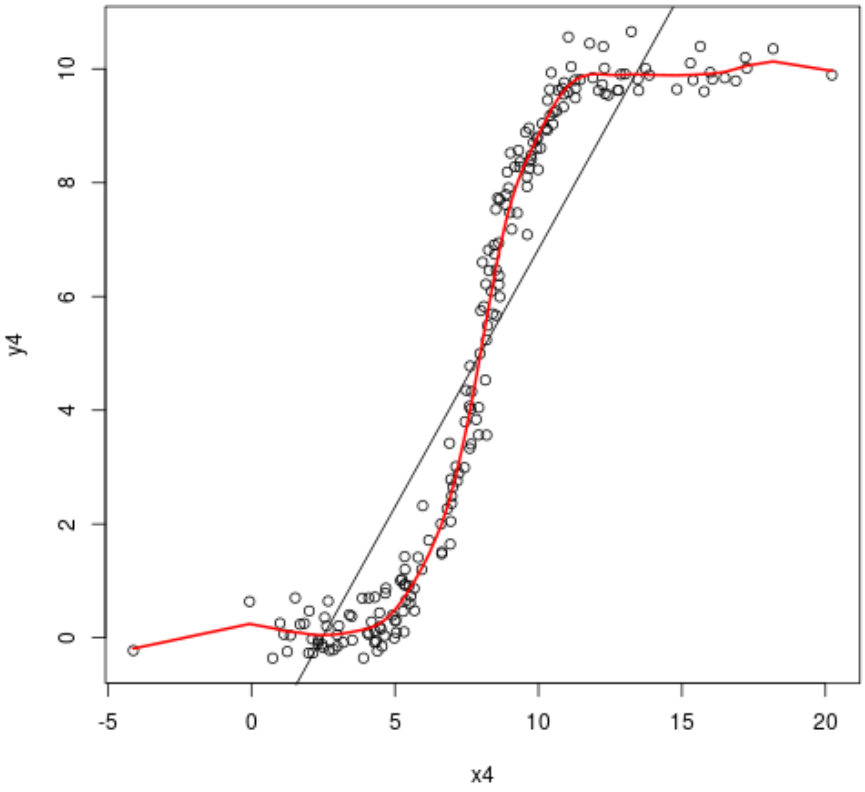


Figura 13.13: plot of chunk corr_plot_4

```
cor.test (x4,y4,method='pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  x4 and y4
## t = 29.482, df = 198, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8730539 0.9253555
## sample estimates:
##       cor
## 0.9024791
```

Coefficiente di Spearman

Calcoliamo, ora, a mano, il coefficiente ρ di Spearman. Come abbiamo detto, la statistica non parametrica utilizza, al posto dei valori numerici di x e y, le loro posizioni ordinali. Il primo passaggio, dunque, è quello di trasformare i punteggi nei rispettivi ranking (e, nel nostro algoritmo, di scalarli). Poi, utilizziamo la consueta formula per calcolare il coefficiente.

```
rankx4 <- scale(rank(x4))
ranky4 <- scale(rank(y4))
spearman4 <- sum(rankx4*ranky4)/(length(rankx4)-1)
spearman4
```

```
## [1] 0.9729603
```

```
plot (rankx4,ranky4)
```

Per capire meglio il meccanismo, disegnamo il grafico di dispersione dei ranking delle due variabili. Come possiamo vedere, la trasformazione rende lineare la relazione monotona.

Coefficiente di Spearman, cor.test

Infine, come di consueto, utilizziamo la funzione di R: `cor.test(x4, y4, method = "spearman")`. Se si vuole usare la ρ di Spearman, è necessario specificare `method = "spearman"`.

```
cor_test_S4 <- cor.test (x4,y4,method='spearman')
cor_test_S4
```

```
##
##  Spearman's rank correlation rho
##
## data:  x4 and y4
## S = 36052, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.9729603
```

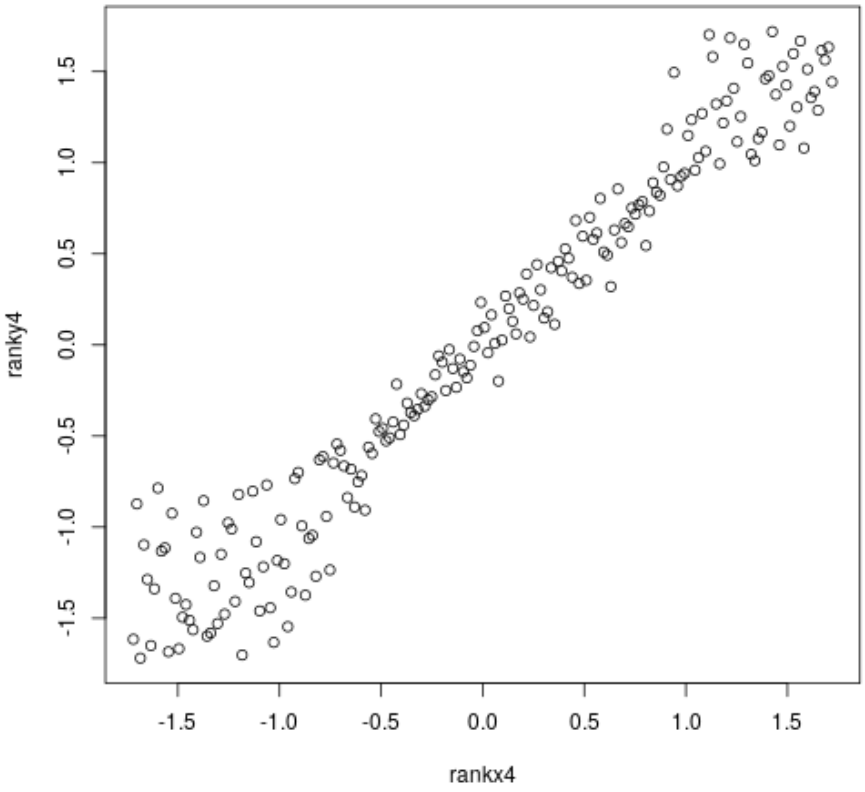


Figura 13.14: plot of chunk corr_spearman4

Leggere i risultati

La funzione calcola la statistica S= 3.6052 × 104, rho= 0.9729603, p= 0.

Conclusioni

Capitolo 14

L'analisi della varianza

Analisi della Varianza

Introduzione

Confronto fra variabili categoriali e a intervalli

Abbiamo visto, nel capitolo [ch:t_test], come il t-test ci permetta di confrontare le medie di due gruppi, e di valutare se la loro differenza è significativa.

Il confronto su coppie di elementi è applicabile se abbiamo una variabile indipendente di tipo categoriale con due soli livelli. Vi sono circostanze, però, in cui il confronto deve avvenire fra più di due gruppi. Il caso più semplice è quando la variabile indipendente ha tre o più livelli. Una circostanza più complessa emerge quando le variabili indipendenti sono più di due.

In quale modo possiamo affrontare una simile eventualità?

Confronto a coppie

Una possibile risposta è quella di confrontare ogni possibile combinazione di coppie di gruppi.

Nell'esempio semplice di una variabile indipendente a tre livelli, confrontare il gruppo 1 con il gruppo 2, il gruppo 1 con il 3, il 2 con il 3.

Sebbene questi confronti a coppie siano non solo possibili, ma realmente utilizzati nei confronti `post-hoc`, l'utilizzo esclusivo di questa metodologia incorre in due limiti.

Problemi del confronto a coppie

Il primo è che, moltiplicando il numero dei confronti, si accresce la probabilità di incorrere in un errore di tipo I.

Immaginiamo, ad esempio, di accettare un valore α pari a 0.05. Questo significa accettare la possibilità che, il 5% delle volte, si rifiuti indebitamente l'ipotesi nulla.

Ma se, invece di un solo confronto, ne facciamo due, qual è la probabilità che si rifiuti indebitamente almeno una volta l'ipotesi nulla?

Ricordiamo che, se l'ipotesi che si sta valutando è se la variabile indipendente influisce su quella dipendente, basta rilevare una differenza significativa fra i gruppi per inferire una influenza.

Ma se i confronti sono numerosi, la probabilità di incorrere in un errore del primo tipo aumenta. Nell'esempio con tre gruppi, poiché i confronti sono tre, con un α pari a 0.05 la probabilità di rifiutare erroneamente l'ipotesi nulla diventa pari a 0.1023.

Più variabili indipendenti

Il secondo inconveniente emerge quando le variabili indipendenti sono più di una. In questo caso, dal confronto a coppie è difficile far emergere quali variabili indipendenti hanno un'influenza significativa sulla variabile dipendente e quali no, così come diventa difficile far emergere l'eventuale interazione fra le variabili indipendenti.

