Laboratorio di Statistica e Analisi Dati: Lezione 3

Tommaso C. & Marco G.

9 - 11 Novembre 2016

Recap sugli esercizi

3.3 Determinare (non a mano) il numero di studenti nelle tre fasce di voto: <22, 23-26, >27

Questo esercizio richiedeva la selezione della colonna relativa ai voti, l'applicazione di una condizione booleana e il conteggio dei TRUE

```
esitiEsame
##           voto ripetizioni
## Luigi       28           3
## Francesca   18           2
## Carlo       25           1
## Alberto     21           1
## Nicola      30           2
## Giovanna    22           2
```

Recap sugli esercizi (cont)

Selezione della colonna contenente i voti

```
esitiEsame[, "voto"]
##     Luigi Francesca     Carlo   Alberto    Nicola  Giovanna
##        28        18        25        21        30        22
```

Recap sugli esercizi (cont)

Generazione di una maschera di booleani da usare come filtro

```
esitiEsame[, "voto"] <= 22
##     Luigi Francesca     Carlo   Alberto    Nicola  Giovanna
##     FALSE      TRUE     FALSE      TRUE     FALSE      TRUE
```

Recap sugli esercizi (cont)

Conteggio dei TRUE nella maschera di booleani

```
sum(esitiEsame[, "voto"] <= 22)
## [1] 3
```

Recap sugli esercizi (cont)

3.4 Aggiungere una colonna “fascia” alla matrice, in cui inseriamo 3 per gli studenti in fascia 18-22, 2 per gli studenti in fascia 23-26 e 1 per quelli in fascia 27-30

In questo caso era necessario applicare la maschera booleana (creata come visto sopra) per effettuare l’accesso ad un nuovo vettore “fascia” e assegnare i valori 1,2,3

Recap sugli esercizi (cont)

```
n = length(esitiEsame[,"voto"])
fascia = rep(0, n)
maschera3 = esitiEsame[, "voto"] <= 22
esitiEsame[, "voto"]
##     Luigi Francesca     Carlo   Alberto    Nicola  Giovanna
##        28        18        25        21        30        22
maschera3
##     Luigi Francesca     Carlo   Alberto    Nicola  Giovanna
##     FALSE      TRUE     FALSE      TRUE     FALSE      TRUE
fascia[maschera3] <- 3
fascia
## [1] 0 3 0 3 0 3
```

Cosa abbiamo visto fino ad ora

- Vettori
- Matrici

La principale caratteristica è che queste strutture contengono dati omogenei. Per avere informazioni su un oggetto a di R esiste la funzione mode(a)

```
c = "pippo"
mode(c)
## [1] "character"
vec <- 1:10
mode(vec)
## [1] "numeric"
matr <- matrix(TRUE, nrow=2, ncol=2)
mode(matr)
## [1] "logical"
```

Liste

- Le liste rappresentano un sequenza di oggetti accessibili tramite un indice
- Una lista viene creata con il comando
  $\text{list}\left(\text{componente}_1, \ldots, \text{componente}_n\right)$
- Le componenti possono *non* essere dello stesso tipo o modo
- Gli elementi di una lista possono essere, a loro volta, delle liste
- Le componenti di una lista sono indicizzate, e sono a loro volta delle sottoliste di lunghezza 1
- Una lista è una struttura ricorsiva

```
esempioLista <- list(
                      c("s1", "s2"),
                      1:3,
```

```
                    list(
                      list(TRUE,FALSE,TRUE),
                      "pippo"
                      )
                    )
```

Liste (cont)

```
esempioLista
## [[1]]
## [1] "s1" "s2"
##
## [[2]]
## [1] 1 2 3
##
## [[3]]
## [[3]][[1]]
## [[3]][[1]][[1]]
## [1] TRUE
##
## [[3]][[1]][[2]]
## [1] FALSE
##
## [[3]][[1]][[3]]
## [1] TRUE
##
##
## [[3]][[2]]
## [1] "pippo"
```

Liste (cont)

Notiamo subito che è cambiata la visualizzazione:

- gli elementi sono sono più tutti su una singola riga
- ogni elemento della lista è separato da una riga vuota
- **N.B.:** I vettori come c("s1", "s2") e 1:3 son rappresentati su un'unica riga, nella lista list(TRUE,FALSE,TRUE) gli elementi, pur essendo omogenei, vengono separati da una riga rendendo molto più complessa la visualizzazione.

