

Attività di allineamento

Laboratorio Informatico-Statistico



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Corso di Laurea Magistrale in Statistica Economia e Impresa

Dipartimento di Scienze Statistiche Paolo Fortunati

Anno Accademico 2021/2022



First Steps

Importare dati esterni

L'elaborazione di dati esterni è la principale funzione di R. Proprio per questo motivo è possibile importare file di diversa natura, dai file di testo separati da tabulazioni a quelli prodotti da altri software (es: SAS).

La funzione più utilizzata per caricare dati esterni è *read.table()* della libreria *utils*:

```
> df = read.table(file.choose())
```

L'unico argomento obbligatorio della funzione *read.table* è il path del file da caricare. Per comodità si usa al suo posto la funzione *file.choose()* che fa apparire il classico pop-up di navigazione dei file.

Una volta aperto il file *esempio.csv* lo si analizza chiamandolo nella console:

```
> df
```

	V1
1	id; sesso; anni; peso; altezza
2	MT ; M ; 69 ; 76 ; 1.78
3	GF ; F ; 56 ; 63 ;
4	MC ; F ; 53 ; 71 ; 1.60
5	SB ; M ; 28 ; 73 ; 1.78
6	FE ; F ; 61 ; 54 ; 1.54
7	AB ; M ; 46 ; 92 ; 1.84
8	RF ; F ; 31 ; 81 ; 1.56

Il file risulta essere evidentemente corrotto.

L'errato caricamento del file è dovuta alla non specificazione degli altri argomenti che risultano essere necessari per questo tipo di file.

First Steps

Importare dati esterni

Per importare correttamente il file è necessario specificare la presenza dei nomi delle colonne nella prima riga del file (*header*) ed il separatore (;):

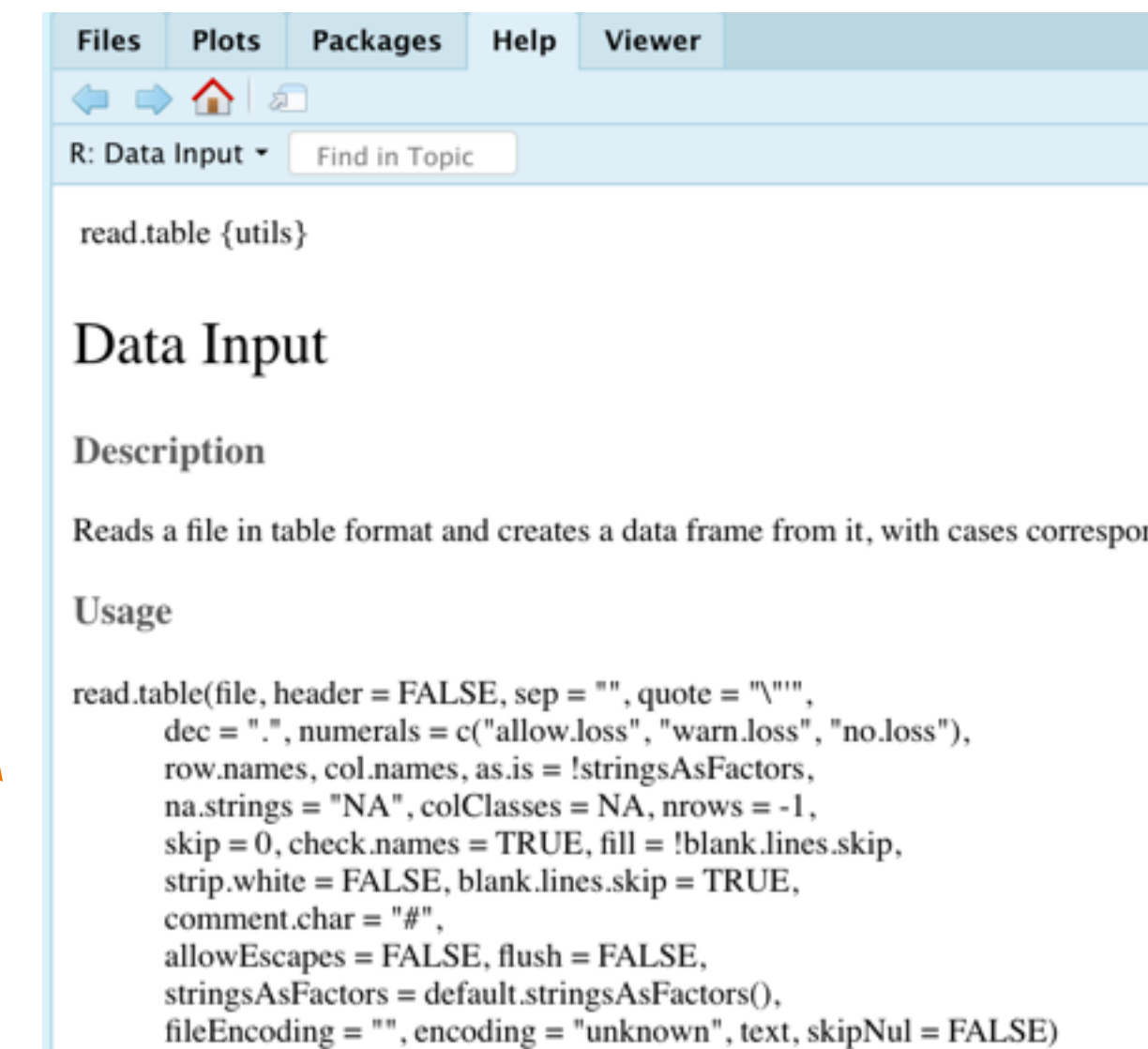
```
> df = read.table(file.choose(), sep=";", header=TRUE )
```

```
> df
```

