

R e l'analisi dei dati: una gentile introduzione

Gaetano Scaduto

2023-10-09

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

Questa lezione è solo un assaggio. Purtroppo in Bicocca che io sappia non abbiamo (per ora!) un corso di R. Un consiglio che vi posso dare per chi avesse voglia è seguire questo libro [CLICCA QUI](#). Se vi imparate questo libro (che è fatto super bene ed è **gratis e disponibile online**, come tutto quello che viene fatto dalla splendida community di R) non vi ferma nessuno. Ed è molto facile da studiare da soli. Iniziamo!

Primissimi passi

R si basa su una filosofia open source. Alcune cose le avete “comprese nel pacchetto base”, altre cose le scaricate da un archivio online. Si tratta di “pacchetti” che vi permettono di espandere le funzionalità base di R. Prima di utilizzare un pacchetto per la prima volta, **solo per la prima volta**, dovete installarlo. Per installarlo vi serve una connessione ad internet ed eseguire i comandi di seguito. Noi per questa lezione useremo tre pacchetti **rio**, **ggplot2** e **texreg**

```
install.packages("rio")
install.packages("ggplot2")
install.packages("texreg")
```

Dopo aver installato, dovete chiamare la libreria. Questo va fatto **ogni volta che riaccendete R** con il comando **library**

```
library(rio)
library(ggplot2)
library(texreg)
```

```
## Version:  1.38.6
## Date:     2022-04-06
## Author:   Philip Leifeld (University of Essex)
##
## Consider submitting praise using the praise or praise_interactive functions.
## Please cite the JSS article in your publications -- see citation("texreg").
```

Un ultima finezza. Voglio i numeri con la virgola, non la notazione scientifica. Per farlo, dico a R questa cosa.

```
options(scipen = 999)
```

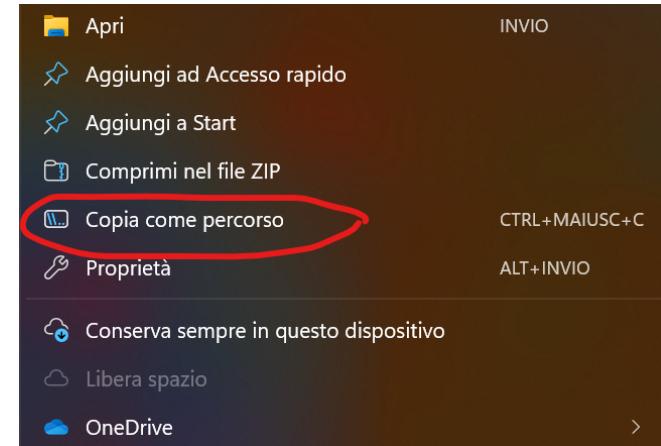
Importare un file

La libreria **rio** (che sta per R Input/Output) vi permette di importare file su R senza sbatti. Nel nostro caso io vi ho preparato un dataset coi **risultati delle elezioni del 2022 su tutti i comuni italiani** più qualche altra info. Questo dataset è basato sugli open data del ministero degli interni che sono stati da me manipolati un po' e uniti con un tot di statistiche dell'Istat. Per importarlo in R dovete fare le seguenti cose:

- Create una cartella da qualche parte nel vostro computer e dategli un nome
- In quella cartella, metteteci il dataset che vi abbiamo caricato online. Potete anche solo copiarlo e incollarlo all'interno di quella cartella.

Dopodiché dovrete dire ad R che state lavorando in quella cartella. Per farlo, dovete eseguire il comando **“setwd()”** (che sta per “set working directory”) e mettere, tra virgolette, il percorso della cartella.

- Per trovare il percorso della cartella, se avete windows basta fare tasto destro sulla cartella e poi “copia come percorso”



- Se avete Mac cercate su google “come copiare percorso cartella Mac”, ma secondo me potete vedere da “proprietà” col tasto destro o qualcosa del genere. Io ho provato a cercare su google ma mica l’ho capito.
 - Se proprio non riuscite: uscite da R e **cliccate sul file del dataset due volte**. Se intendete imparare come si usa R questa cosa non va bene, ma per sta volta passi...
- **Attenzione! Se lo fate su windows, quello che vi copierà sarà così**
 - “C:\Users\gasca\OneDrive - Università degli Studi di Milano-Bicocca\Desktop\cartella esempio”
 - R per uno strano motivo vuole che le stanghette non siano \ ma /, quindi voi dovrete cambiarle manualmente così
 - C:/Users/gasca/OneDrive - Università degli Studi di Milano-Bicocca/Desktop/cartella esempio

Quindi il comando sarà

```
setwd("C:/Users/gasca/OneDrive - Università degli Studi di Milano-Bicocca/Desktop/cartella esempio")
```

Una volta fatto ciò, eseguite il seguente comando (non dimenticate le virgolette ““!)

```
data = import("ris2022_lezione.RDS")
```

- NOTA: se avete fatto doppioclic sul dataset dalla cartella (per quelli che non trovavano come copiare il percorso della cartella), il comando di sopra non eseguitelo!