Questi due inconvenienti ci costringono ad identificare una metodologia capace di generalizzare ai confronti con più variabili indipendenti e capace di mantenere sotto controllo l'errore di tipo I.

Varianze

Nel capitolo [ch:correlazione] sulla regressione lineare sono stati introdotti i concetti di varianza totale, spiegata e residua.
La varianza di una variabile ci offre una misura della distribuzione di quella variabile. Conoscendo la media e la varianza (o la deviazione standard) di una variabile, ed assumendo una distribuzione di tipo normale, possiamo fare delle previsioni su Y. Possiamo assumere che $\bar{Y}$ sia il valore atteso di y più probabile, e possiamo stimare la probabilità dell'occorrenza di una osservazione y.

Naturalmente, la varianza di una variabile influisce sulla nostra capacità di fare delle previsioni. Se la varianza è prossima allo zero, noi possiamo prevedere con certezza che il valore atteso di una osservazione y sarà molto vicino alla media $\bar{Y}$

Varianza spiegata e previsioni

Se riprendiamo l'esempio banale delle precipitazioni atmosferiche: La variazione delle osservazioni può essere molto ampia: da pochi millimetri a decine di centimetri. Tuttavia, tutta la variazione di peso, però, può essere spiegata dal volume dell'acqua (attenzione: **spiegata non significa causata**).

Passiamo al caso meno banale di dover stimare il consumo di carburante durante l'uso di un'auto in una giornata, conoscendo i km percorsi.

In questo caso, come abbiamo visto, la scommessa sarà meno certa, in quanto il consumo è legato anche al tipo di guida, al tipo di percorso, alle condizioni di traffico e così via. Ciononostante, conoscere il numero di km percorsi mi permette una stima molto più accurata di quanto potrei fare semplicemente tirando ad indovinare.

La varianza residua, ovvero la varianza dei residui, sarà molto più bassa della varianza totale. Il numero di km costituisce dunque un predittore molto utile del consumo di carburante.

Varianza residua e previsioni

Il vantaggio di poter applicare la regressione lineare è che la relazione fra due variabili può essere espressa attraverso due soli parametri: β_0 e β_1, ovvero l'intercetta e la pendenza della linea.

Nel caso di variabili indipendenti di tipo categoriale, naturalmente, non è possibile assumere alcuna linearità, e dunque non possono bastarci quei due parametri.

Un esempio: gli affitti in una città

Immaginiamo di voler prendere in affitto un appartamento in una città di medie dimensioni, e vogliamo capire se, in differenti quartieri, i prezzi sono significativamente diversi.

Immaginiamo dunque di fare una ricerca sistematica, usando alcuni siti specializzati, raccogliendo le informazioni relative a 60 appartamenti, distribuiti su 3 quartieri: 20 nel quartiere A, 20 nel B, 20 nel C.

Quello che vogliamo capire è se, nei differenti quartieri, i prezzi sono significativamente diversi.

R: generare l'esempio

Con R, possiamo generare il dataset di 60 valori, invocando `rnorm`.

Decidiamo di generare 20 valori con media 550 e sd 30 (quartiere A), 20 con media 590 e sd 30 (quartiere B), 20 con media 620 e sd 30 (quartiere C).

Usando le funzioni factor, levels e data.frame creiamo il dataframe con 2 colonne (prezzo, quartiere) e 60 righe.

R: il codice

```
zonaA <- round (rnorm (20,55,3))*10
zonaB <- round (rnorm (20,56,3))*10
zonaC <- round (rnorm (20,62,3))*10
zone <- c (zonaA,zonaB,zonaC)
fZone <- factor (c(rep('A',20),rep('B',20),rep('C',20)))
affitti <- data.frame (prezzo=zone,quartiere=fZone)
```

Grafico e varianza

Nel grafico [fig:esempio1], plottiamo le 60 osservazioni: in rosso le venti del quartiere A, verde il quartiere B, blu il C. La linea rossa marca il valore medio di A, la verde la media di B, blu la media di C. Le due righe rosse tratteggiate, l'intervallo di confidenza al 95% delle osservazioni in A, le verdi in B, le blu in C. La riga continua nera rappresenta la media generale, le due righe tratteggiate nere l'intervallo di confidenza generale, al 95%.

```
plot(c(1,60),c(min(affitti$prezzo),max(affitti$prezzo)), type = "n", xlab
points(affitti$prezzo,col=as.integer(affitti$quartiere)+1)

lines (c(1,20),rep(mean(zonaA),2),col=2)
lines (c(21,40),rep(mean(zonaB),2),col=3)
lines (c(41,60),rep(mean(zonaC),2),col=4)

lines (c(1,60),rep(mean(affitti$prezzo),2),col=1)

lines (c(1,20),rep(mean(zonaA)+2*sd(zonaA),2),col=2,lty=2)
lines (c(1,20),rep(mean(zonaA)-2*sd(zonaA),2),col=2,lty=2)
```

```
lines (c(21,40),rep(mean(zonaB)+2*sd(zonaB),2),col=3,lty=2)
lines (c(21,40),rep(mean(zonaB)-2*sd(zonaB),2),col=3,lty=2)

lines (c(41,60),rep(mean(zonaC)+2*sd(zonaC),2),col=4,lty=2)
lines (c(41,60),rep(mean(zonaC)-2*sd(zonaC),2),col=4,lty=2)

lines (c(1,60),rep(mean(affitti$prezzo)+2*sd(affitti$prezzo),2),col=1,lty=2)
lines (c(1,60),rep(mean(affitti$prezzo)-2*sd(affitti$prezzo),2),col=1,lty=2)
```

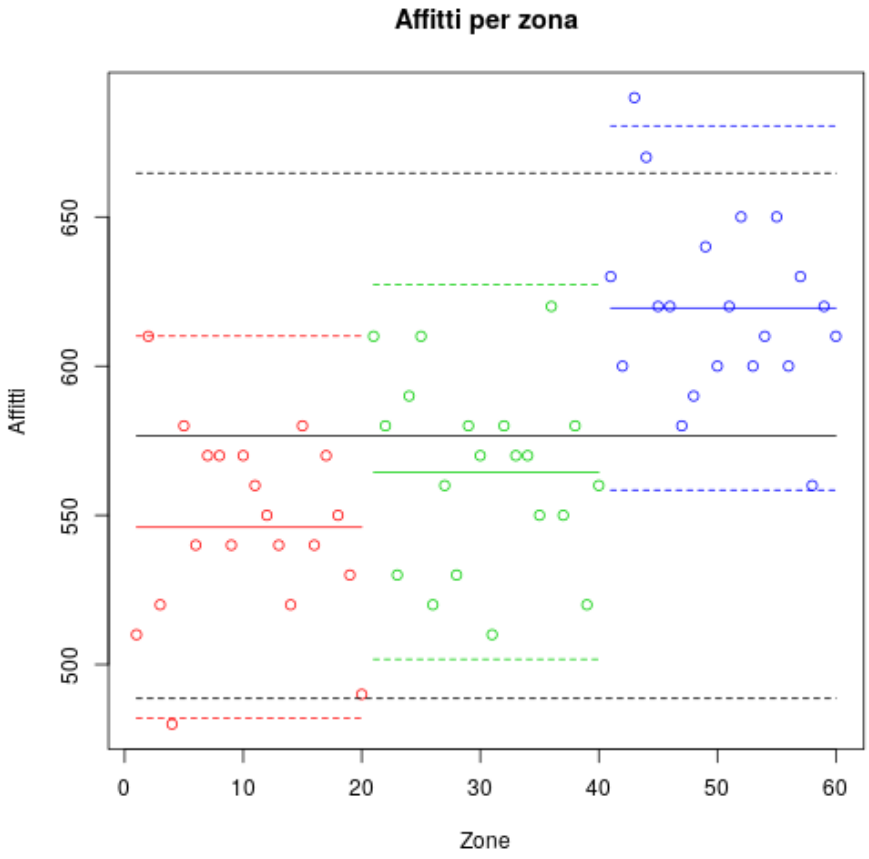


Figura 14.1: plot of chunk unnamed-chunk-2

Nonostante le variazioni dovute al caso, è chiaro che l'intervallo di confidenza dei singoli gruppi è minore (e diverso) dell'intervallo di confidenza totale. Questo significa che conoscere il quartiere dove un appartamento è collocato mi permette di fare delle previsioni migliori in merito al prezzo che mi aspetto di pagare.

Inferenza e previsioni

L'analisi bivariata (descrittiva e inferenziale) ci permette dunque innanzitutto di capire se una variabile influisce su di un'altra.

In secondo luogo, se l'influenza è statisticamente significativa, conoscere il valore della prima variabile ci permette di fare delle previsioni più accurate sulla seconda.

