Note di

STATISTICA

ED

INFERENZA IN R

CON ESERCIZI SVOLTI
ED OSSERVAZIONI

Indice

Capitolo 1

Rappresentazione dei fati

1.1 Tabelle

La rappresentazione dei dati è il primo passo da svolgere per dare inizio ad uno studio statistico. Dato un campione, vogliamo trovare un buon modo per rappresentare visivamente le sue proprietà. Possiamo farlo con tabelle e grafici.

Esempio 1. Si prenda la variabile aleatoria numerica discreta data dai valori rappresentati in tabella:

0	2	1	4	3	1	2	2	3	8
0	2	1	3	3	1	3	2	2	5
5	4	4	2	4	3	3	1	1	2
4	4	2	3	3	3	3	3	5	2

Questi casi possono essere descritti in prima analisi costruendo le *tabelle di distribuzione di frequenza.*

Classi	Frequenza	Frequenza relativa	Frequenza cumulativa
0	2	2/40	2/40
1	6	6/40	8/40
2	10	10/40	18/40
3	12	12/40	30/40
4	6	6/40	36/40
5	3	3/40	30/40
6	0	0	30/40
7	0	0	30/40
8	1	1/40	40/40

- la *frequenza* indica la quantità di volte che un valore appare nell'intero insieme dei dati
- la *frequenza relativa* si ottiene dividendo la frequenza per il numero totale di dati che compaiono nel campione
- la *frequenza cumulativa* si costruisce partendo dalle frequenze relative delle entrate precedenti e rappresenta il valore in quella classe della funzione di distribuzione empirica

$$\hat{F}_n(x)$$

In questo modo possiamo costruire una tabella per un insieme di dati discreto, ma come possiamo fare nel caso continuo?

Esempio 2. Stavolta prendiamo un vettore aleatorio continuo.
La costruzione di particolari sfere di metallo è soggetta ad un errore che fa variare il diametro della sfera prodotta. Di seguito sono riportati i risultati di un campione (espressi in centimetri).

2.08 1.72 1.90 2.11 1.82
2.04 2.04 1.82 2.04 2.07
1.79 1.86 1.80 1.91 1.84
1.86 1.80 1.85 2.08 2.03

I valori possibili sono palesemente infiniti, la precedente tecnica di rappresentazione diventa inutile; un buon modo è dividere in intervalli.
Il metodo di costruzione degli intervalli non è unico, e solitamente si sceglie a seconda del campione in studio. Intuitivamente potrei prendere intervalli che coprano insiemi del tipo $(x, x + 0.5]$, ma anche $(x, x + k]$ per ogni $k > 0$ reale; è bene porre attenzione alla scelta degli intervalli perché un intervallo abbastanza grosso potrebbe assorbire in sé tutti i dati rilevanti. Si immagini ad esempio di prendere nel nostro studio sulle sfere di metallo intervalli del tipo

$$(0, 4],\ (4, 8],\ \ldots$$

Questi sono palesemente inutili e renderebbero poco interessanti i risultati.
Un buon modo può essere dividere come segue.

Classi	Frequenza	Frequenza relativa	Frequenza cumulativa
(1.7, 1.75]	1	1/20	1/20
(1.75, 1.8]	3	3/20	4/20
(1.8, 1.85]	4	4/20	8/20
(1.85, 1.9]	3	3/20	11/20
(1.9, 1.95]	1	1/20	12/20
(1.95, 2.0]	0	0	12/20
(2.0, 2.05]	4	4/20	16/20
(2.05, 2.1]	3	3/20	19/20
(2.1, 2.15]	1	1/20	20/20
(2.15, 2.2]	0	0	20/20

Si noti infatti che, come osservato in precedenza, avrei potuto prendere intervalli di dimensioni diverse tra loro!

Definizione 1. Queste appena viste si dicono **tabelle di distribuzione in frequenza**, rappresentano per ogni classe i tre diversi aspetti della frequenza:

- *frequenza*
- *frequenza relativa*
- *frequenza cumulativa*

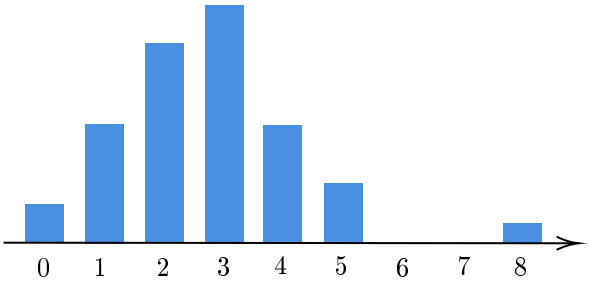
definiti in precedenza.

1.2 Grafici

1.2.1 Istogramma

Definizione 2 (**Istogramma**)**.** Un **istogramma** è una rappresentazione grafica di dati in cui ogni classe è rappresentata da un rettangolo dalle basi allineate rispetto un asse cartesiano, la frequenza di ogni classe è direttamente proporzionale all'*area del rettangolo* della classe stessa.

Esempio 3. Visualizziamo in un istogramma i dati del primo esempio.



Possiamo osservare che:

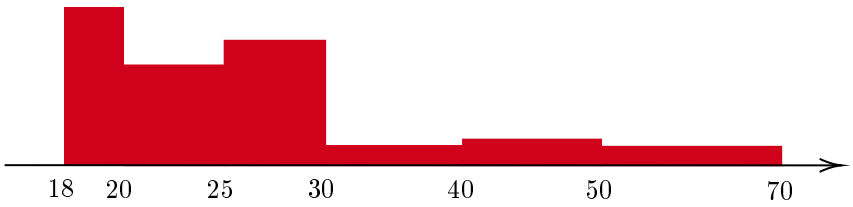
1. qui le classi erano semplicemente numeri interi, questo può trarre in confusione se applicato al caso di variabili continue: si deve indicare la classe (che nel nostro caso è appunto un valore intero)

2. non è riportata la scala, ma avendo tutti la stessa base possiamo subito ricavare la scala utilizzata dalle classi di frequenza unitaria dell'istogramma: quindi non ci sono problemi di interpretazione

Esempio 4. Dati sull'età di raggiungimento della patente raccolti nella stessa autoscuola in un certo periodo di tempo.