REGOLA: Quando vi accorgete di avere una sequenza di **elementi omogenei** usate un **vettore**.

Liste (cont)

```
esempioLista <- list(c("s1", "s2"), 1:3, list(c(TRUE,FALSE,TRUE), "pippo"))
esempioLista
## [[1]]
## [1] "s1" "s2"
##
## [[2]]
## [1] 1 2 3
##
## [[3]]
## [[3]][[1]]
## [1]  TRUE FALSE  TRUE
##
## [[3]][[2]]
## [1] "pippo"
```

Liste (cont)

Un buon modo per visulizzare la struttura di una lista *a* (ma anche di altri oggetti) è il comando *str(a)*

```
str(esempioLista)
## List of 3
##  $ : chr [1:2] "s1" "s2"
##  $ : int [1:3] 1 2 3
##  $ :List of 2
##   ..$ : logi [1:3] TRUE FALSE TRUE
##   ..$ : chr "pippo"
```

Accesso agli elementi delle liste

Per accedere agli elementi di una lista si usa [[]]; se poi il nostro elemento è un altro oggetto accessibile (vettori/matrici) possiamo usare [] per accedere ai suoi elementi, se invece l'elemento è una lista dobbiamo usare nuovamente [[]]

```
esempioLista <- list(c("s1","s2"))
esempioLista
## [[1]]
## [1] "s1" "s2"
#esempio di accesso corretto: esempioLista[[1]]
esempioLista[[1]] #questo è un vettore di stringhe
## [1] "s1" "s2"
esempioLista[[1]][2]
## [1] "s2"
#Esempio di accesso errato: esempioLista[1]
esempioLista[1] #questo è una lista contenente un vettore di stringhe
## [[1]]
## [1] "s1" "s2"
esempioLista[1][2]
## [[1]]
## NULL
```

Accesso/modifica agli elementi delle liste

```
esempioLista <- list(c("s1", "s2"), 1:3, list(c(TRUE,FALSE,TRUE), "pippo"))
esempioLista[[3]][[2]]
## [1] "pippo"
esempioLista[[3]][[1]][2]
## [1] FALSE
esempioLista[[3]][[2]] <- c("pippo", "pluto")
esempioLista[[3]][[2]]
## [1] "pippo" "pluto"
```

Vi ricordate la funzione names()?

Ovviamente non poteva mancare l'accesso tramite caratteri. Ci sono due modi per attaccare le etichette ad una lista:

- in fase di creazione
- usando la cara funzione names()

```
li <- list(val = TRUE, vector = c(1,3,7,0,9), m = matrix(1:12,nrow=2))
li
## $val
## [1] TRUE
##
## $vector
## [1] 1 3 7 0 9
##
## $m
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    3    5    7    9   11
## [2,]    2    4    6    8   10   12
```

Ve lo ricordate names()? (cont)

```
li <- list(TRUE, c(1,3,7,0,9), matrix(1:12,nrow=2))
names(li) <- c("val", "vector", "m")
li
## $val
## [1] TRUE
##
## $vector
## [1] 1 3 7 0 9
##
## $m
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    3    5    7    9   11
## [2,]    2    4    6    8   10   12
```

Il risultato è esattamente identico al comando precedente

Accesso tramite nomi

Si può accedere ad un elemento della lista utilizzando "lista$nomeElemento" o "lista[["nomeElemento"]]"

```
# questi due comandi sono uguali
li$m
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    3    5    7    9   11
## [2,]    2    4    6    8   10   12
li[["m"]]
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    3    5    7    9   11
## [2,]    2    4    6    8   10   12
li$vector == li[["vector"]]
## [1] TRUE TRUE TRUE TRUE TRUE
```

Ahhhhhhhhh adesso si spiega questo….

