R per l'analisi statistica multivariata

Unità C: i `data.frames`

Tommaso Rigon

Università Milano-Bicocca

Unità C

Argomenti affrontati

- L'oggetto `data.frame`
- Importazione di un dataset in memoria
- Esplorazione e manipolazione di un dataset
- Tipologia di variabili
- I valori mancanti

- Esercizi **R** associati: `https://tommasorigon.github.io/introR/exe/es_1.html`

Gli oggetti di tipo `data.frame`

- Un oggetto **R** chiamato `data.frame` corrisponde alla **matrice dei dati**.
- Ciascuna riga rappresenta quindi un'**unità statistica**, mentre ciascuna colonna rappresenta invece una **variabile**.
- Un `data.frame` assomiglia ad una matrice, ma è pensato per l'analisi dei dati.
- Ad esempio, un `data.frame` può contenere anche valori non numerici, come per esempio le **variabili qualitative** o **valori mancanti**.
- Come vedremo, il `data.frame` viene generalmente caricato all'interno di **R** utilizzando ad esempio la funzione `read.table`.

Importazione di un dataset in memoria

- Il modo più frequente di caricare un dataset in memoria è importarlo da un **file esterno**.
- Se i dati hanno dimensioni piccole / medie (dimensione file $<$ 3-4 Gb), sono spesso salvati in formato `.csv` oppure `.txt`.
- In contesti reali e più complessi, è possibile importare i dati a partire direttamente da database relazionali come `SQL`.
- È sconsigliabile importare i dati in **R** a partire da un file Excel. Più in generale, ci sono svariate ragioni per evitare Excel se l'obiettivo è condurre una rigorosa analisi dei dati.
- **Nota**. Per poter procedere con i prossimi comandi, è necessario scaricare il file `calcio.csv` e salvarlo nel proprio computer.
- **Link al file**: `https://tommasorigon.github.io/introR/data/calcio.csv`.

I dati grezzi (editor di testo)

```
"Date","HomeTeam","AwayTeam","FTR","B365H","B365D","B365A"
2014-08-30,"Chievo","Juventus","A",7,4,1.5
2014-08-30,"Roma","Fiorentina","H",1.67,3.8,5
2014-08-31,"Atalanta","Verona","D",2.05,3.4,3.6
2014-08-31,"Cesena","Parma","H",3.1,3.2,2.35
2014-08-31,"Genoa","Napoli","A",3.9,3.4,1.95
2014-08-31,"Milan","Lazio","H",2,3.4,3.75
2014-08-31,"Palermo","Sampdoria","D",2.2,3.25,3.3
2014-08-31,"Sassuolo","Cagliari","D",2.3,3.4,3
2014-08-31,"Torino","Inter","D",3.1,3.3,2.3
2014-08-31,"Udinese","Empoli","H",1.8,3.5,4.5
2014-09-13,"Empoli","Roma","A",5.5,3.75,1.62
2014-09-13,"Juventus","Udinese","H",1.4,4.5,8
2014-09-14,"Cagliari","Atalanta","A",2.25,3.4,3.1
2014-09-14,"Fiorentina","Genoa","D",1.57,3.8,6
2014-09-14,"Inter","Sassuolo","H",1.44,4.33,7
2014-09-14,"Lazio","Cesena","H",1.33,5,9
2014-09-14,"Napoli","Chievo","A",1.36,4.75,8
2014-09-14,"Parma","Milan","A",2.7,3.2,2.6
2014-09-14,"Sampdoria","Torino","H",2.4,3.1,3.1
2014-09-15,"Verona","Palermo","H",2.2,3.3,3.3
2014-09-20,"Cesena","Empoli","D",2.8,3,2.7
2014-09-20,"Milan","Juventus","A",3.2,3.3,2.25
2014-09-21,"Atalanta","Fiorentina","A",3.25,3.1,2.3
```

La cartella di lavoro

- La sessione di **R** attiva è aperta in una specifica **cartella di lavoro**, identificata dal comando `getwd`.
- Ad esempio, la cartella in cui il codice è stato eseguito è la seguente

```
getwd() # Identifica la cartella di lavoro (get working directory)
# [1] "/Users/tommaso/Google Drive/University/Bicocca/Didattica/Laboratorio Statistica/introR"
```