Rapporto fra varianza spiegata e residua

L'analisi della varianza è la statistica inferenziale che valuta se vi è una relazione fra una (o più) variabili indipendenti, di tipo categoriale, e una variabile quantitativa (almeno ad intervalli). Il principio su cui si basa la statistica è proprio la percentuale di varianza spiegata dal modello riespetto alla varianza totale.

La misura che viene presa in considerazione in questa statistica è dunque un rapporto: il rapporto fra varianza spiegata dal modello e varianza residua (ovvero la differenza fra la varianza totale e quella spiegata). Se il rapporto supera un determinato valore critico, si rifiuta l'ipotesi nulla (secondo cui non vi è relazione fra la variabile indipendente e quella dipendente).

L'analisi della Varianza

Quello che l'analisi della varianza ci permette di capire è se le medie della variabile dipendente osservate nei diversi gruppi sono o meno statisticamente diverse. Più precisamente, ci permette di stabilire se esistono almeno due gruppi la cui media sia statisticamente diversa [1].

Il vantaggio di questo approccio, rispetto al confronto fra coppie di gruppi, è duplice:

- non vi è una proliferazione dell'errore di tipo I, in quanto il confronto è unico
- nel caso di più variabili indipendenti, è possibile stimare l'influenza di ognuna delle variabili indipendenti, nonché della loro interazione.

L'ipotesi nulla

L'ipotesi nulla assume che la media dei gruppi non sia fra loro diversa, e dunque che le medie dei gruppi siano approssimativamente pari alla media generale.

Se le medie dei vari gruppi sono tutte perfettamente uguali alla media generale, anche la varianza dei gruppi sarà pari alla varianza generale, e dunque la varianza spiegata sarà pari a zero.

Errore di campionamento

A causa dell'errore di campionamento, però, sappiamo che, anche qualora l'ipotesi nulla sia vera, le medie dei gruppi potranno discostarsi dalla media generale, e dunque la varianza spiegata misurata sarà superiore a zero.

[1] questo tipo di statistica viene definito *omnibus*

Come nei casi già visti (t test, correlazione, chi quadro), il valore del rapporto fra varianze va dunque confrontato con una distribuzione (teorica o generata empiricamente, ad esempio attraverso una simulazione) in modo da valutare se la proporzione di varianza spiegata è da attribuire al caso (errore di campionamento), e dunque va accettata l'ipotesi nulla, oppure no.

Il calcolo

Somme dei quadrati

Per calcolare il test dell'analisi della varianza, dobbiamo calcolare tre valori.

- la somma dei quadrati dell'errore totale, SS_T;
- la somma dei quadrati dell'errore residuo, SS_R;
- la somma dei quadrati del modello, SS_M;

Per calcolare le varianze totale, residua e spiegata dobbiamo dividere gli SS per i rispettivi gradi di libertà

Somma dei quadrati e varianza totale

La somma dei quadrati dell'errore totale si calcola con la formula $SS_T = \sum_{n=1}^{N}(Y_i - \bar{Y_{..}})^2$ dove N è il numero totale di osservazioni e $\bar{Y_{..}}$ è la media totale.

I gradi di libertà della varianza totale sono $df_T = N - 1$.

La varianza totale è pari a $MS_T = SS_T/df_T$.

Somma dei quadrati e varianza residua

La somma dei quadrati dell'errore residuo si calcola con la formula $SS_R = \sum_{i=1}^{I}\sum_{j=1}^{J_i}(Y_{ij} - \bar{Y_{i.}})^2$ dove I sono i livelli della variabile indipendente, J_i il numero di osservazioni del livello i e $\bar{Y_{i.}}$ la media delle osservazioni per il livello i.

I gradi di libertà della varianza residua sono $df_R = N - I$.

La varianza residua è pari a $MS_R = SS_R/df_R$.

Somma dei quadrati e varianza spiegata

La somma dei quadrati del modello si calcola con $SS_M = \sum_{i=1}^{I}(\bar{Y_{i.}} - \bar{Y_{..}})^2 \cdot J_i$ Ovvero, si calcola la differenza fra la media del gruppo i e la media totale, la si eleva al quadrato, e la si moltiplica per il numero di osservazioni di quel gruppo.
I gradi di libertà della varianza del modello sono $df_M = I - 1$.

La varianza spiegata è pari a $MS_M = SS_M/df_M$.

Identità principale dell'ANOVA

Proprio come per il modello di regressione lineare, $SS_T = SS_M + SS_R$. Questa uguaglianza viene definita *identità principale dell'ANOVA*.

La significatività del rapporto fra la variabile indipendente e quella dipendente viene misurata mettendo a rapporto la varianza spiegata dal modello con la varianza residua: $F = MS_M/MS_R$.

Distribuzione dell'errore, inferenza

Per introdurre il calcolo dell'analisi della Varianza, usiamo la consueta sequenza logica

- identificazione di una misura della relazione
- identificazione di una popolazione virtuale
- estrazione casuale di $k \cdot I$ campioni; calcolo della misura per ogni estrazione, e salvataggio nel vettore delle misure
- osservazione della distribuzione delle misure generate
- calcolo del p-value attraverso il confronto con il vettore delle misure
- identificazione di una distribuzione teorica che, previo opportuna trasformaizone, mappa quella osservata
- calcolo del p-value utilizzando la probabilità della distribuzione teorica identificata
- utilizzo della funzione di R

La simulazione

Per la nostra simulazione, immaginiamo un disegno sperimentale bivariato, dove la variabile indipendente ha tre livelli.

- La misura della relazione è la statistica F identificata sopra;
- generiamo per *k* volte tre campioni di numerosità *m*, con stessa media e deviazione standard (media e deviazione standard sono arbitrarie);
- calcoliamo $SS_T, SS_R, SS_M, df_T, df_R, df_M, MS_T, MS_R, MS_M$
- Calcoliamo la statistica $F = MS_M/MS_R$, e la salviamo nel vettore delle misure.

```
k<- 10000
distribuzione <- vector ("numeric",k)

for (i in 1:k) {
n <- 60
osservazioni <- rnorm (n,100,6)
osservazioniA <- osservazioni [1:20]
osservazioniB <- osservazioni [21:40]
osservazioniC <- osservazioni [41:60]
meanA <- mean(osservazioniA)
meanB <- mean(osservazioniB)
meanC <- mean(osservazioniC)
meanTot <- mean (osservazioni)
SSRA <- sum ((osservazioniA-meanA)^2)
```

```
SSRB <- sum ((osservazioniB-meanB)^2)
SSRC <- sum ((osservazioniC-meanC)^2)
SSR <- SSRA + SSRB + SSRC
SSMA <- 20 * (meanA-meanTot)^2
SSMB <- 20 * (meanB-meanTot)^2
SSMC <- 20 * (meanC-meanTot)^2
SSM <- SSMA + SSMB + SSMC
SST <- sum((osservazioni-meanTot)^2)
MSM <- SSM/(3-1)
MSR <- SSR/(60-3)
Fvalue <- MSM/MSR
distribuzione [i] <- Fvalue
}
```

La distribuzione Fisher-Snedecor

```
hist(distribuzione,probability=TRUE)
xrange <- seq (min(distribuzione),max(distribuzione), by=.2)
lines (xrange,df(xrange, 2, 57))
```

Nella figura 1.1, l'istogramma rappresenta la distribuzione dell'errore di campionamento ottenuto con la simulazione.

La linea sovrapposta all'istogramma rappresenta la distribuzione teorica F di Fisher-Snedecor.

Similmente alla distribuzione t di Student e alla distribuzione χ^2, anche la F è una famiglia di distribuzioni, che variano a seconda dei gradi di libertà.

La distribuzione F, però, varia in base a due gradi di libertà. Nell'esempio della simulazione, la linea corrisponde alla distribuzione F (2,57), ovvero ai gradi di libertà della varianza spiegata e della varianza residua.