In questo caso state dicendo ad R: prendi l’oggetto chiamato nel modo dentro le virgolette (dalla cartella che gli abbiamo detto prima), importamelo su R e chiamalo, da ora in poi “data”. “data” è il nome con cui chiameremo i nostri dati. Notate che dovrete sempre riportare il nome del file che volete importare **compreso di estensione**. Con la funzione import() (che fa parte di rio) potete importare senza problemi sostanzialmente qualunque tipo di data file esistente. Anche un foglio excel.

Nota: questo dataset ha qualche dato mancante. Per semplicità noi lavoreremo coi dati completi. Rimuovo quindi tutte le righe che hanno “missing data” con questo comando

```
data = data[complete.cases(data), ]
```

Sbirciamo nei dati

Adesso vogliamo sbirciare nei dati. Possiamo farlo col comando head()

```
head(data)
```

```
##   istat      comune    regione provincia num_residenti superficie Lega   Fi
## 1  1001       AGLIE' Piemonte     TO      2545    13.28 12.15  9.57
## 2  1002       AIRASCA Piemonte    TO      3633    15.70 10.31  7.77
## 3  1003      ALA DI STURA Piemonte     TO      459    46.09 14.35 11.39
## 4  1004  ALBIANO D'IVREA Piemonte     TO      1638    11.73 10.30  7.61
## 5  1006       ALMESE Piemonte     TO      6355    17.91  7.83  6.02
## 6  1007      ALPETTE Piemonte     TO      235     5.65 11.35  1.42
##   PD     AVS     M5S   Aziv   Fdi areeint least_voted macroarea most_voted
## 1 15.06  2.83  9.48 10.32 30.62      C      AVS North-West      fdi
## 2 18.85  3.14 15.47  6.11 26.97      C      AVS North-West      fdi
## 3 13.08  2.95  7.59  4.64 30.80      E      AVS North-West      fdi
## 4 15.81  2.81  6.67  5.97 42.39      C      AVS North-West      fdi
## 5 18.64  5.34 14.78  8.61 22.58      C      AVS North-West      fdi
## 6  3.55 21.28  5.67  5.67 24.11      E      Fi North-West      fdi
```

Se volessimo vedere ancora meglio nei dati, su R fate partire il comando “View(data)” e potete navigare in libertà nel dataset.

Se vogliamo velocemente un sommario di quello che contiene il dataset, possiamo eseguire il seguente comando

```
summary(data)
```

```
##      istat      comune      regione      provincia
## Min.   : 1001  Length:7700  Length:7700  Length:7700
## 1st Qu.: 16171 Class  :character  Class  :character  Class  :character
## Median : 42039 Mode   :character  Mode   :character  Mode   :character
## Mean   : 45932
## 3rd Qu.: 75002
## Max.   :111107
##   num_residenti      superficie      Lega      Fi
## Min.   :    29  Min.   : 0.20  Min.   : 0.00  Min.   : 0.000
## 1st Qu.:  994  1st Qu.: 11.31  1st Qu.: 6.85  1st Qu.: 6.840
## Median : 2438  Median : 22.14  Median :10.80  Median : 8.600
## Mean   : 7598  Mean   : 37.68  Mean   :11.46  Mean   : 9.706
## 3rd Qu.: 6346  3rd Qu.: 43.59  3rd Qu.:15.24  3rd Qu.:11.090
## Max.   :2770226 Max.   :1307.71  Max.   :63.82  Max.   :72.810
##      PD          AVS          M5S          Aziv
## Min.   : 0.00  Min.   : 0.000  Min.   : 0.00  Min.   : 0.000
## 1st Qu.:12.60  1st Qu.: 1.780  1st Qu.: 6.21  1st Qu.: 4.170
## Median :15.88  Median : 2.610  Median : 9.99  Median : 6.165
## Mean   :16.79  Mean   : 2.815  Mean   :13.13  Mean   : 6.350
## 3rd Qu.:19.97  3rd Qu.: 3.520  3rd Qu.:18.00  3rd Qu.: 7.980
## Max.   :67.33  Max.   :36.360  Max.   :85.12  Max.   :55.660
##      Fdi      areeint      least_voted      macroarea
## Min.   : 0.00  Length:7700  Length:7700  Length:7700
## 1st Qu.:23.86  Class  :character  Class  :character  Class  :character
## Median :29.83  Mode   :character  Mode   :character  Mode   :character
```

```
##  Mean    :29.12
##  3rd Qu.:34.59
##  Max.    :65.79
##  most_voted
##  Length:7700
##  Class  :character
##  Mode   :character
##
##
```