	id	sex	age	weight	height
1	MT	M	69	76	1.78
2	GF	F	56	63	NA
3	MC	F	53	71	1.60
4	SB	M	28	73	1.78
5	FE	F	61	54	1.54
6	AB	M	46	92	1.84
7	RF	F	31	81	1.56

Si noti che oltre al *sep* e *header* sono presenti molti altri argomenti nella funzione *read.table*, proprio per la moltitudine di file che possono essere importati.

```
> ?read.table
```



The screenshot shows the R help page for the `read.table` function. The window has tabs for Files, Plots, Packages, Help, and Viewer. The 'Help' tab is active, showing the 'R: Data Input' section. The 'Data Input' section includes a 'Description' and a 'Usage' section. The 'Usage' section shows the full signature of the `read.table` function with all its arguments and their default values.

```
read.table {utils}
```

Data Input

Description

Reads a file in table format and creates a data frame from it, with cases correspor

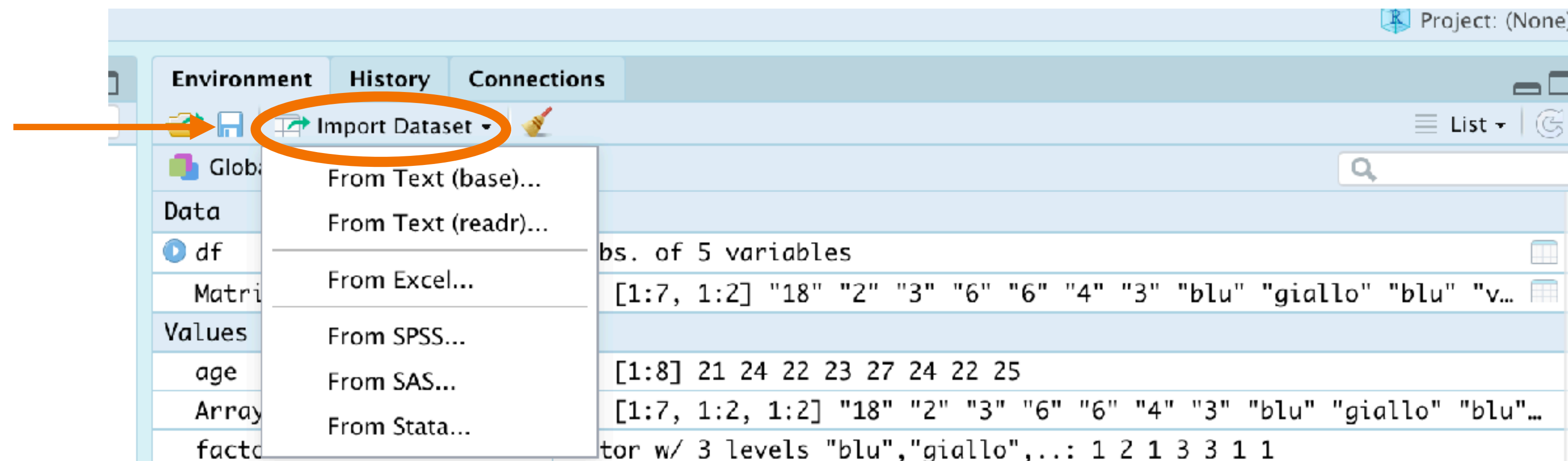
Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"",  
  dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),  
  row.names, col.names, as.is = !stringsAsFactors,  
  na.strings = "NA", colClasses = NA, nrows = -1,  
  skip = 0, check.names = TRUE, fill = !blank.lines.skip,  
  strip.white = FALSE, blank.lines.skip = TRUE,  
  comment.char = "#",  
  allowEscapes = FALSE, flush = FALSE,  
  stringsAsFactors = default.stringsAsFactors(),  
  fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