Classi	Centro	Frequenza	F. relativa	F. cumulativa
$[18, 20)$	19	8	8/60	8/60
$[20, 25)$	22.5	13	13/60	21/60
$[25, 30)$	27.5	16	16/60	38/60
$[30, 40)$	35	5	5/60	43/60
$[40, 50)$	45	6	6/60	49/60
$[50, 70)$	60	11	11/60	60/60

Avendo i rettangoli basi diverse (si vede dai dati che ci hanno fornito), stavolta il grafico avrà le classi dipendenti davvero dall'area. I rettangoli avranno quindi altezza dipendente dalla frequenza.
Il calcolo dell'altezza è una banale proporzione.



(per ovvi motivi di spazio i valori delle età non sono rappresentati in intervalli con proporzioni corrette, tuttavia le altezze dei grafici sono quelle in proporzione rispetto ai valori veri)

1.3 Indici di posizione

Lavoriamo sulla sintesi dei dati: le tabelle danno informazioni importanti, ma a volte a noi interessa un solo numero - o comunque un numero legato al problema. Ad esempio potrebbe interessarci una particolare media.[1]

Gli indici di posizione mostrano come è diviso il campione e permettono di coglierne la natura a prima vista, soprattutto se rappresentati in un grafico.

[1] oppure, più accuratamente, il valore del suo "stimatore campionario". La media di un campione chiaramente non è la media della variabile aleatoria che lo ha generato.

Definizione 3. Sono detti *indici di posizione*:

1. *Media campionaria*: è la media aritmetica dei dati dell'osservazione una volta scritti come campione $X = (X_1, \dots, X_n)$.
$$\overline{X} \coloneqq \frac{\sum_{i=1}^{n} X_i}{n}$$
Se i dati sono raggruppati in classi, si usa *il "valore centrale" di ogni classe.*
Si dice "campionaria" perché ovviamente dipende dal campione.

2. *Mediana*: l'elemento che "divide in due" il campione **ordinato**.

   - se n è dispari ($n = 2k+1$) ne è effettivamente l'elemento centrale x_{k+1}
   - se invece n è pari (quindi $n = 2k$ per un certo k) si prende la media dei due valori centrali, $\frac{x_k + x_{k+1}}{2}$

3. *Quantile*: generalizzazione della mediana, intuitivamente il primo quantile q_α corrisponde "al primo α% dei dati".
Ad esempio i dieci decili $q_1, \dots, q_{10}$ corrispondono al primo $10\%, \dots, 100\%$ dei dati. Si vede che q_2 è la mediana.
Come nel caso della media, se il quantile cade tra due dati (cioè se n/α non è intero) si prende la media dei valori ai margini.

Esempio 5. Nell'esempio **??** la media del campione è

$$\begin{aligned}\overline{X} &= \frac{1}{60}\Big(19 \cdot 8 + 22,5 \cdot 13 + 27,5 \cdot 16 + 35 \cdot 5 + 45 \cdot 7 + 60 \cdot 11\Big) \\ &= 33.9083\dots\end{aligned}$$

Questo è il valore atteso dell'età, ma non necessariamente è uno dei valori del campione!

1.4 Indici di dispersione

Gli indici di dispersione mostrano come varia il campione.

Definizione 4. Sono detti *indici di dispersione*:

1. *IQR (Inter-Quartile Range)*: si calcola con $q_3 - q_1$, fornisce un'idea di dove si trovi la metà centrale del campione.

2. *Range*: solitamente la coppia del minimo e del massimo dei valori, ma può essere anche fornito come un intervallo o come la distanza tra massimo e minimo o come altro ancora, l'importante è che conservi l'informazione di "portata" del campione.

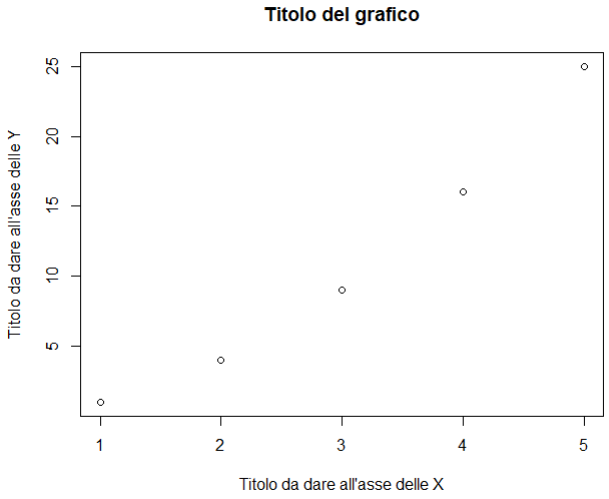
3. *Deviazione standard* σ: distanza degli scarti quadratici rispetto alla media.
$$\sigma \coloneqq \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Big(X_i - \overline{X}\Big)^2}$$

Osservazione 1 (Modelli e realtà)**.** Chiaramente si sottintende che queste siano definizioni campionarie, dipendono dal campione scelto; intuitivamente la deviazione standard descritta è quella campionaria, non quella effettiva della variabile aleatoria che ha generato il campione. Spesso si trascura questo fatto, che però può compromettere la "bontà" dei modelli.
I modelli sono - appunto - una interessante modellizzazione della realtà, ma possono essere diversi dalla realtà; è bene fidarsi dei modelli ma con un certo occhio critico.

