R e l'analisi dei dati: una gentile introduzione

Gaetano Scaduto

2024-07-10

Introduzione a R per l'analisi dati

Questo documento è pensato per chi non è riuscito per problemi tecnici ad installare R nel proprio pc o si è perso qualcosa durante la lezione. I contenuti principali che verranno trattati a lezione sono anche qui, anche se quello che faremo in classe potrebbe essere più approfondito. Le porzioni di testo che vedrete dentro i box sono dei codici che potete tranquillamente copiare e incollare su RStudio, ma sono uguali a quelli che trovate nello script caricato insieme a questo file. Iniziamo.

Perché imparare R?

La prima domanda che potreste farvi è: perché dovrei voler imparare R? Qui una serie di ragioni per cui secondo me può valer la pena.

- E' molto richiesto ed apprezzato nel mercato del lavoro.
- Imparare R vi libera sostanzialmente dalla necessità di imparare qualunque altro software statistico
- Al contrario di tutti gli altri, è **gratis**
- Vi potrà facilitare in tutta una serie di compiti che ancora non sapete nemmeno di dover affrontare nella vostra vita professionale.

Quali sono i contro di usare R?

- All'inizio, è **difficile da imparare**, soprattutto se non avete mai programmato prima. Bisogna essere pazienti e non arrabbiarsi, piano piano ce la si fa.
- Nella vostra vita professionale potreste far parte di una minoranza, e dovete esser pronti ad essere un po' dei lupi solitari
  - In azienda si usa di più **Tableau, SPSS o Python**
  - In università si usa di più **Stata**
  - Nel pubblico si usa di più **Excel/SPSS**
- Forse verremmo tutti spazzati via da ChatGTP, boh

Un consiglio che vi posso dare per chi avesse voglia è seguire questo libro qui, che è disponibile, in una versione leggermente più datata, anche in italiano. Se vi studiate questo libro (che è fatto super bene ed è **gratis e disponibile online**, come tutto quello che viene fatto dalla splendida commuity di R) non vi ferma nessuno. Ed è molto facile da studiare da soli. Iniziamo!

Nota sull'utilizzo di ChatGPT e simili

Per quanto mi riguarda, da ora e per tutto il corso, sentitevi liberi di utilizzare **ChatGPT** (o qualunque altro modello di intelligenza artificiale) per risolvere ogni problema che dovesse saltare fuori con R. In particolare, ogni volta che riscontrate un errore:

- Prima, leggete l'errore dal vostro computer e vedete se riuscite a capire cosa succede
- Poi, chiedete a ChatGPT di spiegarvi l'errore, con una formula come questa.
- Se anche dopo la risposta di ChatGPT non siete in grado di capire l'errore, scrivetemi una mail (gaetano.scaduto@unimib.it)

Notate anche che potrete usare ChatGPT in qualunque fase di questo laboratorio. Non è assolutamente un problema, anzi è incoraggiato. Ciononostante, voglio che siate **trasparenti** ogni volta che lo utilizzate è bene che lo indichiate esplicitamente (ad esempio, con un #commento) e che siate in grado di **spiegare** il funziomento del codice.

Primissimi passi

Comandi di Base

Iniziamo con alcuni comandi di base in R.

Operazioni Aritmetiche

```
# Somma
5 + 3
```

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

```
# Sottrazione
10 - 4
```

```
## [1] 6
```

```
# Moltiplicazione
7 * 3
```

```
## [1] 21
```

```
# Divisione
20 / 5
```

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

Operatori di Confronto

```
# Uguaglianza
5 == 5
```

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

```
# Differenza
5 != 3
```

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

```
# Maggiore di
7 > 5
```

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

```
# Minore di
3 < 8
```

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

```r
# Maggiore o uguale a
6 >= 6
```

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

```r
# Minore o uguale a
4 <= 9
```

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

Operatori Logici

Gli operatori logici ci permettono di combinare più condizioni.

```r
# AND
TRUE & FALSE
```

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

```r
# OR
TRUE | FALSE
```