First Steps

Importare dati esterni

Come gran parte delle funzioni di bas, l'import dei dati può essere eseguito sia da console che in modo grafico sfruttando l'apposito tab in Rstudio.



First Steps

Importare dati esterni

Row names

Automatic

Automatic

Use first column

Use numbers

Separator

Semicolon

Whitespace

Comma

Semicolon

Tab

Comments

None

None

#

!

%

@

/

~

Decimal

Period

Period

Comma

Import Dataset

Name

esempio

Encoding

Automatic

Heading

Yes

No

Row names

Automatic

Separator

Semicolon

Decimal

Period

Quote

Double quote (")

Comment

None

na.strings

NA

☒ Strings as factors

Input File

id; sesso; anni; peso; altezza
MT ; M ; 69 ; 76 ; 1.78
GF ; F ; 56 ; 63 ;
MC ; F ; 53 ; 71 ; 1.60
SB ; M ; 28 ; 73 ; 1.78
FE ; F ; 61 ; 54 ; 1.54
AB ; M ; 46 ; 92 ; 1.84
RF ; F ; 31 ; 81 ; 1.56

Data Frame

id	sesso	anni	peso	altezza
MT	M	69	76	1.78
GF	F	56	63	NA
MC	F	53	71	1.60
SB	M	28	73	1.78
FE	F	61	54	1.54
AB	M	46	92	1.84
RF	F	31	81	1.56

Import

Cancel

Environment

History

Connections

View(dataframe)

rm(dataframe)

t

Data manipulation

Data Exploration

Dataframe Manipulation

Una volta importato il df si passa alla fase di esplorazione dati.

Tramite la funzione *str* si visualizzano i formati delle colonne:

> View(df)

	id	sex	age	weight	height
1	MT	M	69	76	1.78
2	GF	F	56	63	NA
3	MC	F	53	71	1.60
4	SB	M	28	73	1.78
5	FE	F	61	54	1.54
6	AB	M	46	92	1.84
7	RF	F	31	81	1.56

> str(df)

```
data.frame': 7 obs. of 5 variables:
 $ id      : Factor w/ 7 levels "AB ", "FE ", "GF ",...: 5 3 4 7 2 1 6
 $ sesso   : Factor w/ 2 levels " F ", " M ": 2 1 1 2 1 2 1
 $ anni    : num  69 56 53 28 61 46 31
 $ peso    : num  76 63 71 73 54 92 81
 $ altezza: num  1.78 NA 1.6 1.78 1.54 1.84 1.56
```

La funzione *summary()* fornisce invece una descrizione delle variabili

> str(df)

id	sex	age	weight	height
AB :1	F :4	Min. :28.00	Min. :54.00	Min. :1.540
FE :1	M :3	1st Qu.:38.50	1st Qu.:67.00	1st Qu.:1.570
GF :1		Median :53.00	Median :73.00	Median :1.690
MC :1		Mean :49.14	Mean :72.86	Mean :1.683
MT :1		3rd Qu.:58.50	3rd Qu.:78.50	3rd Qu.:1.780
RF :1		Max. :69.00	Max. :92.00	Max. :1.840
SB :1				NA's :1