1.4.1 Plot

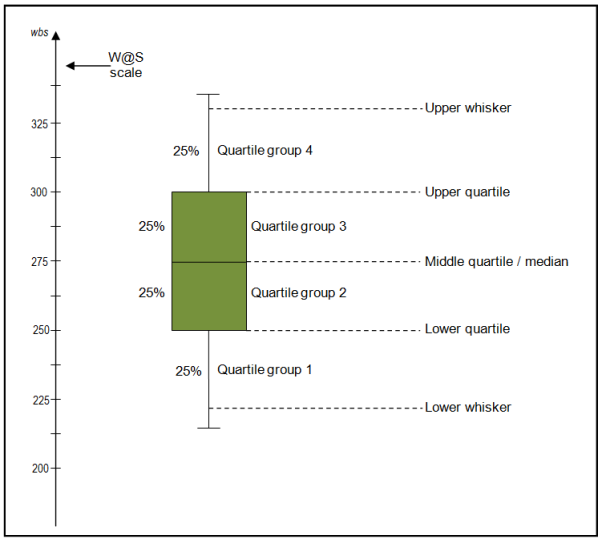
Supponiamo di avere un dataframe df con due attributi `x,y` dunque per poter rappresentare tali valori con un grafico cartesiano si fa

```
boxplot(x,y, data=df,
        main = "Titolo del grafico",
        xlab = "Titolo da dare all'asse delle X",
        ylab = "Titolo da dare all'asse delle Y")
```



1.4.2 BoxPlot

Un Box Plot è un metodo di rappresentazione grafica che fornisce informazioni su entrambi i tipi di indici.



Supponiamo di avere un dataset mtcars che presenta come dati mpg e cyl, per richiamare un boxplot su questi due dati in R bisogna scrivere

```
boxplot(mpg~cyl, data=mtcars,
        main = "Titolo del grafico",
        xlab = "Titolo da dare all'asse delle X",
        ylab = "Titolo da dare all'asse delle Y",
        horizontal= TRUE)
```

Esempio 6. Supponiamo di avere i seguenti dati:

$$11 \quad 13 \quad 15 \quad 15 \quad 16$$
$$18 \quad 21 \quad 22 \quad 23 \quad 23 \quad 29$$

- Siccome il numero di elementi è dispari, allora la mediana è evidentemente l'elemento centrale 18.
- Il primo ed il terzo quartile sono 15 e 23. Posso vederlo immaginando di rimuovere il 18 e vedere i due lati come campioni distinti di cui trovare la mediana, oppure posso fare il calcolo.
- La distanza interquartile è 8.

Osservazione 2 (Costruire BoxPlot orizzontali)**.** Disegniamo un rettangolo con estremi q_1 e q_3, poi rappresentiamo la mediana come barra verticale dentro il rettangolo. Infine si aggiungono due "baffi", cioè due barre orizzontali, uscenti dai due lati verticali del rettangolo, lunghe entrambe 1.5 volte la distanza interquartile.

- Se $q_1 - 1.5 \times IQR$ è minore del valore minimo, allora fermo il baffo sinistro al valore minimo. Se questo non vale vuol dire che il valore minimo è più lontano dalla fine del segmenti sinistro, e tutti i punti esterni sono indicati da un punto.
- Analogamente se $q_3 + 1.5 \times IQR$ è maggiore del valore massimo fermo il baffo destro al valore massimo, e nello stesso modo segno i punti esterni.

I punti che distano più di 1.5 IQR da q_1 o q_3 sono così lontani dalla distribuzione degli altri dati che si dicono **potenziali outliers**, sono valori molto importanti perché devono essere valutati attentamente; non necessariamente sono da eliminare, ma vale la pena di studiarli approfonditamente.
Fino a quando non motiviamo il loro valore lontano dalla distribuzione (es. stavamo aggiornando i dati durante quella particolare rilevazione dopo un po' di tempo, questo spiegherebbe gli outliers e potrebbe volendo permettere di rimuoverli) li diciamo "potenziali". E fin quando non abbiamo una motivazione, *eliminarli dai nostri dati è un grave errore e può compromettere l'efficacia del modello che andremo a costruire*.

Osservazione 3 (Nota sui BoxPlot verticali)**.** Si noti che il BoxPlot non è necessariamente orizzontale, spesso è verticale (ed alcuni programmi come R li visualizzano verticali di default!). Rispetto alla stessa scala possono essere fatti più BoxPlot di variabili diverse, questo risulta importante per la visualizzazione di grandi quantità di dati.

1.4.3 QQPlot

Vi è anche la possibilità di disegnare la correlazione che sussiste tra un campione dato e una distribuzione normale grazie al comando

```
qqplot()
```

Questo comando fa parte della libreria `car`

Capitolo 2

Comandi di base

2.1 Gestire l'ambiente di lavoro

Comando	Azione
`getwd()`	ricava il path (indirizzo) della cartella di lavoro
`setwd()`	modifica il path della cartella di lavoro

2.2 Calcoli numerici

Comando	Azione
`+, -, *, /`	classiche operazioni
`%/%, %%`	divisione intera e resto della divisione
`^`	elevamento a potenza
`sqrt()`	radice quadrata
`log(x, base - n)`	logaritmo di x in base n
`exp()`	funzione esponenziale
`sin()`	funzione trigonometrica (in radianti!)
`sinpi()`	funzione trigonometrica (in numeri di volte $\frac{\pi}{2}$!)