Aggiunta di elementi da una lista

- L'aggiunta di un elemento avviene in coda alla lista usando l'indice pari a "length +1"
- Non è necessario utulizzare le doppie parentesi [[]]

```
li <- list ("pippo")
li[length(li)+1] <- FALSE
li[length(li)+1] <- 2
li
## [[1]]
## [1] "pippo"
##
## [[2]]
## [1] FALSE
##
## [[3]]
## [1] 2
```

Rimozione di elementi da una lista

- La rimozione avviene settando a NULL l'elemento

```
li[[2]] <- NULL
li
## [[1]]
## [1] "pippo"
##
## [[2]]
## [1] 2
```

N.B.

Modifica/aggunta/rimozione di elementi può essere effettuta usando gli operatori "lista[[nomeElemento]]" e "lista$nomeElemento"

Dataframe

- Un data frame può essere considerato come una matrice le cui colonne rappresentano dati eterogenei (a differenza di matrici e vettori che possono contenere solo dati omogenei)

```
frame
##   ripetizioni probSuccesso alunno
## 1           6    0.1000000     A1
## 2           7    0.3333333     A2
## 3           8    0.5666667     A3
## 4           9    0.8000000     A4
```

Elementi di un Dataframe

Le colonne del data frame possono essere costituiti da:

- Vettori (numerici, a caratteri, logici)
- Fattori (???)
- Matrici numeriche
- Liste
- Altri data frame

Fattori

Sono variabili qualitative nominali…

eeeeeeeeh???

Fattori

Sono variabili qualitative nominali…

eeeeeeeeh???

Sono variabili non numeriche (ad esempio gli attributi “pessimo”, “cattivo”, “mediocre”, “buono” e “ottimo”):

- il numero di possibili valori è finito
- possono possedere un ordine, oppure non averlo

Vi ricordate l’indice di eterogeneità di Gini o l’Entropia?

Costruzione (spero non vi capiti mai)

I data frame sono costruiti tramite la funzione data.frame:

```
x <- 6:9
y <- seq(from = 0.1, to = 0.8, length = 4)
z <- paste("A",1:4,sep="")
frame <- data.frame(ripetizioni = x, probSuccesso = y ,alunno= z)
frame
##   ripetizioni probSuccesso alunno
## 1           6    0.1000000     A1
## 2           7    0.3333333     A2
## 3           8    0.5666667     A3
## 4           9    0.8000000     A4
```

Costruzione (spero vi capiti spesso)

Nell'ambito scientifico i dati vengono solitamente salvati in formato ".csv" ("comma separated values") o altri derivati (come il ".tsv" tab separated values).

```
Eta,StatoCivile,Nazione,Reddito
25,Never-married,United-States,<=50K
38,Married-civ-spouse,United-States,<=50K
28,Married-civ-spouse,United-States,>50K
44,Married-civ-spouse,United-States,>50K
34,Never-married,United-States,<=50K
63,Married-civ-spouse,United-States,>50K
24,Never-married,United-States,<=50K
55,Married-civ-spouse,United-States,<=50K
65,Married-civ-spouse,United-States,>50K
```

Costruzione (spero vi capiti spesso)

R mette a disposizione un comando per la lettura dei file .csv (e suoi derivati)

```
reddito <- read.csv(file = "soluzioni/dataset/reddito.csv")
head(reddito)
##   Eta         StatoCivile       Nazione Reddito
## 1  25       Never-married United-States   <=50K
## 2  38  Married-civ-spouse United-States   <=50K
## 3  28  Married-civ-spouse United-States    >50K
## 4  44  Married-civ-spouse United-States    >50K
## 5  34       Never-married United-States   <=50K
## 6  63  Married-civ-spouse United-States    >50K
```