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

```r
# NOT
!TRUE
```

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

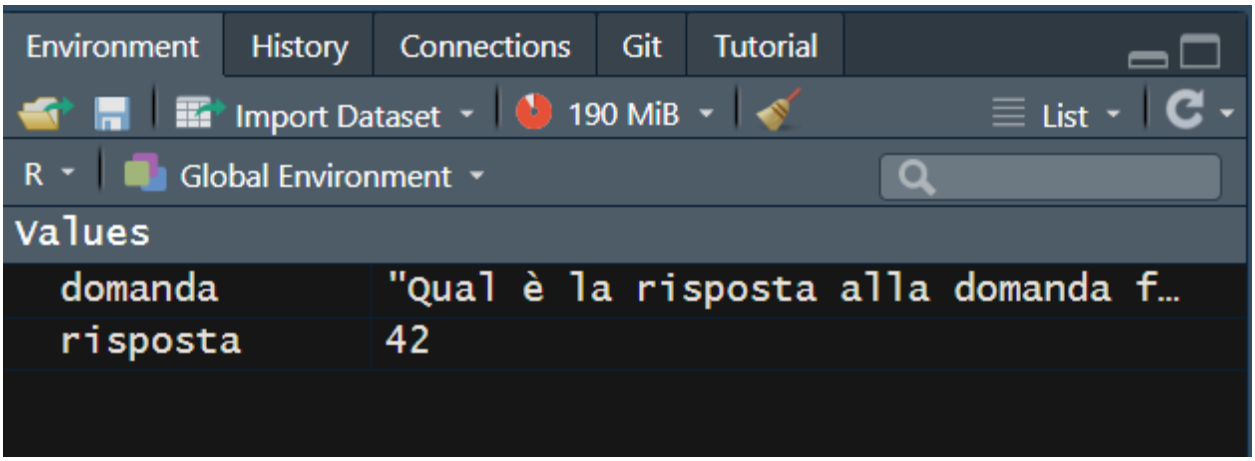
Strutture di Dati: Variabili

Variabili In R, possiamo memorizzare valori in **variabili**.

```r
# Variabile numerica
risposta <- 42

# Stringa
domanda <- "Qual è la risposta alla domanda fondamentale sulla vita, l'universo e tutto quanto?"
```

Quando memorizzate un valore in una variabile, questa comparirà nel vostro **environment** di Rstudio (di solito, in alto a destra)



Una volta che un valore è salvato in una variabile, tutte le volte che avrete bisogno di "evocare" il valore contenuto nella variabile, basterà chiamare il nome della variabile stessa. Ad esempio, se volessi porre questa domanda....

```
domanda
```

```
## [1] "Qual è la risposta alla domanda fondamentale sulla vita, l'universo e tutto quanto?"
```

Potrei poi darmi una risposta chiamando la variabile ***risposta***

```
risposta
```

```
## [1] 42
```

In più, posso fare con le variabili tutte le operazioni di cui abbiamo parlato in precedenza. Per esempio, consideriamo queste due variabili.

```
pigreco=3.1415
costante_gravitazionale=9.81
```

Posso fare tante cose con queste due variabili: sommarle, moltiplicarle, confrontarle...

```
pigreco+costante_gravitazionale
```

```
## [1] 12.9515
```

```
pigreco*costante_gravitazionale
```

```
## [1] 30.81812
```

```
pigreco>costante_gravitazionale
```

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

Strutture di Dati: Vettori

Un vettore è una **sequenza di dati** dello stesso tipo (esempio: numerico, testuale). Per creare un vettore si utilizza il comando ***c(),*** inserendo gli elementi del vettore separati da una virgola.

```
# Vettore numerico
voti <- c(30, 17, 24, 15, 19)

voti
```

```
## [1] 30 17 24 15 19
```

Per chiamare uno specifico valore all'interno del vettore, si usano le parentesi quadre e si indica, con un numero, la posizione dell'elemento che vogliamo estrarre. Ad esempio, se volessi sapere cosa contiene il primo elemento del vettore ***voti,*** dovrei scrivere:

```
voti[1]
```

```
## [1] 30
```

Mentre se volessi il primo ed il terzo elemento, dovremmo mettere un vettore dentro il vettore...

```
voti[c(1,3)]
```

```
## [1] 30 24
```

E se volessi sommarli...

```
voti[1]+voti[3]
```

```
## [1] 54
```

I vettori possono anche avere valori non numerici. Ad esempio, possiamo creare un vettore di stringhe

```r
# Vettore di stringhe
esiti <- c("promosso", "bocciato", "promosso", "bocciato", "promosso")
```

Anche qui, possiamo chiamare i singoli elementi come prima.

```r
esiti[1]
```

```
## [1] "promosso"
```

```r
esiti[3]
```

```
## [1] "promosso"
```

```r
esiti[c(1,3)]
```

```
## [1] "promosso" "promosso"
```

Il Dataframe

Un DataFrame è una struttura di dati fondamentale in R che è molto simile a una tabella di dati. Ogni colonna di un DataFrame può contenere un diverso tipo di dati (numerico, stringhe, ecc.), e tutte le colonne **devono avere la stessa lunghezz**a.