2.3 Assegnazioni

Comando	Azione
`a <- n`	assegnare ad a il valore n
`v <- c(a,b,c)`	assegnare ad v il vettore colonna (a, b, c)
`w <- t(v)`	assegnazione a w del vettore v trasposto
`length(v)`	lunghezza del vettore v
`v*w`	prodotto puntuale tra vettori
`v%*%w`	prodotto tra vettori per righe e colonne

2.4 Operazioni sui vettori

Comando	Azione
`a:b`	sequenza di valori tra a e b di passo unitario
`seq(a, b, p)`	sequenza di valori tra a e b di passo p, possono anche non essere interi o negativi
`rep(x,k)`	vettore colonna di valori x ripetuti k volte
`v[i]`	elemento i-esimo del vettore v, utile con il comando `which` (da preporre, vedi l'esempio!) per prendere le componenti con una certa proprietà
`max(v),min(v)`	massimo e minimo del vettore v

Osservazione 4. Supposto di avere un dataframe df, un comando molto importante per la ricerca è

`which(df$x=c)`

Dove c è una certa condizione che scegliamo noi.
Come output è un vettore che indica le righe del dataframe che soddisfano tale condizione.
Questa funzione ha anche altri aspetti implementativi come

`which.min(v)`

Che riporta la prima posizione del valore minimale del vettore riporta solo il primo che incontra.

Comando	Azione
`sum(v)`	somma dei valori del vettore
`diff(v,k)`	vettore che presenta nella componente i-esima la differenza tra la componente i-esima e la $i-k$-esima del vettore v; se k è omesso vale 1 di default, ha dimensione $dim(v)-1$

Osservazione 5. R non gestisce vettori di elementi di classi miste.

```
nomi <- c("Alice", "Bob", "Eve", 42)
```

In questo caso R rende 42 un carattere (`char`) perché il vettore è di caratteri.
Se il vettore è misto, la classe del primo elemento viene applicata a tutti gli altri elementi. Vedremo nei DataFrames e nelle liste come fare in quel caso.

2.5 Matrici

R non è ottimizzato per i calcoli matriciali, in quel caso è meglio usare MatLab. Nonostante questo alcuni sono supportati.

Comando	Azione
`A[i,j]`	elemento (i,j) della matrice
`cbind()`	lega i vettori come colonne di una matrice
`rbind()`	lega i vettori, ma come righe
`length(A)`	prodotto delle dimensioni della matrice
`A <- c(r,c)`	modifica la matrice A perché abbia r righe e c colonne; da evitare!

Si noti che in modo simile alla modifica della forma (intesa come righe e colonne) di una matrice posso usare questi comandi per modificare vettori, matrici ed altro. Ad esempio i nomi delle colonne di un DataFrame.

2.6 Miscellanea

Comando	Azione
`paste()`	lega elementi come stringhe di caratteri
`if(condizione) A else B`	(`A` e `B`| sono dei comandi), nel caso di vettori guarda solo la prima componente
`ifelse(condizione, A, B)`	identico, ma controlla tutte le componenti
`for(condizione){comando}`	ciclo "for"
`repeat`	ciclo infinito, devo spezzarlo con un "if"
`f <-function(input){ return(output)}`	come definire una funzione `f`

Esempio 7. `nth <- paste(1:12, c("st", "nd", "rd", rep("th",9)))`

output: `"1st", "2nd", "3rd", ..., "11th", "12th"`

2.7 Liste e DataFrames

2.7.1 Liste

Le liste sono oggetti in cui possiamo definire dei campi di caratteri, valori numerici, booleani. In questo modo posso assegnare più campi ad un solo elemento dell'insieme.

```
lista <- list(campoA = c(valoriA),
              campoB = c(valoriB), campoC = c(valoriC))
```

dove i "valori" sono vettori di informazioni.

Comando	Azione
`lista$campoC`	accede al vettore `campoC`
`lista[["campoC"]]`	accede al vettore `campoC`

2.7.2 DataFrame

Definizione 5. I DataFrames sono particolari liste che invece di avere dei campi hanno delle vere e proprie colonne.

Sono il modo migliore di salvare ed accedere a dati in R.

Se presento un data frame `df` allora per eliminare alcuni suoi campi devo scrivere

```
df<-df[,-c(v)]
```

In questo caso sto eleminando le colonne che hanno indice dato dal vettore `v`, se volessi eliminare determinate righe di cui so la posizione allora salvo la posizione su un vettore `v` e scrivo

```
df<-df[-c(v),]
```

Quello con osservazioni sulle righe e classi su colonne non è l'unico modo per scrivere un DataFrame, ma è stato scelto come convenzione.

Comando	Azione
`View(df)`	visualizza il DataFrame `df`
`df$colonna2`	accede alla colonna `colonna2`

Supponiamo di avere due dataframe `dataframe1, dataframe2` dunque se vogliamo unire i due dataframe dal punto di vista di colonne, quello che bisogna fare è

```
unione_df <- data.frame(dataframe1, dataframe2)
```

Il dataframe risultante avrà come colonne le colonne di dataframe1 e dataframe2.Mentre nel caso in cui scrivessi

```
unione_df <- c(dataframe1, dataframe2)
```

Questo creerebbe un vettore composto prima dai dati di `dataframe1` e poi da `dataframe2`.
Nel caso in cui un dataframe `df` abbia più colonne delle quali solo poche sono spesso utilizzate allora conviene usare il comando

```
attach(df)
```

In questo maniera posso accedere direttamente alle colonne di `df` senza dover ogni volta scrivere `df$colonnavoluta`.
In maniera del tutto opposta funziona il comando

```
detach(df)
```

Che non pemette più l'accesso diretto alle colonne del dataset `df`.
(A scanso di equivoci useremo DataFrames e DataSets come sinonimi per non appesantire il testo)

Capitolo 3

Lavorare su un DataSet

3.1 Lettura e studio dei dati

Alcuni DataSets sono subito disponibili nella memoria, ma potremmo necessitare di DataSet o funzioni particolari (salvate in pacchetti).
I packages vengono scaricati da CRAN, la principale repository di R, e possiamo caricarli dall'interfaccia grafica con

```
tools > install packages
```

oppure scrivendo il comando

```
install.packages('hflights', dependencies = TRUE)
```

a cui come visto posso aggiungere opzioni di installazione: ad esempio
`dependencies = TRUE` richiede di installare tutti i pacchetti che possono servirmi nel suo utilizzo. La funzione `search()` mi mostrerà tutti i pacchetti caricati.
Infatti dopo averlo installato non è pronto all'utilizzo, devo caricarlo nella sessione con

```
library(hflights)
```

Si noti che posso necessitare di una funzione o di un DataSet presente in un package (nell'esempio il package sarà `MASS`), in quel caso uso il comando

```
MASS::beav1
```

che può essere utile in caso di conflitti tra nomi di funzioni appartenenti a packages diversi.