- Per aprire il file `calcio.csv`, bisogna fornire ad **R** il **percorso del file**, che dipende da dove questo è stato salvato (sul Desktop, cartella downloads, etc).

```
# LA VARIABILE PATH VA CAMBIATA A SECONDA DI DOVE SIA SALVATO IL FILE .csv NEL PROPRIO PC
path <- "data/calcio.csv" # Questo è il percorso del file, relativo a getwd()
```

- Il percorso del file può anche essere un **link**, come in questo caso:

```
path <- "https://tommasorigon.github.io/introR/data/calcio.csv"
```

Importazione del dataset

- È anche possibile **cambiare la cartella di lavoro** tramite il comando `setwd`.
- In Rstudio, si può usare l'opzione `More -> Set as working directory` nella finestra `Files` e selezionando la cartella di interesse.
- Una volta identificato il corretto `path` del file, per **importare il dataset** in memoria si usa ad esempio il comando `read.table`. Pertanto:

```
calcio <- read.table(path, header = TRUE, sep = ",")
```

- **Nota 1**. L'opzione `header = TRUE` significa che la prima riga contiene i nomi delle variabili.
- **Nota 2**. L'opzione `sep = ","` indica che i vari valori di ciascuna variabile sono separati da una virgola.
- Infine, si noti che nel caso in cui il file sia contenuto nella cartella di lavoro, è quindi sufficiente usare:

```
calcio <- read.table("calcio.csv", header = TRUE, sep = ",")
```

Esplorazione di un `data.frame`

- In primo luogo, siamo interessati a capire quante **variabili** e quante **osservazioni** sono contenute nel dataset `data.frame`:

```
dim(calcio) # Equivalente a c(nrow(calcio), ncol(calcio))
# [1] 1900   7
```

- Per verificare di non aver commesso errori grossolani nell'importazione del `data.frame`, possiamo visualizzare le prime 6 osservazioni con il comando `head`:

```
head(calcio) # Comando equivalente: calcio[1:6, ]
#         Date HomeTeam   AwayTeam FTR B365H B365D B365A
# 1 2014-08-30   Chievo   Juventus   A  7.00   4.0  1.50
# 2 2014-08-30     Roma Fiorentina   H  1.67   3.8  5.00
# 3 2014-08-31 Atalanta     Verona   D  2.05   3.4  3.60
# 4 2014-08-31   Cesena      Parma   H  3.10   3.2  2.35
# 5 2014-08-31    Genoa     Napoli   A  3.90   3.4  1.95
# 6 2014-08-31    Milan      Lazio   H  2.00   3.4  3.75
```

- Il comando `tail` può essere usato per mostrare le ultime 6 osservazioni:

```
tail(calcio) # Comando equivalente: calcio[1885:1900, ]
```

Il dataset calcio

- Il dataset calcio contiene quindi un totale di 7 variabili. Ciascuna osservazione è una **partita di calcio** avvenuta nel campionato di **serie A** dal 2008 al 2015.
- La variabile `Date` indica la data della partita.
- Le variabili `HomeTeam`, `AwayTeam` indicano la squadra che ha giocato in casa ed in trasferta, rispettivamente.
- La variabile `FTR` ("Full Time Result") indica se la partita è stata vinta dalla squadra di casa (`H`), dalla squadra in trasferta (`A`), o si è conclusa con un pareggio (`D`).
- Le variabili `B365H`, `B365D` e `B365A`, indicano le quote della compagnia di scommesse **Bet365 Group Ltd** relative ai 3 eventi, prima dell'inizio della partita.
- Per **accedere** e **modificare** i nomi delle variabili, si utilizza il comando `colnames`.

```
colnames(calcio) # Per accedere ai nomi delle variabili
# [1] "Date"     "HomeTeam" "AwayTeam" "FTR"      "B365H"    "B365D"    "B365A"
```

Tipologia di variabili

- In un dataset sono presenti diverse **tipologie di variabili**.
- Le variabili quantitative discrete e continue in **R** sono codificate come oggetti di tipo `integer` e `numeric`, rispettivamente
- Le variabili qualitative in **R** sono codificate come oggetti di tipo `character` oppure di tipo `factor`; la differenza tra queste tipologie verrà chiarita nelle prossime slides.
- Infine le date in **R** si codificano con la tipologia `Date`.
- **Nota**. **R** non sempre riconosce correttamente la tipologia di variabili, come evidenziato dal comando `str`, che fornisce un sommario del `data.frame`.

```
str(calcio)
# 'data.frame': 1900 obs. of  7 variables:
#  $ Date    : chr  "2014-08-30" "2014-08-30" "2014-08-31" "2014-08-31" ...
#  $ HomeTeam: chr  "Chievo" "Roma" "Atalanta" "Cesena" ...
#  $ AwayTeam: chr  "Juventus" "Fiorentina" "Verona" "Parma" ...
#  $ FTR     : chr  "A" "H" "D" "H" ...
#  $ B365H   : num  7 1.67 2.05 3.1 3.9 2 2.2 2.3 3.1 1.8 ...
#  $ B365D   : num  4 3.8 3.4 3.2 3.4 3.4 3.25 3.4 3.3 3.5 ...
#  $ B365A   : num  1.5 5 3.6 2.35 1.95 3.75 3.3 3 2.3 4.5 ...
```