Potete creare un DataFrame utilizzando la funzione ***data.frame()***. Ecco un esempio:

```r
# Creazione di un DataFrame con dati fittizi
dati <- data.frame(
  nomi = c("Anna", "Luca", "Marco"),
  eta = c(28, 34, 29),
  città = c("Roma", "Milano", "Napoli")
)
```

Per visualizzare il dataframe abbiamo varie opzioni, la più completa è la funzione ***View()***, che vi apre direttamente una finestra su Rstudio dove potete esplorare l'intero dataframe

```r
View(dati)
```

Accesso ai Dati in un DataFrame

Potete accedere ai dati in un DataFrame in vari modi:

```r
# Visualizzare la colonna "nomi"
dati$nomi
```

Accesso tramite nome della colonna

```
## [1] "Anna"  "Luca"  "Marco"
```

```r
# Visualizzare la colonna "eta"
dati$eta
```

```
## [1] 28 34 29
```

```r
# Visualizzare la prima colonna
dati[,1]
```

Accesso tramite indici

```
## [1] "Anna"  "Luca"  "Marco"
```

```r
# Visualizzare la prima riga
dati[1,]
```

```
##   nomi eta città
## 1 Anna  28  Roma
```

```r
# Visualizzare l'elemento nella prima riga e nella seconda colonna
dati[1, 2]
```

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

NOTA BENE: il primo indice è sempre l'indice di riga, il secondo indice è sempre l'indice di colonna

Modifica di un DataFrame

Potete aggiungere o modificare colonne e righe in un DataFrame.

```r
# Aggiungere una nuova colonna "sesso"
dati$sesso <- c("F", "M", "M")
# Visualizzare il DataFrame aggiornato
View(dati)
```

Aggiungere una colonna

```r
# Modificare la colonna "eta"
dati$eta <- c(29, 35, 30)  # Visualizzare il DataFrame aggiornato
View(dati)
```

Modificare una colonna esistente

Aggiungere una riga Per aggiungere una riga, bisogna creare un nuovo dataset e successivamente utilizzare la funzione ***rbind()***.

```r
# Creare una nuova riga
nuova_riga <- data.frame(   nomi = "Giulia",   eta = 26,   città = "Torino",   sesso = "F" )
# Aggiungere la nuova riga al DataFrame esistente
dati <- rbind(dati, nuova_riga)  # Visualizzare il DataFrame aggiornato
View(dati)
```

Operazioni sui DataFrame

I DataFrame in R supportano una vasta gamma di operazioni. Ecco alcune delle più comuni:

```r
# Filtrare le righe dove "eta" è maggiore di 30
dati_filtrati <- dati[dati$eta > 30, ]
print(dati_filtrati)
```

```
##   nomi eta  città sesso
## 2 Luca  35 Milano     M
```

```r
# Ordinare i dati in base alla colonna "eta"
dati_ordinati <- dati[order(dati$eta), ]
print(dati_ordinati)
```

Ordinare i dati

```
##     nomi eta  città sesso
## 4 Giulia  26 Torino     F
## 1   Anna  29   Roma     F
## 3  Marco  30 Napoli     M
## 2   Luca  35 Milano     M
```

```r
# Selezionare solo le colonne "nomi" e "eta"
dati_selezionati <- dati[, c("nomi", "eta")]
print(dati_selezionati)
```

Selezionare colonne specifiche

```
##     nomi eta
## 1   Anna  29
## 2   Luca  35
## 3  Marco  30
## 4 Giulia  26
```

Esempio Pratico

Mettiamo insieme tutto ciò che abbiamo imparato in un esempio pratico.

```r
# Creazione di un DataFrame con informazioni sui partecipanti a un sondaggio

partecipanti <- data.frame(
  ID = 1:5,
  Nome = c("Alice", "Bob", "Charlie", "David", "Eve"),
  Età = c(24, 30, 22, 28, 35),
  Città = c("Roma", "Milano", "Napoli", "Torino", "Firenze"),
  Genere = c("F", "M", "M", "M", "F")
  )

# Visualizzare il DataFrame
print(partecipanti)
```

```
##   ID    Nome Età   Città Genere
## 1  1   Alice  24    Roma      F
## 2  2     Bob  30  Milano      M
## 3  3 Charlie  22  Napoli      M
## 4  4   David  28  Torino      M
## 5  5     Eve  35 Firenze      F
```

```r
# Filtrare i partecipanti con età maggiore di 25
partecipanti_over_25 <- partecipanti[partecipanti$Età > 25, ]

print(partecipanti_over_25)
```

```
##   ID  Nome Età   Città Genere
## 2  2   Bob  30  Milano      M
## 4  4 David  28  Torino      M
## 5  5   Eve  35 Firenze      F
```

```r
# Ordinare i partecipanti in base all'età

partecipanti_ordinati <- partecipanti[order(partecipanti$Età), ]
```

```r
print(partecipanti_ordinati)
```

```
##   ID    Nome Età   Città Genere
## 3  3 Charlie  22  Napoli      M
## 1  1   Alice  24    Roma      F
## 4  4   David  28  Torino      M
## 2  2     Bob  30  Milano      M
## 5  5     Eve  35 Firenze      F
```

```r
# Selezionare solo i nomi e le città dei partecipanti
nomi_città <- partecipanti[, c("Nome", "Città")]

print(nomi_città)
```

```
##      Nome   Città
## 1   Alice    Roma
## 2     Bob  Milano
## 3 Charlie  Napoli
## 4   David  Torino
## 5     Eve Firenze
```