Come caricare un DataFrame:

```
df <- read.table('indirizzo sul computer', header = TRUE)
```

In questo modo ho caricato un DataFrame con un header (intestazione). Alternativamente posso anche scegliere di caricarlo da internet inserendo l'URL.

```
df <- read.table(url('indirizzo'), header = TRUE)
```

Oppure posso usare

```
df <- read.csv('indirizzo sul computer')
```

- Comma Separated Values è il formato in cui troveremo molte tabelle, i valori sono separati da virgole
- Il comando imposta automaticamente `header = TRUE`
- Posso definire altri delimitatori per sostituire le virgole, per informazioni si veda la schermata help del comando

Potrei anche costruire DataSets dal nulla, ma non vedremo come farlo in questo corso; non sarà richiesto.

Comando	Azione
`head(df)`	visualizza le prime 6 righe del DataFrame (posso specificare quante righe mostrare)
`dim(df)`	dimensioni del DatFrame
`names(df)`	nomi delle colonne
`str(df)`	struttura del DataFrame e classi dei dati
`summary(df)`	informazioni sintetizzate sulle variabili (per le numeriche media, quartili, ...) (per le categoriche frequenza, ...)
`apply(array, f)`	applica la funzione `f` a tutto l'array
`lapply(array, funzione)`	come prima ma sulle liste
`unique(array)`	mostra le variabili "uniche"
`any(df)`	verifica se una condizione è vera, esito booleano
`table()`	dato un vettore di variabili categoriche, conta quante volte appare ogni elemento
`proptable()`	come sopra ma con proporzioni

3.2 Pulizia dei dati

Comando	Azione
`round()`	arrotondamento
`na.omit()`	rimuove i valori NA
`na.rm = TRUE`	uguale a sopra, ma usato in argomento a funzioni
`sort(array)`	ordina l'array
`as.Date()`	modifica lo stile di scrittura della data

Esempio 8. Se ho un vettore v di dati categorici, posso vederlo **come fattore**.

```
v <- as.factor(v)
```

In questo modo lo sovrascrivo con il suo stesso fattore. Usando `levels(v)` vedo che ora ho ottenuto dei livelli (numerici), e posso riassegnarli in questo modo.

```
levels(v) <- c('primo', 'secondo', 'terzo')
```

Se svolgo queste operazioni su un DataSet (cioè se il vettore v è in realtà la colonna di un DataSet) le modifiche sono inserite nello stesso set.

3.3 Visualizzazione dei dati

Possiamo estrarre informazioni dai dati e renderle leggibili a colpo d'occhio.

Comando	Azione
`par(mfrow = c(1,2))`	modalità di visualizzazione dei grafici, ora ne vedo due in una riga (si pensi ad una matrice $m \times n$)
`boxplot(v)`	BoxPlot di un'osservazione v
`boxplot(df$x ~ df$y, data = df)`	BoxPlot di df rispetto x ed y (una box diversa per ogni valore di x)
`hist(v)`	istogramma di un'osservazione v (di default l'asse y è la frequenza)
`plot(v,w)`	grafico delle relazioni tra v e w
`abline()`	inserisce una linea nel grafico
`aggregate()`	come apply ma per DataFrames
`merge()`	fonde due DataSets rispetto ad una variabile
`any()`	"risponde" ad una richiesta (deve essere presente in entrambi!)

Esempio 9. Vediamo la seguente funzione:

```
aggregate(. ~ Species, data = iris, mean)
```

Utilizza tutte le variabili rispetto a Species| (categorica), ed applica la funzione `mean` a tutte le altre classi.
Il punto indica che uso tutto il DataSet.

Esempio 10. Se vogliamo trovare degli NA in un vettore:

```
v <- c(1,2,3,NA)
any(is.na(v))
```

E questo risulta nell'output `[1] TRUE`. Per capire quale valore abbia prodotto questo output, useremo `which()`.

```
which(is.na(v))
```

Ed eventualmente, avendo ottenuto `[1] 4`, possiamo scrivere con uno dei seguenti:

```
v[is.na(v)] <- 0
v[which(is.na(v))] <- 0
na.omit(v)
```

uno qualunque di questi metodi può funzionare, i primi due sostituiscono con un altro valore, l'ultimo invece lo rimuove totalmente.
Vedremo più avanti l'ultimo metodo nel dettaglio.

Capitolo 4

Probabilità in R

4.1 Distribuzioni di probabilità

Esistono diverse funzioni per le principali distribuzioni di probabilità che partono dal loro nome e si ottengono aggiungendo un prefisso al nome della distribuzione.

Prefisso	Azione
`r-`	genera una serie di numeri a random seguendo la distribuzione di parametri dati (campionamento casuale)
`d-`	densità della distribuzione data
`q-`	quantili della distribuzione data

Una lista completa si può trovare dalla finestra help di R.