Adesso andiamo a vedere qualche statistica nello specifico. Per fare così, dovremo “chiamare” direttamente le colonne del dataset. Per farlo si usa l’operatore \$ dopo il nome del dataset. Rstudio vi suggerirà poi lui stesso i nomi delle colonne disponibili di modo che possiate accedere a queste. Calcoliamoci qualche statistica di base come esempio. Tipo la media del numero di residenti nei comuni del nostro dataset

```
mean(data$num_residenti)
```

```
## [1] 7598.046
```

Oppure la mediana

```
median(data$num_residenti)
```

```
## [1] 2438.5
```

Oppure ci chiediamo: “ma nel comune dove il movimento 5 stelle ha preso la percentuale più alta, ma quanto ha preso?”. La risposta è data da questo comando.

```
max(data$M5S)
```

```
## [1] 85.12
```

Se invece ci chiedessimo quanto è piccolo il più piccolo comune che abbiamo nel dataset...

```
min(data$superficie)
```

```
## [1] 0.2
```

Adesso ci chiediamo se, per esempio, la media del numero di voti presi da Forza Italia nei comuni della Calabria è diversa da quella dei comuni in Umbria

```
mean(data[data$regione == "Calabria", ]$Fi)
```

```
## [1] 18.14832
```

```
mean(data[data$regione == "Umbria", ]$Fi)
```

```
## [1] 7.315217
```

Le tabelle

Adesso vedremo un po' di tabelle. Le tabelle possono essere molto utili per capire come si distribuiscono i nostri dati. Possiamo fare tabelle con una variabile sola o con due variabili, ed in entrambi i casi si usa il comando “table”. Vediamo ad esempio come è distribuita la variabile areeint (codice aree interne istat - indica la perifericità del comune: A è un comune centrale, F è un comune sperduto).

```
table(data$areeint)
```

```
##  
##      A      B      C      D      E      F  
## 180   59 3770 1863 1474   354
```

Se vogliamo creare una tabella di due variabili incrociate, allora basta metterle in table separate da una virgola.

```
table(data$macroarea, data$areeint)
```

```
##  
##          A      B      C      D      E      F  
## Center     38      6    394    319    190     21  
## North-East  45      9    738    237    203     38  
## North-West  47     28   1873    576    326     66  
## South      50     16    765    731    755    229
```

Questa tabella non è molto informativa, possiamo pensare a fare una tabella con le proporzioni con il seguente comando.

```
prop.table(table(data$macroarea, data$areeint))
```

```
##  
##          A          B          C          D          E  
## Center 0.0049350649 0.0007792208 0.0511688312 0.0414285714 0.0246753247  
## North-East 0.0058441558 0.0011688312 0.0958441558 0.0307792208 0.0263636364  
## North-West 0.0061038961 0.0036363636 0.2432467532 0.0748051948 0.0423376623  
## South    0.0064935065 0.0020779221 0.0993506494 0.0949350649 0.0980519481  
##  
##          F  
## Center 0.0027272727  
## North-East 0.0049350649  
## North-West 0.0085714286  
## South    0.0297402597
```

Non si capisce niente! Troppe cifre decimali! Risolviamo con la funzione “round” dicendogli di arrotondare a due cifre soltanto.

```
round(prop.table(table(data$macroarea, data$areeint)), digits = 2)
```

```
##  
##          A      B      C      D      E      F  
## Center 0.00 0.00 0.05 0.04 0.02 0.00  
## North-East 0.01 0.00 0.10 0.03 0.03 0.00  
## North-West 0.01 0.00 0.24 0.07 0.04 0.01  
## South    0.01 0.00 0.10 0.09 0.10 0.03
```