Variabili quantitative

- Per **estrarre una variabile** da un `data.frame` si procede come nel caso delle liste, ovvero tramite il simbolo `$`. Le variabili possono essere salvate altrove.

- Una variabile numerica è un vettore **R** a tutti gli effetti, pertanto possiamo applicare le funzioni che abbiamo già incontrato finora

```
is.numeric(calcio$B365H) # Verifico che si tratta di una variabile di tipo numeric
# [1] TRUE
class(calcio$B365H)
# [1] "numeric"

calcio$B365H[1:10] # Primi 10 elementi di un vettore R
# [1] 7.00 1.67 2.05 3.10 3.90 2.00 2.20 2.30 3.10 1.80
```

- Le **analisi descrittive** di una variabile numerica verranno affrontate nelle unità successive, per ora ci limitiamo alla manipolazione del `data.frame`.

Variabili qualitative I

- Le variabili di tipo `character` sono appropriate per le stringhe di testo, ovvero se ciascun campo corrisponde ad un breve testo, come ad esempio un Tweet.
- Le variabili qualitative con valori ripetuti vanno invece codificate come `factor`.
- Dobbiamo quindi **convertire** le 3 variabili `HomeTeam`, `AwayTeam` e `FTR` come segue:

```
calcio$HomeTeam <- factor(calcio$HomeTeam)
calcio$AwayTeam <- factor(calcio$AwayTeam)
calcio$FTR <- factor(calcio$FTR)
```

- In alternativa, possiamo fare questa operazione in fase di lettura di dati, tramite l'opzione `stringsAsFactors`. Quindi:

```
calcio <- read.table(path, header = TRUE, sep = ",", stringsAsFactors = TRUE)
```

Variabili qualitative II

- In primo luogo, verifichiamo che la conversione sia avvenuta correttamente ed esploriamone il contenuto della variabile `HomeTeam`:

```
class(calcio$HomeTeam)
# [1] "factor"
calcio$HomeTeam[1:10]
# [1] Chievo   Roma     Atalanta Cesena   Genoa    Milan    Palermo  Sassuolo
# [9] Torino   Udinese
# 28 Levels: Atalanta Bari Bologna Brescia Cagliari Catania Cesena ... Verona
```

- Nelle variabili di tipo `factor` sono pertanto elencate le varie **modalità**, chiamati `levels` in **R**.

- Per accedere alle modalità si usa il comando `levels`, che elenca le modalità della variabile considerata in ordine alfabetico:

```
levels(calcio$HomeTeam)
#  [1] "Atalanta"   "Bari"       "Bologna"    "Brescia"    "Cagliari"   "Catania"
#  [7] "Cesena"     "Chievo"     "Empoli"     "Fiorentina" "Genoa"      "Inter"
# [13] "Juventus"   "Lazio"      "Lecce"      "Livorno"    "Milan"      "Napoli"
# [19] "Palermo"    "Parma"      "Reggina"    "Roma"       "Sampdoria"  "Sassuolo"
# [25] "Siena"      "Torino"     "Udinese"    "Verona"
```

Variabili qualitative III

- Il comando `levels` permette di **rinominare le modalità**, ad esempio:

```
calcio$FTR[1:10]
#  [1] A H D H A H D D D H
# Levels: A D H

levels(calcio$FTR) <- c("Away", "Draw", "Home") # Rinomino le modalità

calcio$FTR[1:10]
#  [1] Away Home Draw Home Away Home Draw Draw Draw Home
# Levels: Away Draw Home
```

- Il comando `levels` consente inoltre di accorpare le modalità di una variabile qualitativa:

```
calcio$Draw <- calcio$FTR # Creo una copia della variabile FTR chiamata Draw
levels(calcio$Draw) <- c("Not_Draw", "Draw", "Not_Draw") # Accorpamento delle modalità

calcio$Draw[1:10]
# [1] Not_Draw Not_Draw Draw     Not_Draw Not_Draw Not_Draw
# [7] Draw     Draw     Draw     Not_Draw
# Levels: Not_Draw Draw
```

Le variabili di tipo date

- La variabile `Date` del dataset `calcio` rappresenta una data e pertanto va codificata come tale.