4.2 Test d'ipotesi

Supponiamo di avere un campione statistico $(X_1, \cdots, X_n)$ dunque vi possono essere 3 casi riguardanti la sua distribuzione e su come stimare media e varianza:

	Media	**Varianza**
Non conosco nulla	$\frac{\overline{X}_n - \mu}{\sqrt{S_n^2/n}} \sim t_{n-1}$	$\frac{(n-1)S_n^2}{\sigma} \sim \chi^2_{n-1}$
Si conosce la media μ		$\frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} \sim N(0,1)$ $\quad \frac{(n-1)S_n^2}{\sigma} \sim \chi^2_{n-1}$
Si conosce la varianza σ^2	$\frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} \sim N(0,1)$	

Per tali test esistono vari comandi:

- Test per la media
  Supponiamo avere un sistema di ipotesi della forma

  $$\begin{cases} H_0 : \mu = \mu_0 \\ H_0 : \mu \neq \mu_0 \end{cases}$$

  Con μ_0 un certo valore che sappiamo a priori o che speriamo si verifichi (in questo caso supponiamo che sia 135) con un livello di affidabilità del 99% e sia `df` un dataframe dunque il comando da inviare è

  ```
  t.test(campione_statistico,
         alternative = 'two.side',
         mu = 135,
         var.equal=TRUE
         conf.level = .99)
  ```

In tutto questo è importante osservare alcuni parametri dell'input:

Alternative serve per sapere che tipo di ipotesi alternativa ho. In questo caso ho che è two side in quanto richiedo come ipotesi nulla un determinato valore ma nel caso in cui l'ipotesi nulla sia $\mu < \mu_0$ allora dovrei mettere

`alternative = 'greater'`

Mentre se l'ipotesi nulla è $\mu > \mu_0$ allora dovrei mettere

`alternative = 'less'`

var.equal serve per sapere se sussiste l'omoschedasticità.

- Test per la varianza

4.3 Regressione lineare

Supponiamo di avere un dataframe "df" con attributi x, y e di voler valutare una dipendenza lineare del tipo:

$$y = \beta_1 x + \beta_0$$

dunque per calcolare la regressione lineare (avvicinare il più possibile una retta ai dati in un modello lineare) esiste una funzione detta *linear model*

```
reg <- lm(y ~ x, data = df)
```

Dove:

- `x` rappresenta la variabile indipendente.
- `y` rappresenta la variabile dipendente.
- `df` è il nome del DataFrame.

A questo punto è consigliabile usare summary per capire cosa abbiamo creato:

```
summary(reg)
```

Coefficient:

	Estimated	Std. Error	t.value	Pr(>|T|)
(Interceptor)				
x				

In questo caso abbiamo che:

- `Estimated` di Interceptor = β_0 mentre `Estimated` di x = β_1
- `t.value` di x rappresenta t_{oss} ovvero il valore della distribuzione di Student alla luce dei nostri dati con H_0 vera.
- $P(> |T|)|$ rappresenta il p-value calcolato mediante un T-test.
  In questo modo controllando tale valore potrò affermare se considerare o meno come parametri della retta di regressione quelli Stimati da questo metodo.

Possiamo fare anche un ANalysis Of VAriance mediante il comando

```
anova(reg)
```

Il cui output, considerando p i parametri da stimare (in questo caso 2), è

Analysis of Variance Table
Response: y

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
x	p-1	SSR	$\frac{SSR}{p-1}$		
Residuals	$n-2$	SSE	$\frac{SSE}{n-p}$		

È importante osservare che $F-value$ ritorna il valore osservato della distribuzione di fisher-sneidecor alla luce dei dati osservati.

Un altro comando importante per l'analisi dei dati è

```
confint(reg)
```

Che restituisce l'intervallo di confidenza dei due parametri con un livello di accuratezza del 95%.

Osservazione 6. Da osservare che `reg` salva anche le coppie $(x_i, \hat{Y}_i)$ che vengono restituite da

```
reg$fitted.value
```

Abbiamo informazioni su R^2 ed il p-value sui due coefficienti visti come vettore.

In conclusione posso selezionare con notazione vettoriale i valori di `reg`.

1. **Visualizzare la retta di regressione**:
   Se si richiede

   ```
   plot(rev)
   ```

   allora R ci mostra subito una serie di grafici riguardo il modello costruito. I punti del grafico a cui sono assegnati i loro nomi sono gli *outliers*.
   Un grafico particolarmente interessante è quello della distanza di Cook, che ci dice quanto un valore influenzi (negativamente) il nostro modello. Per capirlo si veda se il punto rientra nelle aree tratteggiate.

2. **Studiare i residui** (per verificare che il modello di fit sia "ragionevole"):
   Possiamo studiare i residui per vedere se la nostra approssimazione è buona.

Comando	Azione
`predict()`	prevede il fit di un modello
`qqnorm(rev$residuals)`	confronta i residui con la normale
`qqline(rev$residuals)`	confronta i residui con la normale

   Osservazione 7. Posso aggiungere altri dati alla regressione lineare per ottenere un modello migliore.

   ```
   reg <- lm(x ~ y + z, data = df)
   ```

   Figura 4.1: aggiunta di un dato

   Il modello è migliore perché ho costruito il fit della prima variabile sia rispetto alla seconda che alla terza, ma è difficilmente interpretabile dato che ha tre dimensioni.
   Posso comunque usare `plot` sull'output e dare inizio ad una sfida per chi capisca prima il fit dei dati.

   Aumentando il numero di variabili però perdo molta confidenza nelle variabili da considerare, il modello peggiora.

Un comando interessante è `step`, che parte da un modello con fit su tutte le variabili del DataFrame e gradualmente toglie un elemento ogni volta fino alla situazione in cui non riuscirebbe a togliere altri elementi senza peggiorare la stima (valutata secondo un particolare indice detto AIC).
In sostanza costruisce un modello il più semplice ed ottimale possibile.
A volte cambiando l'ordine degli input cambia anche il risultato!

Si noti che non è sempre buona etica usare questo comando perché potrebbe rimuovere come nulla fosse una variabile fondamentale ed assegnare il fit finale a delle variabili insensate, questo solitamente si verifica a posteriori (o non si verifica e rovina il modello). [1]

[1] ad esempio dopo un comando step la gente potrebbe iniziare a credere che portare le mascherine sia inutile.