Data Exploration

Dataframe Manipulation

Per visualizzare il numero di NA su ogni colonna:

```
> colSums(is.na(df))
```

```
      id      sesso      anni      peso altezza  
      0         0         0         0         1
```

Gli NA sono problematici, ad esempio se volessimo calcolare la media dell'altezza:

```
> mean(df$altezza)
```

```
[1] NA
```

Per ovviare a questo problema usiamo l'argomento *na.rm*:

```
> mean(df$altezza, na.rm=T)
```

```
[1]1,68333
```

Per eliminare gli NA ed eseguire la maggior parte delle operazioni di data manipulation ci avvaliamo del pacchetto *dplyr*.

Per installare ed importare *dplyr* utilizziamo il pacchetto *tidyverse* che contiene al suo interno anche altre librerie e funzioni utili:

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

```
> library("tidyverse")
```



Data Exploration

Dataframe Manipulation

Per eliminare gli NA utilizziamo la funzione *drop_na()*:

```
> df = drop_na(df)
```

La libreria *dplyr* permette di filtrare il dataframe in maniera agile:

```
> filter(df, sesso == "F")
```

Ovviamente la condizione di filtro può essere più complessa:

```
> filter(df, sesso == "F" | altezza > 1.78)
```

Per selezionare le righe per posizione si usa invece *slice()*:

```
> slice(df, 1:3)
```

Data Exploration

Dataframe Manipulation

Per ordinare il df utilizziamo *arrange()*:

```
> arrange(df, altezza, peso)
```

Il primo argomento è il dataframe, mentre successivamente possiamo inserire le colonne sulle quali vogliamo ordinare il df. Se ne indichiamo più di una, ogni colonna aggiuntiva verrà usata per ordinare i record che hanno pari valore nella colonna precedente. Per ordinare in maniera discendente si utilizza la funzione *desc()* prima del nome della colonna:

```
> arrange(df, desc(altezza), desc(peso))
```

Per selezionare una o più colonne si utilizza *select()*:

```
> select(df, sesso, anni)
```

```
> select(df, anni:altezza)
```

Per rinominare una colonna si utilizza *rename()*:

```
> rename(df, ID = id)
```

Data Exploration

Dataframe Manipulation

Per creare una (o più) nuova colonna si utilizza *mutate()*:

```
> mutate(df, scarto_peso = peso - mean(peso))
```

La funzione *summarise()* collassa il dataframe su una sola riga:

```
> summarise(df, avg_altezza = mean(altezza, na.rm=T))
```

Per campionare *n* righe utilizziamo *sample_n()* se vogliamo specificare il numero di record da restituire o *sample_frac()* se siamo interessati ad una porzione percentuale del df:

```
> sample_n(df, 10)
```

```
> sample_frac(df, 0.2)
```

Per campionare con reinserimento si usa il parametro *replace=T*:

Data Exploration

Dataframe Manipulation

È spesso utile aggregare i dati secondo i livelli dei fattori o gruppi di record di particolare tipologia. Per fare ciò si usa la funzione *groupby()*:

```
> group_by(df, sesso)
```

Per essere utile però la funzione di *groupby* deve essere accompagnata da altre funzioni, come ad esempio applicare la funzione *media* ai gruppi ottenuti precedentemente.

È per questo motivo che si usa l'operatore *pipe* : `%>%`

Per capire il funzionamento dell'operatore *pipe* si pensi agli argomenti delle funzioni di *dplyr* viste finora. Tutte presentano come primo argomento il dataframe su cui applicare la funzione. L'operatore *pipe* sfrutta questa particolarità permettendo di applicare più funzioni allo stesso dataframe.

Data Exploration

Dataframe Manipulation

Per utilizzare l'operatore pipe si dichiara il dataset su cui si lavora e poi si concatenano le funzioni da applicare l'una dopo l'altra con il *pipe*:

```
> df %>% group_by(sezzo) %>% summarise(media = mean(anni))
```

Si noti come l'argomento data delle funzioni *dplyr* non viene più utilizzato se è presente l'operatore pipe