R: calcolo di F value

Mostriamo il calcolo della statistica F con R, calcolando le tre medie, la media totale, SS_R, SS_M, SS_T, MS_R, MS_M, ed infine F.

```
osservazioniA <- zonaA
osservazioniB <- zonaB
osservazioniC <- zonaC
osservazioni <- zone
meanA <- mean(osservazioniA)
meanB <- mean(osservazioniB)
meanC <- mean(osservazioniC)
meanTot <- mean (osservazioni)
SSRA <- sum ((osservazioniA-meanA)^2)
SSRB <- sum ((osservazioniB-meanB)^2)
SSRC <- sum ((osservazioniC-meanC)^2)
SSR <- SSRA + SSRB + SSRC
SSMA <- 20 * (meanA-meanTot)^2
```

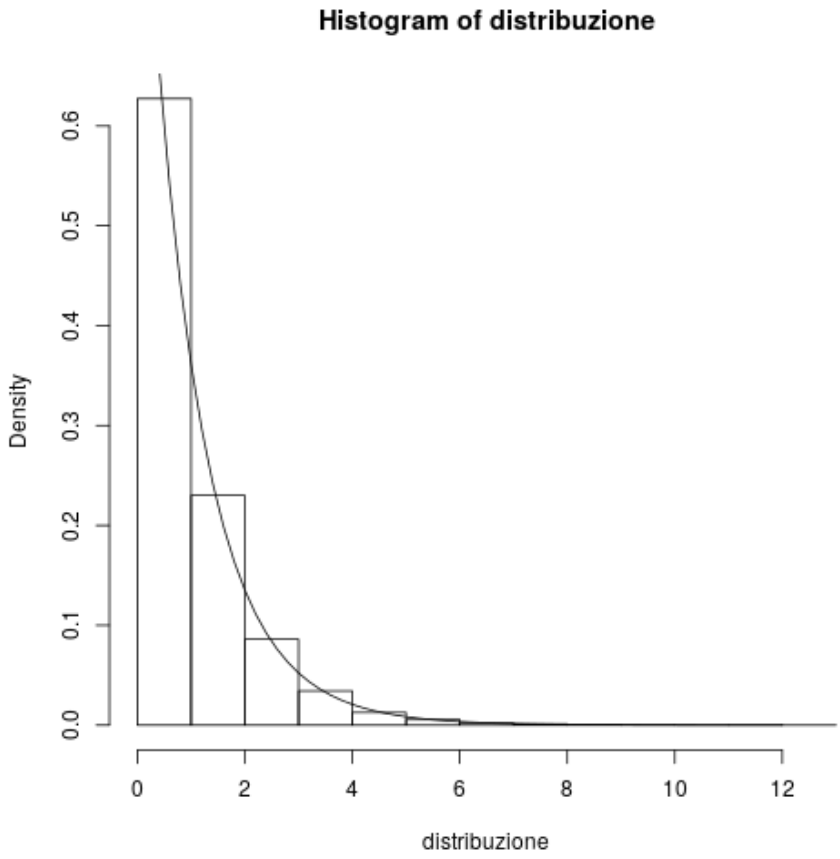


Figura 14.2: plot of chunk unnamed-chunk-4

```
SSMB <- 20 * (meanB-meanTot)^2
SSMC <- 20 * (meanC-meanTot)^2
SSM <- SSMA + SSMB + SSMC
SST <- sum((osservazioni-meanTot)^2)
MSR <- SSR/(n-3)
MSM <- SSM/(3-1)
F_Affitti <- MSM/MSR
c(SSM,SSR,SST)
```

```
## [1]  58463.33  55870.00 114333.33
```

```
c(MSM,MSR)
```

```
## [1] 29231.6667   980.1754
```

```
F_Affitti
```

```
## [1] 29.82289
```

F è dunque pari a 29.8228924. Ora, possiamo calcolare il p-value nel modo consueto, confrontando la posizione di questa statistica con la distribuzione calcolata prima.

```
# utilizzo la distribuzione simulata
p_value_empirica_1 <- 1-rank (c(F_Affitti,distribuzione))[1]/(k+1)
p_value_empirica_1
```

```
## [1] 0
```

Ora, calcoliamo il p-value usando la funzione `pf`.

```
# utilizzo la distribuzione statistica
p_value_F_1 <- 1 - pf (F_Affitti,2,57)
p_value_F_1
```

```
## [1] 1.369826e-09
```

Il risultato è sostanzialmente simile.