Capitolo 5

Esempi di esercizi svolti

5.1 Esercizio da esame

Si trovi un modo adeguato per importare i dati relativi al reddito nazionale lordo pro capite e alla percentuale di strade asfaltate in R. Dopo aver analizzato e preparato i DataSet, si usino i dati per investigare le due variabili rispetto ad un paese del G7, un paese in via di sviluppo ed un paese del terzo mondo a scelta, nel periodo dal 1990 al 2009. Si analizzino e si visualizzino i dati.

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

dfPPP <- read.xlsx("./data/indicatorGNIpercapitaPPP.xlsx")
dfroad <- read.xlsx("./data/roads paved.xlsx")

dfPPP <- read.xlsx("../../Lezione 5/data
                   /indicatorGNIpercapitaPPP.xlsx")
dfroad <- read.xlsx("../../Lezione 5/data/
                    roads paved.xlsx")
```

Seleziono **solo** le colonne relative agli anni che mi interessano per le analisi, ed ovviamente la prima colonna con i nomi.

```
dfroad1 <- dfroad[,paste(c("Roads,.paved.
                  (%.of.total.roads)",paste(1990:2009)))]
dfPPP1 <- dfPPP[,paste(c(names(dfPPP)[1], 1990:2009))]

View(dfroad1)
View(dfPPP1)
```

Cerchiamo di capire quanti e quali sono i paesi per cui ho dati completi: questo passaggio non è obbligatorio, si possono anche scegliere paesi con dati incompleti, facendo attenzione ai NA.

```
View(na.omit(dfroad1) )
View(na.omit(dfPPP1) )
```

Voglio controllare quali sono i paesi nell'intersezione dei due DataFrame (Si può anche fare a mano).

```
na.omit(dfroad1)[1]
na.omit(dfPPP1)[1]
intersect(as.matrix(na.omit(dfroad1)[1]),
          as.matrix(na.omit(dfPPP1)[1])
```

Scelgo come paesi G7, in via di sviluppo e terzo mondo, usando la mappa in `http://www.nationsonline.org/oneworld/t`
e prendo UK (unico con dati completi), Ukraine e Marocco. Seleziono i dati relativi a questi paesi.

```
dfroad2 <- dfroad1[dfroad1$‘Roads,.paved.(%.of.total.roads)‘
                   == "United Kingdom" | dfroad1$‘Roads,
                   .paved.(%.of.total.roads)‘
                   == "Ukraine"| dfroad1$‘Roads,
                   .paved.(%.of.total.roads)‘
                   == "Morocco",]
dfPPP2 <- dfPPP1[dfPPP1$‘GNI.per.capita,
                 .PPP.(current.international.$)‘
                 == "United Kingdom"| dfPPP1$‘GNI.per.capita,
                 .PPP.(current.international.$)‘
                 == "Ukraine"| dfPPP1$‘GNI.per.capita,
                 .PPP.(current.international.$)‘
                 == "Morocco",]
View(dfroad2)
View(dfPPP2)
```

Studiamo il summary per anno.

```
summary(dfroad2)
summary(dfPPP2)
```

Studiamo il summary per paese (si può fare in più modi).

```
dfroad3 <- t(dfroad2)[-1,]
dfroad3 <- as.data.frame(dfroad3)
names(dfroad3) <- t(dfroad2)[1,]
summary(dfroad3)
summary(t(dfPPP2)[-1,] )
dfPPP3 <- t(dfPPP2)[-1,]
dfPPP3 <- as.data.frame(dfPPP3)
names(dfPPP3) <- t(dfPPP2)[1,]

dfPPP3$Morocco <- as.numeric
                  (as.character(dfPPP3$Morocco))
dfPPP3$Ukraine <- as.numeric
                  (as.character(dfPPP3$Ukraine))
dfPPP3$‘United Kingdom‘ <- as.numeric(as.character
                           (dfPPP3$‘United Kingdom‘))
summary(dfPPP3)
```

Finita la parte opzionale, visualizzo i dati.

```
par(mfrow = c(3,2))
for (i in 1:3){
plot(1990:2009, dfroad2[i,-1],
     main = dfroad2$‘Roads,
     .paved.(%.of.total.roads)‘[i],
     ylab = "% of paver roads")
plot(1990:2009,dfPPP2[i,-1],
     main = dfPPP2$‘GNI.per.capita,
     .PPP.(current.international.$)‘[i],
     ylab = "GPD $")   }
```

Scriviamo anche il codice per avere i nomi dei paesi nei BoxPlot, invece che il numero di riga originale:

```
row.names(dfroad2) <- dfroad2[,1]
row.names(dfPPP2) <- dfPPP2[,1]

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

```
boxplot(t(dfroad2[,-1]) , main = "Paved roads (%)")
boxplot(t(dfPPP2[,-1]), main = "GPD $" )
```

Inoltre si usi la regressione lineare per analizzare la relazione tra le variabili per uno dei due paesi scelti. Si commentino e discutano i risultati.
Voglio analizzare i dati per il Marocco e vedere se la relazione tra GPD e strade asfaltate è lineare.

```
reg1 <- lm(t(dfroad2[1,-1]) ~ t(dfPPP2[1,-1]) )
summary(reg1)
```

Da una prima analisi dell'indice R^2 la nostra supposizione sembra giusta. Visualizziamo più nel dettaglio i risultati.

```
par(mfrow = c(2,2), mar = c(2,2,1,1))
# Retta di regressione
plot(t(dfroad2[1,-1]) ~ t(dfPPP2[1,-1]))
abline(reg1$coefficients, col = "green")
# Pattern nei residui
plot(reg1$residuals, main = "Residui")
# Distribuzione in quantili
qqnorm(reg1$residuals)
qqline(reg1$residuals)
hist(reg1$residuals)

## 29/05/2019, Federico Reali
```

Sembra che il modello sia piuttosto adeguato; i residui non presentano pattern evidenti, mentre la distribuzione in quantili si allontanata da quella di riferimento per i valori più grandi.