- In **R** la conversione viene eseguita dal comando `as.Date`, seguita dal **formato** in cui è espressa la data:

```
calcio$Date <- as.Date(calcio$Date, format = "%Y-%m-%d")
class(calcio$Date)
# [1] "Date"

calcio$Date[1:10]
#  [1] "2014-08-30" "2014-08-30" "2014-08-31" "2014-08-31" "2014-08-31" "2014-08-31"
#  [7] "2014-08-31" "2014-08-31" "2014-08-31" "2014-08-31"
```

- Una volta convertito in formata `Date`, è possibile svolgere alcune operazioni aggiuntive. Ad esempio:

```
min(calcio$Date) # Prima partita giocata
# [1] "2008-08-30"
max(calcio$Date) # Ultima partita giocata
# [1] "2015-05-31"
```

Manipolazione di un dataset

- Per **selezionare le righe** di un dataset si procede come nel caso delle matrici:

```
calcio[c(1806, 501, 109), ]
          Date HomeTeam AwayTeam  FTR B365H B365D B365A     Draw
# 1806 2011-03-13    Genoa  Palermo Home  1.91  3.25   4.2 Not_Draw
# 501  2013-11-23    Milan    Genoa Draw  1.44  4.20   7.5     Draw
# 109  2014-11-09  Palermo  Udinese Draw  2.15  3.30   3.4     Draw
```

- È ovviamente possibile (e tipicamente molto più utile) selezionare le osservazioni che soddisfino una determinata **condizione**.

- Supponiamo ad esempio di voler identificare le partite terminate con un pareggio:

```
calcio[c(1806, 501, 109), ]
calcio_draw <- calcio[calcio$FTR == "Draw", ]
head(calcio_draw)
#          Date   HomeTeam  AwayTeam  FTR B365H B365D B365A Draw
# 3  2014-08-31   Atalanta    Verona Draw  2.05  3.40   3.6 Draw
# 7  2014-08-31    Palermo Sampdoria Draw  2.20  3.25   3.3 Draw
# 8  2014-08-31   Sassuolo  Cagliari Draw  2.30  3.40   3.0 Draw
# 9  2014-08-31     Torino     Inter Draw  3.10  3.30   2.3 Draw
# 14 2014-09-14 Fiorentina     Genoa Draw  1.57  3.80   6.0 Draw
# 21 2014-09-20     Cesena    Empoli Draw  2.80  3.00   2.7 Draw
```

I valori mancanti

- Supponiamo di voler identificare le partite aventi in cui la squadra di casa ha una bassissima probabilità di vittoria secondo Bet365:

```
calcio_home <- calcio[calcio$B365H > 9, ]
calcio_home
#            Date HomeTeam AwayTeam  FTR B365H B365D B365A     Draw
# 162  2015-01-06   Cesena   Napoli Away   9.5  4.75  1.36 Not_Draw
# 224  2015-02-15   Cesena Juventus Draw  15.0  6.00  1.22     Draw
# NA         <NA>     <NA>     <NA> <NA>    NA    NA    NA     <NA>
# 1520 2010-05-16    Siena    Inter Away  21.0  9.00  1.10 Not_Draw
# NA.1       <NA>     <NA>     <NA> <NA>    NA    NA    NA     <NA>
# NA.2       <NA>     <NA>     <NA> <NA>    NA    NA    NA     <NA>
```

- **Errore**. Qualcosa è andato storto: ci sono dei simboli (`NA`) che non avevamo incontrato finora.

- La ragione per cui il comando produce dei risultati apparentemente strani è dovuto alla presenza di alcuni **dati mancanti**, codificati come `NA` (Not Available).