R: uso di aov

R mette a disposizione, per il calcolo dell'analisi della varianza, la funzione $aov(y\ x)$, dove y è la variabile dipendente, numerica, e x è il fattore.

```
aovAffitti <- aov(prezzo~quartiere,data=affitti)
summary(aovAffitti)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## quartiere    2  58463   29232   29.82 1.37e-09 ***
## Residuals   57  55870     980
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Leggere l'output

utilizzando la funzione `summary` su `aov` è possibile avere il dettaglio dei risultati dell'analisi. Nel caso di una analisi ad una via, avremo una tabella con due righe. La

seconda riga calcola i gradi di libertà, la somma dei quadrati, e la media dei quadrati dei residui (ovvero df_R, SS_R, MS_R). La prima riga calcola i gradi di libertà, la somma dei quadrati, e la media dei quadrati del modello: df_M, SS_M, MS_M); Inoltre, calcola $F = MS_M/MS_R$; infine, calcola il p-value. I codici simbolici (gli asterischi) ci suggeriscono la significatività: *** significa che $p - value < 0.01$.

Anova a due vie

Due variabili indipendenti

L'analisi della varianza che abbiamo introdotto, può essere estesa anche ai casi in cui le variabili indipendenti sono più di una.

Nell'analisi della varianza a due vie, ad esempio, si indaga la relazione fra due variabili indipendenti, entrambe categoriali, ed una variabile dipendente, quantitativa.

In questa sezione analizziamo la circostanza in cui le variabili indipendenti sono due, ma la logica rimane la stessa anche nelle circostanze in cui le variabili indipendenti sono più di due.

Le ipotesi

Nell'Anova a due vie, le domande che il ricercatore si pone sono tre:

- La prima delle variabili indipendenti, influisce significativamente sulla variabile dipendente?
- La seconda delle variabili indipendenti, influisce significativamente sulla variabile dipendente?
- Vi è una interazione fra le due variabili?

Un esempio: antidepressivi e attività aerobica

Introduciamo l'analisi della varianza a due vie con un esempio. Dei ricercatori sono interessati ad analizzare l'influenza dell'attività aerobica e di un tipo di farmaci antidepressivi (ad esempio, gli RSSI, gli inibitori del reuptake della serotonina) sul tono dell'umore di pazienti con diagnosi di depressione maggiore.

Decidono pertanto di selezionare 120 pazienti con diagnosi di depressione, e di assegnarli casualmente a 4 gruppi sperimentali, in un disegno 2x2.

I fattori

Un fattore è dunque l'antidepressivo: a 60 pazienti verrà somministrato, per 30 giorni, una dose di RSSI, mentre agli altri 60 verrà somministrato, per lo stesso periodo, un placebo.

L'altro fattore, l'attività aerobica: a 60 pazienti verrà chiesto di fare 20 minuti di attività aerobica due volte al giorno per 30 giorni. Agli altri 60 pazienti verrà chiesto di fare una attività non aerobica, di controllo, per lo stesso periodo di tempo.

I gruppi sperimentali

Avremo dunque 30 pazienti con placebo e attività non aerobica, 30 con placebo e attività aerobica, 30 con farmaco e attività non aerobica, 30 con farmaco e attività aerobica.

Alla fine dei 30 giorni, verrà somministrato un questionario (ad esempio il BDI, Beck depression inventory), per valutare il loro tono dell'umore alla fine del trattamento.

Il calcolo

Somma dei quadrati totale e residua

La somma dei quadrati totale è identico al caso dell'anova ad una via: si somma il quadrato della differenza di ogni osservazione con la media generale. Per calcolare la varianza totale si divide il tutto per i gradi di libertà della varianza totale, pari a n-1.

Anche la somma dei quadrati residui è identico al caso dell'anova ad una via: per ogni condizione sperimentale, si sommano i quadrati delle differenze fra i valori osservati e la media di quel gruppo, si sommano i valori così ottenuti da ogni gruppo. Di nuovo, la varianza è data dalla somma dei quadrati divisa per i gradi di libertà

Somma dei quadrati dei fattori

Il calcolo di SS_A: per ogni livello della variabile A, si calcola la media delle osservazioni di quel livello.

Si calcola il quadrato della differenza fra questa media e la media generale.

Si moltiplica questo risultato per il numero di osservazioni del livello.

Alla fine, si sommano i valori ottenuti per ognuno dei livelli. Si dividono per i gradi di libertà (pari al numero di livelli meno uno) per ottenere la varianza MS_A

In pratica, nel calcolare SS_A si fa come se il fattore B non esistesse. Lo stesso procedimento viene usato per calcolare la varianza spiegata dal fattore B.

Somma dei quadrati dell'interazione

Nel caso dell'anova ad una via, la varianza totale era pari alla somma della varianza residua e della varianza spiegata dall'unica variabile indipendente.

Nel caso dell'anova a due vie, però, la somma di varianza residua, varianza spiegata da A e varianza spiegata da B sarà minore della varianza totale.

La differenza è data dalla varianza spiegata dall'interazione fra i due fattori A e B. Più forte è l'interazione fra le due variabili indipendenti, più alta sarà la varianza spiegata dall'interazione (e dunque maggiore sarà la differenza fra la somma delle varianze residue, di A, di B e la varianza totale).

L'interazione fra le variabili indipendenti

Per introdurre il calcolo della varianza spiegata dall'interazione fra A e B, può essere utile riprendere il concetto di frequenza attesa introdotta nel test del χ^2. Anche in quel caso si trattava di valutare l'interazione fra due variabili categoriali. La differenza è che mentre nel χ^2 si misurano le frequenze, in questo caso la misura è data da una variabile quantitativa.

In maniera simile al χ^2, però, è possibile, conoscendo le medie marginali dei livelli di A e B, costruire una tabella delle medie attese, che costituisce il caso perfetto di assenza di interazione fra i due fattori. Questa tabella costituisce il caso ottimale di accettazione dell'ipotesi nulla relativa all'interazione fra le due variabili.

L'esempio: calcolo delle somme dei quadrati

Per esemplificare, torniamo all'esempio di un disegno 2x2. Quello che vogliamo fare è creare una tabella 2x2 delle medie attese.

Il primo passaggio, è calcolare le medie marginali per ogni livello della variabile A, e dunque la media di A1 e la media di A2.

Il secondo passaggio, è calcolare le medie marginali per i livelli di B, e dunque la media di B1 e la media di B2.

Infine, per le quattro celle [1,1] [1,2] [2,1] e [2,2], calcolare la media attesa.

La media attesa della cella [1,1] è pari alla media generale + la differenza fra A1 e la media generale e la differenza fra B1 e la media generale. Dunque, A1 + B1 - media.

Calcolo delle somme dell'interazione

SS_{Int} si basa sul quadrato delle differenze fra la tabella delle medie attese e la tabella delle medie osservate, moltiplicata per il numero di osservazioni per gruppo.

Dunque, più la tabella delle medie osservate è simile alla tabelle delle medie attese, minore è l'interazione fra le due variabili indipendenti, e dunque minore è la varianza spiegata dall'interazione.

Viceversa, maggiore è la differenza, maggiore l'interazione, maggiore la varianza spiegata dall'interazione.

Il calcolo, formalizzazione

Somma dei quadrati e varianza totale

La somma dei quadrati dell'errore totale si calcola con la formula $SS_T = \sum_{n=1}^{N}(Y_i - \bar{Y_{...}})^2$ dove N è il numero totale di osservazioni e $\bar{Y_{...}}$ è la media totale.

I gradi di libertà della varianza totale sono $df_T = N - 1$.

La varianza totale è pari a $MS_T = SS_T/df_T$.

Somma dei quadrati e varianza residua

La somma dei quadrati dell'errore residuo si calcola con la formula $SS_R = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}(Y_{ijk} - \bar{Y_{ij.}})^2$ dove I sono i livelli di A, J i livelli di B, K il numero di osservazioni per ogni livello e $\bar{Y_{ij.}}$ la media delle osservazioni per il gruppo ij.

I gradi di libertà della varianza residua sono $df_T = N - I * J$.

La varianza residua è pari a $MS_R = SS_R/df_R$.

Somma dei quadrati e varianza spiegata

La somma dei quadrati del modello si calcola con $SS_A = K \cdot J \cdot \sum_{i=1}^{I}(\bar{Y_{i..}} - \bar{Y_{...}})^2$ $SS_B = K \cdot I \cdot \sum_{j=1}^{J}(\bar{Y_{.j.}} - \bar{Y_{...}})^2$ I gradi di libertà della varianza del modello sono $df_A = I - 1, df_B = J - 1$.

Le varianze spiegate sono $MS_A = SS_A/df_A$, $MS_B = SS_B/df_B$.

Somma dei quadrati e varianza dell'interazione

$SS_{int} = K \cdot \sum_{i=1}^{I}\sum_{j=1}^{J}(\bar{Y_{ij.}} - \bar{Y_{i..}} - \bar{Y_{.j.}} + \bar{Y_{...}})^2$ ovvero la media osservata meno la media marginale di A_i, meno la media marginale di B_j, più la media totale.

I gradi di libertà della varianza dell'interazione sono $df_{int} = (I - 1) \cdot (J - 1)$.

La varianza dell'interazione è $MS_{int} = SS_{int}/df_{int}$.

Le ipotesi inferenziali

Le ipotesi inferenziali sono tre:

- H_0^A: l'influenza della variabile A sulla variabile dipendente non è significativa
- H_0^B: l'influenza della variabile B sulla variabile dipendente non è significativa
- H_0^{AB}: l'interazione fra A e B non è significativa.

Ipotesi inferenziali e F

Per valutare le tre ipotesi inferenziali, vengono calcolati i tre rapporti:

- $F_A = MS_A/MS_R$;
- $F_B = MS_B/MS_R$;
- $F_inte = MS_{int}/MS_R$;

Ognuno dei rapporti viene confrontato con la distribuzione F di Fisher-Snedecor.

Modello lineare

In maniera simile alla regressione lineare, anche l'analisi della varianza (sia a una che a due vie) può essere rappresentata attraverso un modello lineare. Il modello generale per l'anova a due vie è $Y_{ijk} = \mu + \alpha_i + \beta_j + \delta_{ij} + \epsilon_{ijk}, i = 1...I, j = 1...J, k = 1...K$ dove I è il numero di livelli del fattore A, J il numero di livelli del fattore B, K il numero di osservazioni per ogni gruppo.

μ corrisponde alla media totale di tutte le osservazioni.

α_i corrisponde allo scostamento dalla media totale del livello A_i

β_j corrisponde allo scostamento dalla media totale del livello B_j

δ_{ij} corrisponde alla differenza fra la media del campione osservata e quella attesa in base all'ipotesi di non interazione fra A e B.

ϵ_{ijk} è la componente di errore, ovvero la differenza fra il valore atteso dal modello e il valore osservato.

L'esempio dei trattamenti per la depressione

Torniamo all'esempio dei trattamenti per la depressione, e generiamo tre diversi scenari.

Primo scenario

```
punteggioBase <- 22
effettoFarmaco <- 8
effettoAerobico <- 7
# gl genera un fattore di n livelli, k volte
farmaco  <- gl (2,60,labels=c("placebo","farmaco"))
aerobica <- gl (2,30,120,labels=c("non aerobico","aerobico"))

placeboNA <- rnorm (30,punteggioBase,8)
placeboA  <- rnorm (30,punteggioBase+effettoAerobico,8)
farmacoNA <- rnorm (30,punteggioBase+effettoFarmaco,8)
farmacoA  <- rnorm (30,punteggioBase+effettoFarmaco+effettoAerobico,8)
punteggi  <- c(placeboNA,placeboA,farmacoNA,farmacoA)
expDep1 <- data.frame (punteggi=punteggi,farmaco=farmaco,aerobica=aerobica
medie.osservate <- tapply (expDep1$punteggi,list(expDep1$farmaco,expDep1$a
medie.osservate
```

```
##         non aerobico aerobico
## placebo     22.26042 31.51356
## farmaco     32.23289 37.00520
```

Rappresentiamo

```
interaction.plot (expDep1$farmaco,expDep1$aerobica,expDep1$punteggi)
```

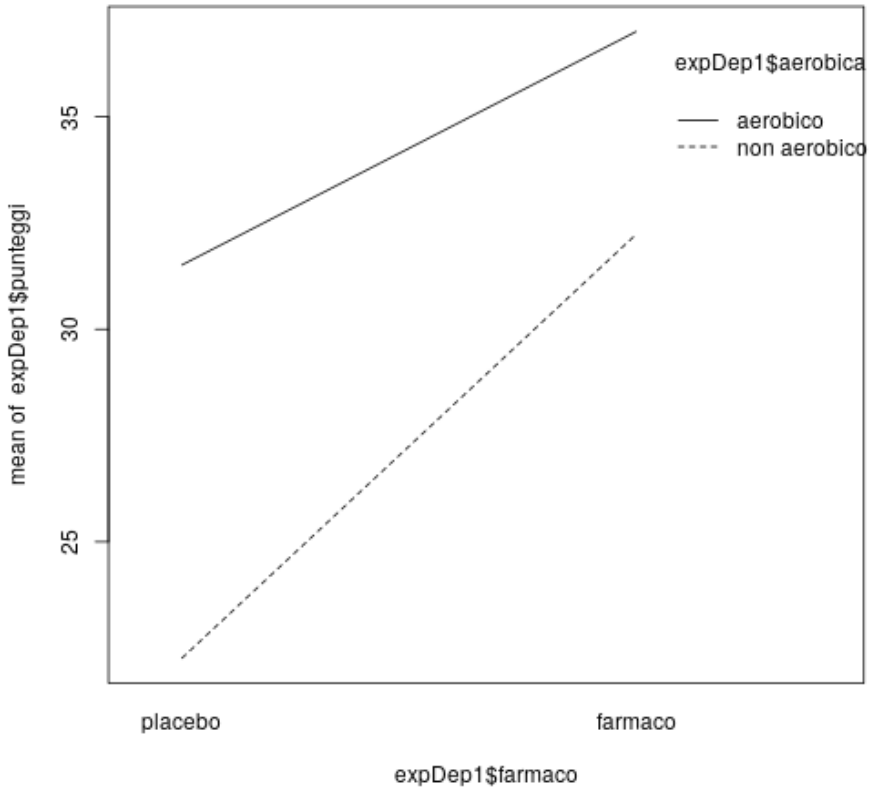


Figura 14.3: plot of chunk unnamed-chunk-9

La funzione `interaction.plot` rappresenta la media della variabile dipendente (numerica) nelle diverse condizioni sperimentali in un disegno con due variabili indipendenti categoriche.

Dal grafico si intuisce che entrambe le variabili indipendenti hanno un effetto sulla variabile dipendente, ma che probabilmente non vi sarà interazione fra le due.

Applichiamo la funzione `aov`