(Crediti per l'esercizio e lo svolgimento: Federico Reali)

5.2 Pulizia di un DataFrame

Valori NA in un DataFrame

Vediamo un modo semplice di rimuovere i valori NA da un DataFrame.
Costruiamo un DataFrame magnifico:

```
a <- c(rep('Stark',3),'not Stark','Stark')
b <- c(1,2,3,NA,5)

df <- data.frame(' Utente ' = a, ' Malvagita' ' = b)
```

Possiamo costruire un DataFrame senza NA come segue:

```
na.omit(df)
```

Con i metodi visti possiamo anche sostituire i valori NA con altri (ad esempio la media, o la mediana).

Valori numerici letti erroneamente

Supponiamo invece di avere un DataFrame con colonne x ed y in cui sappiamo di avere dei valori numerici erroneamente letti come non numerici, possiamo risolvere con:

```
df$x <- as.numeric(as.matrix(dff$x))
```

Si noti che questo è fondamentale, **le funzioni che operano su valori numerici altrimenti rischiano di non funzionare**.
Questo comando sfrutta il passaggio dei valori non numerici tra classi numeriche.

Valori non numerici

Per rendere NA valori che non sono numerici posso sfruttare un trucco simile:

```
v <- c(1,2,3,NA,'Eulero')
as.numeric(as.character(v))
```

ed otteniamo un warning.

```
Warning message:  si è prodotto un NA per coercizione
```

Questo perché, se stampiamo v:

```
[1]  1  2  3 NA NA
```

Possiamo ovviamente sfruttare questo trucco per pulire i dati, sfruttando tutti i metodi visti per gli NA in precedenza.

Capitolo 6

Nota al lettore e formulario

Formulario

Nella prossima pagina troverete un utile formulario con tutti i comandi più importanti visti nel corso della teoria, ed alcuni interessanti esempi della loro applicazione.
Per ovvie questioni di spazio - fondamentale in un formulario da esame - non è stato possibile essere troppo prolissi.

Breve nota stilistica

Si noti che abbiamo scritto più volta la tilde, purtroppo l'ambiente LaTeXper inserire codice la riporta più in alto di come dovrebbe essere.

- In ambiente Verbatim: `aggregate(.~ species, data=iris,mean)`

Sto lavorando sui miei poteri paranormali per risolvere il problema.

Disclaimer

Questo PDF è stato scritto da un povero studente per altri studenti, non intende in alcun modo sostituire la magnificenza delle esercitazioni dal vivo.
Andate a lezione, gente!

Se doveste riscontrare errori contattatemi:

`projectprometheus.org@gmail.com`

Tutti gli abusi di notazione saranno puniti.
- Progetto Prometeo

Tavola dei comandi

Comando	Azione
`read.table()`, `read.csv()`, `read.xls()`	legge un dataset
`install.packages("maps")`	installa il package
`a%%b`, `a%/%b`	resto, divisione intera
`as.factor`,`as.numeric`	fattori e classi
`levels(v)<-names()`	nomi per il fattore
`aggregate(.~ species, data=iris,mean)`	fattori e classi
`cbind`, `rbind`	lega per colonne e righe
`ppois(q,lambda)`	$\mathbb{P}(X \leq p)$ in Poisson
`dpois(q,lambda)`	distribuzione di Poisson in q
`log(n,base)`	logaritmo
`seq(a,b,step)`	sequenza di uno step dato
`mean`, `sd`, `IQR`	media, deviazione, IQR
`view(df)`, `str(df)`, `head(df)`,`summary(df)`	boxplot delle due variabili
`table()`, `proptable()`	rappresentazioni in tabella
`table(df[,c('col1','col2')])`	relazioni tra le due colonne
`par(mfrow=c(a,b))`	dimensione dell'array dei plot
`boxplot(y~x,data=df,main="titolo")`	boxplot delle due variabili
`plot(y~x,type="l")`	grafico per linea
`hist(y~x,data=df,xlab='x',ylab='y'freq=F)`	istogramma, `freq=F` per la densità
`abline(reg$coefficients,col="green")`	aggiunge una retta al grafico
`ylim=c(0,300)`	forzare la scala del grafico
`reg <- lm(x~y,data=df)`	regressione lineare
`step`	meglio del precedente, più variabili
`plot(reg$residuals)`	studio dei residui
`qqnorm(reg$residuals)`	visualizza il fit dei dati
`qqplot(reg$residuals)`	visualizza il fit dei dati
`t.test(x[],y[])`	correlazione tra le due osservazioni
`t.test(x,y)`	esiste una correlazione?

Esempi di comandi

`var <- which(var == Inf | var == -Inf | var > 0)`

`round`, `unique`, `any`, `which`, `apply`

`t.test(x,alternative='greater',mu=135,conf.level=95)`

`v <- as.numeric(as.matrix(v))`, `v <- as.numeric(as.character(v))`

`date <- strptime(as.character(date) "%m/%d/%y")`

`weekdays(as.Date("2020-05-21"))`

`col=as.factor(data$var>150)`

`tz="America/Los_Angeles"`

`diff(c(1,2,7))` (1,6)

`myfun <- function(a,b){return()}`

`ifelse(cond,com1,com2)`, `if(cond) com1 else com2`

`for(i in int){}`

`data$var[which(is.na(data$var))] <- median(data$var, na.rm=T)`

`map("world","italy")`

`z <- which(is.na(v))`, `for(i in z){v[i] <- mean(v, na.rn=T)}`

`index <- which(is.na(v))`, `v[index] <- mean(v, na.rn=T)`