Vincolo sulle righe

Durante la creazione a partire da vettori, questi devono avere la stessa lunghezza. Può capitare che non ci siano dei dati disponibili, ma se il file .csv è formattato correttamente, read.csv() associa NA per i valori vuoti di tipo numerico mentre una stringa vuota nel caso di fattori

```
redditoNA <- read.csv(file = "soluzioni/dataset/redditoValoriNA.csv")
redditoNA
##   Eta        StatoCivile       Nazione Reddito
## 1  25      Never-married United-States   <=50K
## 2  38 Married-civ-spouse United-States
## 3  28 Married-civ-spouse United-States    >50K
## 4  44                    United-States    >50K
## 5  NA           Divorced United-States   <=50K
## 6  63 Married-civ-spouse                  >50K
redditoNA$StatoCivile[4] #Elemento vuoto
## [1]
## Levels:  Divorced Married-civ-spouse Never-married
redditoNA$StatoCivile[5]
## [1] Divorced
## Levels:  Divorced Married-civ-spouse Never-married
```

Accesso agli elememnti

- Niente di nuovo, le opzioni sono:
  - *Accesso tramite indice numerico delle colonne*
  - *Accesso tramite coppia di coordinate*
  - *Accesso tramite il nome delle componenti*
  - *Accesso tramite indice "a caratteri"*

```
redditoNA[[1]]
## [1] 25 38 28 44 NA 63
redditoNA[1, 3]
## [1] United-States
## Levels:  United-States
redditoNA$Eta
## [1] 25 38 28 44 NA 63
redditoNA[["Eta"]]
## [1] 25 38 28 44 NA 63
```

N.B. Gli ultimi 2 casi funzionano se nella prima riga del file sono presenti

gli headers.

Creare sottoinsiemi di dataframe

- Si possono selezionare componenti del dataframe per crearne un sottoinsieme ridotto

```
redditoNA[1, ]# selezionare solo la prima riga
##   Eta   StatoCivile       Nazione Reddito
## 1  25 Never-married United-States   <=50K
redditoNA[c(3,5), 2]#selezionare della terza e della quinta riga la seconda colonna
## [1] Married-civ-spouse Divorced
## Levels:  Divorced Married-civ-spouse Never-married
```

La riga "## Levels:…" indica tutti i valori che è possibile siano contentuti nella colonna

Selezione di membri del dataset mediante espressioni logiche

- Usando dei confronti su delle variabili è possibile selezionare le righe del dataset in cui si verificano le condizioni

```
redditoNA[redditoNA$Eta>50, ] #righe in cui età è maggiore di 50
##    Eta        StatoCivile Nazione Reddito
## NA  NA               <NA>    <NA>    <NA>
## 6   63 Married-civ-spouse            >50K
# i valori NA generano dei problemi
```

Selezione di membri del dataset mediante espressioni logiche

- Usando dei confronti su delle variabili è possibile selezionare le righe del dataset in cui si verificano le condizioni

```
redditoNA[redditoNA$Eta>50, ] #righe in cui età è maggiore di 50
##    Eta        StatoCivile Nazione Reddito
## NA  NA               <NA>    <NA>    <NA>
## 6   63 Married-civ-spouse            >50K
# i valori NA generano dei problemi
# soluzione: filtrarli via
redditoNA[redditoNA$Eta>50 & !is.na(redditoNA$Eta), ]
##   Eta        StatoCivile Nazione Reddito
## 6  63 Married-civ-spouse            >50K
redditoNA[redditoNA$Eta>50 & !is.na(redditoNA$Eta),"Reddito"]
## [1] >50K
## Levels:  <=50K >50K
```

Selezione di membri del dataset mediante espressioni logiche

- Equivalentemente si può usare la funzione subset()

```
subset(redditoNA, Eta>50 & !is.na(Eta) )
##   Eta        StatoCivile Nazione Reddito
## 6  63 Married-civ-spouse            >50K
```