```
aovScenario1 <- aov(punteggi~farmaco+aerobica+farmaco:aerobica,data = expD
(summaryAov1 <- summary(aovScenario1))
```

```
##                   Df Sum Sq Mean Sq F value   Pr(>F)
## farmaco            1   1794  1793.5  30.149 2.39e-07 ***
## aerobica           1   1475  1475.3  24.800 2.24e-06 ***
## farmaco:aerobica   1    151   150.6   2.531    0.114
## Residuals        116   6901    59.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Invocando `summary(aov())` otteniamo una tabella che riassume i gradi di libertà, la somma dei quadrati, la media, la statistica F ed il p-value del fattore `farmaco`, del fattore `aerobica`, dell'interazione `farmaco:aerobica` e dei residui.

Dal modello dell'analisi della varianza possiamo dedurre che vi è una influenza significativa sia del primo fattore (farmaco) che del secondo (attività aerobica); non vi è, però, interazione significativa fra i due fattori.

L'esempio, secondo scenario

```
farmacoA  <- rnorm (30,punteggioBase+effettoFarmaco+effettoAerobico+10,8)
punteggi  <- c(placeboNA,placeboA,farmacoNA,farmacoA)
expDep2 <- data.frame (punteggi=punteggi,farmaco=farmaco,aerobica=aerobica
summary (expDep2)
```

```
##     punteggi         farmaco           aerobica
##  Min.   : 7.344   placebo:60   non aerobico:60
##  1st Qu.:25.465   farmaco:60   aerobico    :60
##  Median :33.624
##  Mean   :33.106
##  3rd Qu.:40.506
##  Max.   :70.111
```

```
interaction.plot (expDep2$farmaco,expDep2$aerobica,expDep2$punteggi)
```

```
medie.osservate <- tapply (expDep2$punteggi,list(expDep2$farmaco,expDep2$a
medie.osservate
```

```
##         non aerobico aerobico
## placebo     22.26042 31.51356
## farmaco     32.23289 46.41744
```

```
aov_sc2 <- aov(punteggi~farmaco+aerobica+farmaco:aerobica)
risultati <- unlist(summary(aov_sc2))
summary(aov_sc2)
```

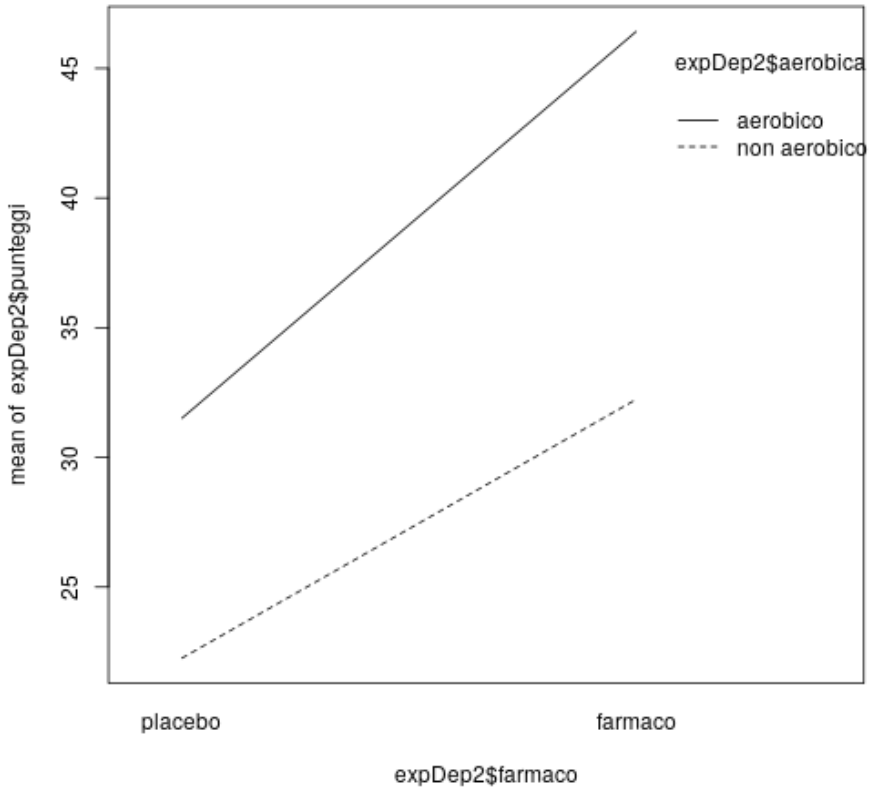


Figura 14.4: plot of chunk dep_sc2

```
##                    Df Sum Sq Mean Sq F value   Pr(>F)
## farmaco             1   4641    4641  78.701 1.04e-14 ***
## aerobica            1   4120    4120  69.861 1.59e-13 ***
## farmaco:aerobica    1    182     182   3.093   0.0813 .
## Residuals         116   6841      59
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nel secondo scenario manipoliamo la media della variabile dipendente nella condizione "farmaco + aerobica".

In questo caso

- non rifiutiamo l'ipotesi nulla H_0^A, in quanto l'effetto del farmaco diventa non significativo: $F = 78.7009615$, $p =< 0.001$;
- rifiutiamo l'ipotesi nulla H_0^B, in quanto l'effetto dell'attività aerobica è significativo: $F = 69.8612262$, $p =< 0.001$;
- rifiutiamo l'ipotesi nulla H_0^{AB}, in quanto è significativa l'interazione fra i due fattori: $F = 3.0927855$, $p = 0.081$;

Confronti multipli

Confronti multipli ed errore

L'analisi della varianza ci permette di verificare se le differenze fra le medie di tre o più campioni sono da attribuire all'errore campionario, o se sono significative.

Una volta rifiutata l'ipotesi nulla, però, resta da determinare quali differenze sono significative. L'analisi della varianza, infatti, ci dice se vi è almeno una coppia di gruppi la cui differenza è significativa, ma non ci dice quali differenze lo sono.

Per poter determinare quali differenze sono significative, diventa necessario confrontare i gruppi a due a due.

Come abbiamo visto all'inizio del capitolo, si potrebbe decidere di utilizzare, per confrontare a due a due i diversi gruppi, il t-test. Ma, come abbiamo già accennato, applicare ripetutamente il t-test aumenta la probabilità di incorrere in un errore del primo tipo.

Diventa dunque necessario adottare dei test di confronti multipli capaci di mantenere sotto controllo l'errore del I tipo.

La correzione di Bonferroni

Un possibile approccio, finalizzato a controllare la proliferazione dell'errore del I tipo, è quello di adottare la correzione di Bonferroni, che consiste nel dividere il valore α per il numero di confronti effettuati (o, in maniera corrispondente, moltiplicare il p-value per il numero di confronti).

Se, ad esempio, dobbiamo confrontare le medie di 4 gruppi e decidiamo per un valore $\alpha = 0.05$, in base alla correzione di Bonferroni dovremo considerare significativi soltanto quei confronti il cui p-value sia inferiore a $0.05/6 = 0.0083$ (in quanto i confronti previsti sono 6).

Il problema di questo metodo è che tende ad essere eccessivamente conservativo.

Il test di Tukey

Un metodo di confronto multiplo meno conservativo è il test di Tukey.

Anche attraverso il test di Tukey è possibile mantenere l'errore di tipo I entro un predeterminato valore di α (generalmente pari a 0.05).

Il test di Tukey permette di *correggere* il p-value in base al numero di confronti che vengono effettuati nel confronto multiplo, senza però penalizzare eccessivamente la statistica.

Per correggere l'errore, il test di Tukey confronta la statistica calcolata con la distribuzione *studentized range*

Il test di Tukey: calcolo

Per calcolare la significatività della differenza fra due gruppi con il metodo di Tukey si utilizza il seguente algoritmo:

- si calcola l'errore standard, con la formula $SE = \sqrt{(MS_R/n)}$ dove n è il numero di osservazioni per gruppo. Nel caso di gruppi con numerosità diversa, la formula diventa $SE = \sqrt{\frac{MS_R}{2} \cdot (\frac{1}{n_a} + \frac{1}{n_b})}$ dove n_a e n_b sono la numerosità del primo e del secondo gruppo
- si calcola la statistica $Q = \frac{|Y_a - Y_b|}{SE}$
- si calcola il p-value, usando la funzione `ptukey (Q,k,Df_R)` dove k è il numero di confronti effettuati, e Df_R i gradi di libertà della varianza residua. La funzione ptukey calcola la probabilità sulla distribuzione studentized range.

Affitti: confronti multipli

Torniamo all'esempio degli affitti, e mostriamo il calcolo di uno dei confronti multipli con il test di Tukey. Calcoliamo le tre medie.