I valori mancanti

- In particolare, ci sono 3 osservazioni (la 1501, 1846 e la 1848) i cui valori relativi a Bet365 non sono disponibili.

```
calcio[rowSums(is.na(calcio)) > 0, ] # Identifica le righe con valori mancanti
#            Date HomeTeam AwayTeam  FTR B365H B365D B365A     Draw
# 1501 2010-05-09  Bologna  Catania Draw    NA    NA    NA     Draw
# 1846 2011-04-17   Chievo  Bologna Home    NA    NA    NA Not_Draw
# 1848 2011-04-17    Genoa  Brescia Home    NA    NA    NA Not_Draw
```

- La funzione `is.na` produce una matrice **logica** della stessa dimensione di `calcio`, contenente `TRUE` se il valore è mancante e `FALSE` altrimenti.
- La funzione `rowSums(matrice)` produce un vettore i cui valori sono pari di ciascuna riga della `matrice`.
- Di conseguenza, `rowSums(is.na(calcio))` indica quanti valori mancanti sono presenti in ciascuna riga.

Gestire i dati mancanti

- La gestione statistica dei dati mancanti non è tra gli obiettivi di questo corso.
- La soluzione più facile, ma potenzialmente molto pericolosa, consistente semplicemente nel **rimuovere** le righe incomplete tramite il comando `na.omit`:

```
calcio_no_na <- na.omit(calcio)
dim(calcio_no_na)
# [1] 1897    8
```

- Per risolvere il problema originario, ovvero identificare le partite in cui la squadra di casa ha bassa probabilità di vittoria, si può usare `subset`:

```
calcio_home <- subset(calcio, subset = B365H > 9)
calcio_home
#            Date HomeTeam AwayTeam  FTR B365H B365D B365A     Draw
# 162  2015-01-06   Cesena   Napoli Away   9.5  4.75  1.36 Not_Draw
# 224  2015-02-15   Cesena Juventus Draw  15.0  6.00  1.22     Draw
# 1520 2010-05-16    Siena    Inter Away  21.0  9.00  1.10 Not_Draw
```

- Il comando `subset` implicitamente rimuove i valori mancanti.

Selezione delle variabili

- Il comando `subset` può essere usato sia per **selezionare le righe** (osservazioni) che per **selezionare le colonne** (variabili).

- Nel secondo caso, si può procedere tramite l'opzione `select`:

```
calcio_B365 <- subset(calcio, select = c(B365H, B365D, B365A))
head(calcio_B365)
#   B365H B365D B365A
# 1  7.00   4.0  1.50
# 2  1.67   3.8  5.00
# 3  2.05   3.4  3.60
# 4  3.10   3.2  2.35
# 5  3.90   3.4  1.95
# 6  2.00   3.4  3.75
```

- Ovviamente è possibile anche selezionare **contemporaneamente** sia le righe che le colonne.

Il comando `attach`

- In **R** esiste un comando chiamato `attach`. Sebbene usato da molti, questo comando è **diabolico** e sarebbe meglio evitare di utilizzarlo.
- Il comando `attach(dataframe)` consente di utilizzare le variabili di un `data.frame` come se queste fossero presenti nel workspace.
- Questa pratica tuttavia rende il codice molto meno leggibile e spesso conduce ad errori di varia natura.
- Nonostante le varie proteste a favore della rimozione di `attach` da parte di alcuni, questo comando continua a (r)esistere e viene tuttora usato da molti.
- In questo corso non verranno forniti esempi di utilizzo di `attach` per evitare di indurre lo studente in tentazione.

Sommario

- A seguito di tutte queste operazioni, il dataset risulta molto modificato rispetto alla sua versione iniziale.

- Ri-eseguendo il comando `str` otteniamo infatti che

```
str(calcio)
# 'data.frame':        1900 obs. of  8 variables:
#  $ Date    : Date, format: "2014-08-30" "2014-08-30" "2014-08-31" ...
#  $ HomeTeam: Factor w/ 28 levels "Atalanta","Bari",..: 8 22 1 7 11 17 19 24 26 27 ...
#  $ AwayTeam: Factor w/ 28 levels "Atalanta","Bari",..: 13 10 28 20 18 14 23 5 12 9 ...
#  $ FTR     : Factor w/ 3 levels "Away","Draw",..: 1 3 2 3 1 3 2 2 2 3 ...
#  $ B365H   : num  7 1.67 2.05 3.1 3.9 2 2.2 2.3 3.1 1.8 ...
#  $ B365D   : num  4 3.8 3.4 3.2 3.4 3.4 3.25 3.4 3.3 3.5 ...
#  $ B365A   : num  1.5 5 3.6 2.35 1.95 3.75 3.3 3 2.3 4.5 ...
#  $ Draw    : Factor w/ 2 levels "Not_Draw","Draw": 1 1 2 1 1 1 2 2 2 1 ...
```