- Se si vogliono selezionare elementi da un insieme si può usare l'operatore %in%:

```
subset(redditoNA, StatoCivile %in% c ("Divorced", "Never-married"))
##   Eta   StatoCivile       Nazione Reddito
## 1  25 Never-married United-States   <=50K
## 5  NA      Divorced United-States   <=50K
```

Esercizio 1

Siano dati i seguenti nominativi di donatori di sangue ed i corrispondenti gruppi sanguigni (GS):

estrarre dai seguenti vettori 100 campioni

```
c("Luca","Mario", "Tommaso", "Marco", "Gianni", "Pino", "Alessia", "Francesca", "Anna", "Stefania", "Viola
c("Rossi", "Neri", "Bianchi", "Verdi", "Brambilla", "Ferrari", "Russo", "Esposito", "Romani", "Marino", "C
c("0", "A", "B", "AB")
```

ed inserire nomi, cognomi e GS in un dataframe con colonne etichettate

1. Rappresentare i gruppi sanguigni in una tabella delle frequenze (assolute)
2. Rappresentare i gruppi sanguigni in una tabella delle frequenze relative

3. Restituire i nomi ed i cognomi delle persone che hanno gruppo sanguigno B
4. Restituire i cognomi delle persone che hanno gruppo sanguigno B e che si chiamano Cristina

N.B. chi verrà sorpreso a scrivere 100 nomi a mano uno per uno, verrà fucilato sul posto.

Esercizio 2

1. Scaricare il dataset mtcars.csv
2. Importare il dataset creando un dataframe
3. Visualizzare la struttura del dataset
4. Visualizzare le prime 6 righe del dataset
5. Convertire i pesi (wt) da libbre a chilogrammi e sovrascrivere il dataframe (attenzione i valori iniziali sono 1000 lb)
6. Stampare la riga dell'auto più pesante
7. Calcolare la tabella delle frequenze assolute del numero dei cilindri (cyl)
8. Determinare le auto che hanno 4 cilindri e più di 70 cavalli (hp) e stamparne modello (model) e peso (wt)(provate sia con subset che

con [])

Esercizio 3

1. Scaricare il dataset atleti.csv
2. Importare il dataset creando un dataframe e controllarne la struttura
3. I dati dei pesi soddisfano le tre condizioni necessarie affinché siano approsimativamente normali? Ross pag. 105
4. I dati dell'età sono simmetrici intorno al valore medio? Ross pag 56

Cambiare directory di lavoro (non strettamente necessario ma comodo)

- Al fine di importare il dataset senza usare un path assoluto è importante conoscere la posizione dell'attuale cartella di lavoro e/o impostare la nuova cartella di lavoro.

Su sistemi unix-based

Poniamo il caso che abbiate scaricato i file del dataset nella directory "/home/pippo/Scaricati/"

```
getwd()
## [1] "/Users/genuzio/Documents"
setwd("/home/pippo/Scaricati/")
list.files()
## [1] "atleti.csv"          "dataset1.csv"        "mtcars.csv"
## [4] "reddito.csv"         "redditoValoriNA.csv" "titanic.csv"
```

RICORDATE: il tasto **TAB** è vostro amico

Cambiare directory di lavoro (non strettamente necessario ma comodo)

- Al fine di importare il dataset senza usare un path assoluto è importante conoscere la posizione dell'attuale cartella di lavoro e/o impostare la nuova cartella di lavoro.

Su sistemi windows

Poniamo il caso che abbiate scaricato i file del dataset nella directory "C:/Users/TommasoR/Downloads/"

```
getwd()
## [1] "C:/Users/TommasoR/Documents"
setwd("C:/Users/TommasoR/Downloads/")
list.files()
## [1] "atleti.csv"          "dataset1.csv"        "mtcars.csv"
## [4] "reddito.csv"         "redditoValoriNA.csv" "titanic.csv"
```

RICORDATE: il tasto **TAB** è vostro amico

Questionario