```
medieAffitti <- tapply(affitti$prezzo,affitti$quartiere,mean)
medieAffitti
```

```
##     A     B     C
## 546.0 564.5 619.5
```

Tukey: il calcolo di un confronto

Calcoliamo il conftonto fra le medie dei gruppi A e B.

```
confronto1 <- abs(medieAffitti[1] - medieAffitti[2])
SE <- sqrt (MSR/20)
Q <- confronto1/SE
p_value <- 1 -ptukey (Q,3,57)
c(confronto1,SE,Q,p_value)
```

```
##          A                     A          A
## 18.5000000  7.0006265  2.6426206  0.1571556
```

Il p-value è alto, e dunque la differenza fra i gruppi A e B non è significativa.

La funzione R TukeyHSD

La funzione di R per il calcolo del confronto con il metodo Tukey è `TukeyHSD`. Coerentemente con l'uso dei confronti multipli, la funzione si applica sul risultato della corrispondente analisi della varianza.

```
(confronti1 <- TukeyHSD(aovAffitti, ordered = TRUE))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##     factor levels have been ordered
##
## Fit: aov(formula = prezzo ~ quartiere, data = affitti)
##
## $quartiere
##     diff       lwr      upr     p adj
## B-A 18.5 -5.324457 42.32446 0.1571556
## C-A 73.5 49.675543 97.32446 0.0000000
## C-B 55.0 31.175543 78.82446 0.0000023
```

La funzione ritorna una tabella, con una riga per ogni confronto, dove vengono mostrate:

- la coppia confrontata (es, il confronto fra il gruppo B ed il gruppo A); l'ordine è tale che il gruppo con media più alta è davanti all'altro;
- la differenza (positiva) fra i due gruppi;
- l'intervallo di confidenza della differenza; ad esempio, nel secondo confronto (C-A), la differenza è di 73.5, l'intervallo di confidenza va da un minimo di 49.6755434 ad un massimo di 97.3244566. *p adj* è il p-value aggiustato, che nel confronto `C - A` è pari a <0.001; nell'esempio, i confronti C-A e C-B sono significativi, il confronto B-A no (0.16).

Analisi della Varianza: assunti

Come ogni approccio parametrico, anche l'analisi della varianza fa delle assunti:

- indipendenza delle osservazioni
- distribuzione normale degli errori
- omoschedasticità: la varianza dell'errore è costante
- gli errori sono fra loro indipendenti

Distribuzione degli errori

Si assume che gli errori abbiano una distribuzione normale, con media pari a 0, e varianza costante fra i gruppi. Per testare l'ipotesi di normalità, è possibile usare il test di Shapiro-Wilk sui residui del modello dell'analisi della varianza:

```
shapiro.test(aovAffitti$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  aovAffitti$residuals
## W = 0.99128, p-value = 0.9461
```

Per testare l'ipotesi di omoschedasticità, si può usare il test di Bartlett:

```
bartlett.test(prezzo ~ quartiere, data = affitti)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  prezzo by quartiere
## Bartlett's K-squared = 0.04306, df = 2, p-value = 0.9787
```

Test non parametrico

Vi sono circostanze in cui l'analisi della varianza non può essere applicata, in quanto vengono meno alcuni assunti o condizioni:

- non si può assumere la normalità della distribuzione degli errori
- il numero di osservazioni per ogni gruppo è minore di 10
- la variabile dipendente non è ad intervalli, ma ordinale

In questi casi è possibile applicare il test non parametrico di Kruskal-Wallis

Il test di Kruskal-Wallis

il test di Kruskal-Wallis è un'estensione del test di Wilcoxon, che abbiamo visto nel capitolo dedicato al t-test [@ChanWalmsley:1997]. Nel test di Kruskal-Wallis, la prima operazione da compiere è quella di trasformare i punteggi osservati nel loro rango. A questo punto, si applica la formula $K = (n-1)\frac{\sum_{i=1}^{I} n_i(\bar{r_{i.}} - \bar{r})^2}{\sum_{i=1}^{I}\sum_{j=1}^{n_i}(r_{ij} - \bar{r})^2}$ dove n_i è il numero di osservazioni nel gruppo i, r_{ij} è la posizione ordinale dell'osservazione j del gruppo i, N è il numero totale delle osservazioni, $\bar{r_{i.}}$ è la media dei rank del gruppo i.

Semplificazioni

L'equazione [eq:Kruskal] può in realtà essere semplificata, in quanto $\bar{r} = (N+1)/2$ e denominatore è pari a $(N-1)N(N+1)/12$, e dunque otteniamo

$K = \frac{12}{N(N+1)} \sum_{i=1}^{I} n_i(\bar{r_{i.}} - \frac{N+1}{2})^2$

La statistica K assume una distribuzione χ^2 con i-1 gradi di libertà.

R: la funzione kruskal.test

Applichiamo il test di *Kruskal-Wallis* al nostro esempio degli affitti.

```
(kt1 <- kruskal.test(prezzo ~ quartiere, data = affitti))
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  prezzo by quartiere
##  Kruskal-Wallis  chi-squared  =  31.32,  df  =  2,  p-
value = 1.581e-07
```

Leggere i risultati

La funzione restituisce la statistica, *Kruskal-Wallis chi-squared = 26.4763*; I gradi di libertà: *df = 2*; il *p-value = 1.781e-06.*

Conclusioni

Da fare.

Capitolo 15

Il processo di ricerca

Strutturare il processo di analisi

Introduzione

L'analisi dei dati si colloca, generalmente, in un progetto di ricerca.

La ricerca è un processo strutturato e sistematico di indagine di un fenomeno. Nel processo vengono utilizzate appropriate procedure, metodologie e tecniche di dimostrata validità all'interno di un quadro epistemologico e metodologico di riferimento. Nel processo si possono identificare alcuni passaggi:

- definizione di un problema o di una domanda di ricerca
- analisi della letteratura
- definizione della metodologia
- identificazione dei *parametri* e delle dimensioni da indagare per rispondere alla domanda di ricerca
- raccolta di dati
- analisi dei dati
- interpretazione dei risultati
- *disseminazione* dei risultati

L'analisi dei dati, dunque, è generalmente uno dei passaggi di un progetto di ricerca.

Negli ultimi anni, anche a causa del problema della non replicabilità di molti lavori, si parla molto dei concetti di replicabilità, riproducibilità e open science. Per mettere la cosa nella giusta prospettiva, è necessario considerare che replicabilità/riproducibilità e open science sono alcuni degli strumenti utili a massimizzare alcuni princìpi della ricerca scientifica:

- l'efficienza nel processo
- la correttezza dei risultati
- la solidità (robustness) dei risultati e delle inferenze
- la trasparenza del processo che ha portato ai risultati

Nello strutturare un processo di analisi dei dati, è opportuno adottare metodi e soluzioni capaci di garantire dei buoni standard qualitativi in base a questi princìpi.

L'efficienza

In questo contesto, per efficienza si intende la necessità di non sprecare ed anzi di ottimizzare l'utilizzo delle risorse necessarie per effettuare la ricerca:

- risorse economiche
- stutture e risorse tecniche (laboratori, macchinari, materiale)
- risorse umane (ricercatori, analisti, personale tecnico)

La *risorsa* più importante, nelle scienze mediche, sociali e psicologiche sono però i partecipanti (umani e non umani) coinvolti nella ricerca. Adottare processi capaci di massimizzare l'utilità della ricerca e l'efficienza nel coinvolgimento dei partecipanti, minimizzandone l'impatto negativo, è di primaria importanza e costituisce uno dei criteri etici della ricerca.

Correttezza

Il progetto di ricerca ed i suoi risultati sono corretti se si eliminano o si minimizzano i possibili errori in cui i ricercatori possono incorrere:

- errori nel modello teorico di riferimento
- errori metodologici che minano alcuni aspetti della validità della ricerca
- errori nella definizione delle variabili da studiare
- errori sistematici (bias) nel campionamento
- errori nella misura delle variabili
- errori sistematici (bias) nell'analisi dei dati
- errori nella sintesi e nella pubblicazione dei risultati.

Vi è una relazione fra correttezza e validità. Le tecniche di campionamento e l'analisi inferenziale ci permettono di minimizzare gli errori non sistematici di campionamento e di stima delle statistiche.

Solidità (robustness)

Il costrutto di solidità rende esplicito il legame fra riproducibilità e validità esterna. Nel senso più stretto, un risultato è robusto se può essere replicato nella stessa forma, con campioni estratti dalla stessa popolazione, anche in laboratori diversi (replicabilità).
Nel senso più ampio, un risultato è robusto se può essere replicato (e dunque generalizzato) in contesti in parte diversi, su popolazioni in parte diverse - o più ampie - e con paradigmi di ricerca diversi.

Trasparenza

Correttezza e trasparenza dei protocolli sono criteri che permettono di aumentare la credibilità della ricerca, ed alcuni aspetti della sua validità.

Per soddisfare il criterio di trasparenza, è opportuno rendere espliciti e disponibili:

- i finanziamenti alla ricerca e gli eventuali conflitti di interesse
- il protocollo di ricerca
- i dati
  - i dati grezzi raccolti
  - i dati puliti
- il processo di analisi dei dati
- i risultati del processo di analisi
- i report e le pubblicazioni
- definire e rendere pubblici, a priori, i protocolli di ricerca, le ipotesi, i paradigmi e gli esperimenti che si intende intraprendere;

Alcuni di questi requisiti sono finalizzati a contrastare problemi quali il *publication bias*, il *p-hacking* ed in genere quelle che in letteratura vengono definite *Questionable research practices* (Measuring the Prevalence of Questionable Research Practices With Incentives for Truth Telling).

Protocollo di ricerca

Soprattutto nei progetti di ricerca, è opportuno definire a priori un chiaro protocollo di ricerca:

WHO | Recommended format for a Research Protocol

- sommario del protocollo
- informazioni generali
  - titolo del progetto, id
  - budget e costi previsti
  - finanziamenti
  - ricercatori coinvolti
  - laboratori coinvolti
- motivazioni della ricerca
- informazioni di contesto
- bibliografia
- scopi e obiettivi della ricerca
- design sperimentale
- metodologia
- sicurezza e salvaguardia dei partecipanti
  - gestione del consenso informato
  - gestione della privacy
- gestione dei dati
- analisi statistica
- quality Assurance
- risultati attesi
- Dissemination of Results and Publication Policy
- collegamenti con altri progetti

Pratiche scorrette

@john2012measuring citano i dieci peccati capitali nella riceca (questionable research practices)

1. Non riportare tutte le misure dipendenti di uno studio
2. Decidere di aumentare la numerosità del campione per provare a rendere significativi i dati
3. Nella pubblicazione dei risultati, non riportare tutte le condizioni sperimentali
4. Decidere di diminuire la numerosità del campione una volta raggiunta la significatività dei riusltati
5. Arrotondare il p-value per farlo apparire significativo
6. Nella pubblicazione, riportare solo gli studi che confermano le ipotesi sperimentali e non quelle i cui risultati non sono significativi

7. Escludere selettivamente dei dati per *migliorare* i risultati
8. Aggiustare le ipotesi in base ai risultati
9. Riportare in maniera non corretta l'effetto delle variabili estranee al disegno di ricerca (ad esempio le variabili demografiche)
10. Falsificare i dati

@begley2013reproducibility Citano sei sintomi che lasciano intendere che gli standard qualitativi non siano stati rispettati. Fra i criteri che andrebbero rispettati:

- Utilizzare, quando possibile e opportuno, del doppio cieco nell'esperimento
- Replicare gli esperimenti più importanti (possibilmente in laboratori diversi)
- Pubblicare tutti i risultati, anche quelli negativi
- Applicare un'appropriata analisi statistica

Replicabilità e riproducibilità

In letteratura spesso i due termini sono utilizzati come sinonimi. Altri autori distinguono due scenari diversi:

- *Replicabilità*: raccogli nuovi dati con lo stesso paradigma sperimentale e lo stesso protocollo
- *Riproducibilità*: possibilità di analizzare i dati raccolti nell'esperimento originale - e riprodurne i risultati

Fonti:

- Replicability vs Reproducibility - Replicability Research Group
- Reproducibility vs. Replication
- @schloss2018identifying

Per permettere la replicabilità di una ricerca è necessario mettere a disposizione il protocollo ed eventualmente il software ed il materiale per la raccolta e l'elaborazione dei dati.
Per permettere la riproducibilità è necessario mettere a disposizione i dati ed il software utilizzato per l'analisi.

L'approccio open

Recentemente, sta prendendo piede nel mondo della ricerca l'approccio *open*: open research, open data, strumenti e software open source: rendere disponibili i protocolli di ricerca, i dati raccolti, gli script utilizzati per fare le analisi.

I vantaggi della open science sono molteplici:

- *efficienza*: rendere disponibili dati e codice permette ad altri ricercatori e centri di ricerca di fare tesoro del lavoro fatto, e semplifica la collaborazione fra ricercatori e centri di ricerca diversi;
- *trasparenza*: rendere pubblici protocolli, dati, script e software è una delle migliori garanzie di trasparenza;
- *correttezza*: l'approccio open permette una più approfondita peer review: non solo dell'articolo da pubblicare, ma anche dei dati, dei processi, dell'analisi

- *replicabilità*: diventa più semplice replicare una ricerca - o parte di essa - avendo a disposizione il materiale.

Automazione

Nell'elaborazione ed analisi dei dati, una delle raccomandazioni più importanti riguarda la possibilità di automatizzare il più possibile il processo, riducendo le attività che il ricercatore e l'analista devono fare a mano.
Automatizzare il processo garantisce numerosi vantaggi:

- *efficienza*: una volta automatizzato, il processo diventa molto più veloce; questo vale però solo per quelle attività che tendono a ripetersi nei diversi progetti e nelle diverse analisi;
- *trasparenza*: se il codice utilizzato per processare ed analizzare i dati viene condiviso (approccio *open*), altri ricercatori avranno la possibilità di verificarlo ed eventualmente correggerlo, e di riprodurre l'elaborazione e l'analisi dei dati;
- *correttezza*: gli script utilizzati per automatizzare il processo vanno attentamente verificati, per evitare errori di elaborazione ed analisi; vengono però minimizzate le possibilità di errore umano insito nelle operazioni fatte *a mano*;
- *riproducibilità*: se l'analisi è automatizzata, è più facile da riprodurre, ed il protocollo di ricerca è più facile da replicare

Strutturare il processo di ricerca

- modularità
- automazione
- standardizzazione (interna, esterna)
- sicurezza (dei dati)
- trasparenza

Sicurezza dei dati

- salvare i dati grezzi su più supporti distinti: hd esterno, in cloud

Trasparenza

- scrivi codice facile da leggere (*write code for humans, write data for computer*), sia per te stesso/a (quando andrai a rileggere il codice) che per gli altri (collaborazione, replicabilità)
  - documenta e commenta ampiamente il codice; usa il codice come documentazione
- salva i dati, compresi i dati e i risultati intermedi, in formati aperti e standard
  - salva gli oggetti grafici con cui crei i plot
- rendi esplicito il legame fra l'analisi dei dati e la loro interpretazione (ad esempio attraverso l'uso di rMarkdown)

Standardizzazione

- usa pacchetti consolidati e testati

Modularità

- crea un'unica directory con tutto il materiale (dati, codice, testi)

- adotta degli standard per organizzare il materiale, ad esempio salvando in sotto-directory separate i dati (grezzi ed elaborati), il codice, le parti testuali / le parti in rMarkdown, i risultati
- documenta le versioni del software (in R, usando `session_info()`)
- opzionale: crea una macchina virtuale
- in rStudio, creare un progetto
- negli script usa percorsi relativi per caricare e salvare dati e figure
- trasforma gli script che usi più di frequente in funzioni (se possibile), moduli, eventualmente pacchetti

Tracciabilità

- tieni nota dell'origine e della versione dei dati
- usa un sistema di versioning (ad esempio git) per tenere traccia dell'evoluzione degli script

Replicabilità

- fa in modo che statistiche, tabelle e figure siano il risultato di script di calcolo
- per ogni risultato, tieni traccia di come è stato prodotto
- se usi delle funzioni di randomizzazione, stabilisci e salva il *random seed*
- automazione: evita (o minimizza) processi di pulizia e trasformazione manuali dei dati
- rendi pubblici i dati grezzi e puliti, gli script, i risultati.

Fonti:

- @sandve2013ten
- Reproducible Research Using RMarkdown
- Best Practice for good documented reproducible analysis - RStudio Community
- Tools for Reproducible Research - Organizing projects; exploratory data analysis

Conclusioni