Queste sono le percentuali su tutta la tabella. Forse sarebbe più utile averle per riga o per colonna. Per fare così dobbiamo aggiungere a prop.table un'opzione. 1 corrisponde alle percentuali per riga. 2 alle percentuali per colonna.

```
round(prop.table(table(data$macroarea, data$areeint), 1), digits = 2)
```

```
##  
##          A     B     C     D     E     F  
##  Center    0.04 0.01 0.41 0.33 0.20 0.02  
##  North-East 0.04 0.01 0.58 0.19 0.16 0.03  
##  North-West 0.02 0.01 0.64 0.20 0.11 0.02  
##  South      0.02 0.01 0.30 0.29 0.30 0.09
```

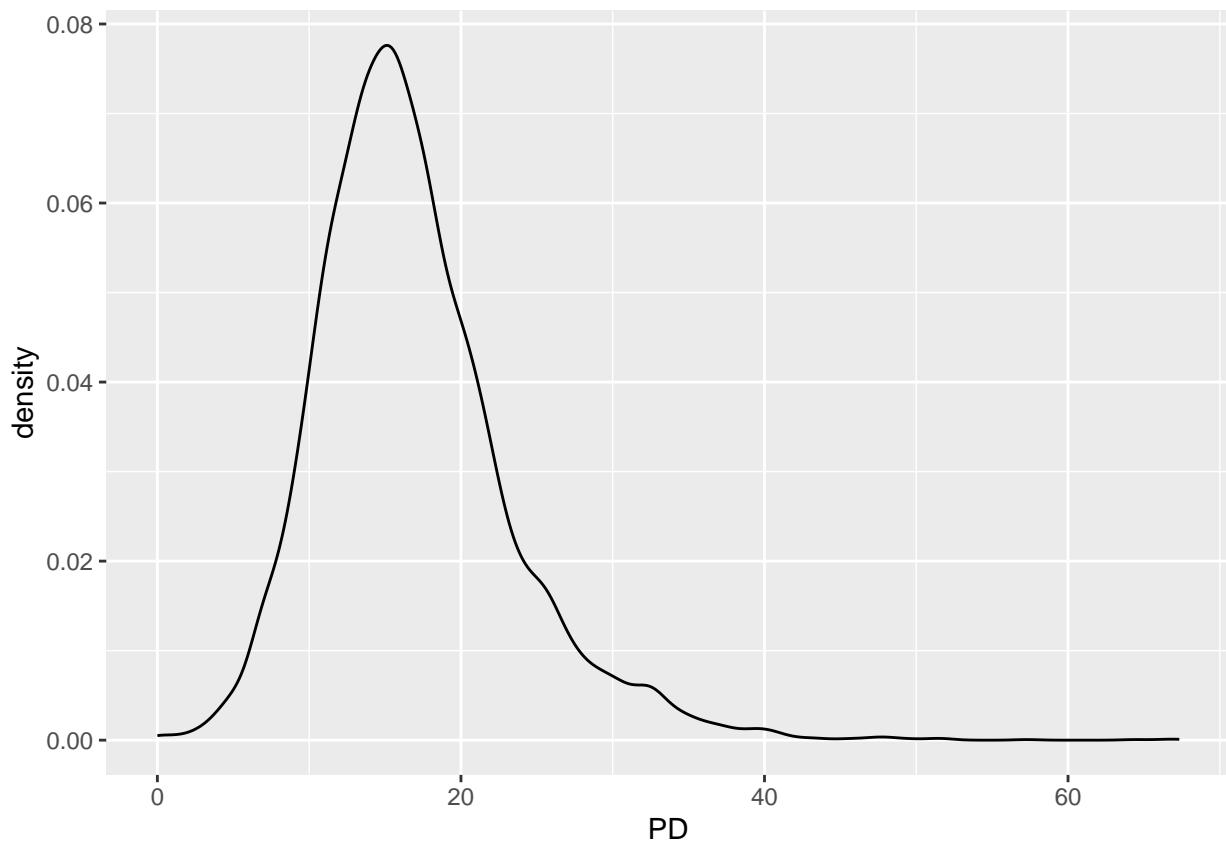
```
round(prop.table(table(data$macroarea, data$areeint), 2), digits = 2)
```

```
##  
##          A     B     C     D     E     F  
##  Center    0.21 0.10 0.10 0.17 0.13 0.06  
##  North-East 0.25 0.15 0.20 0.13 0.14 0.11  
##  North-West 0.26 0.47 0.50 0.31 0.22 0.19  
##  South      0.28 0.27 0.20 0.39 0.51 0.65
```

Grafici

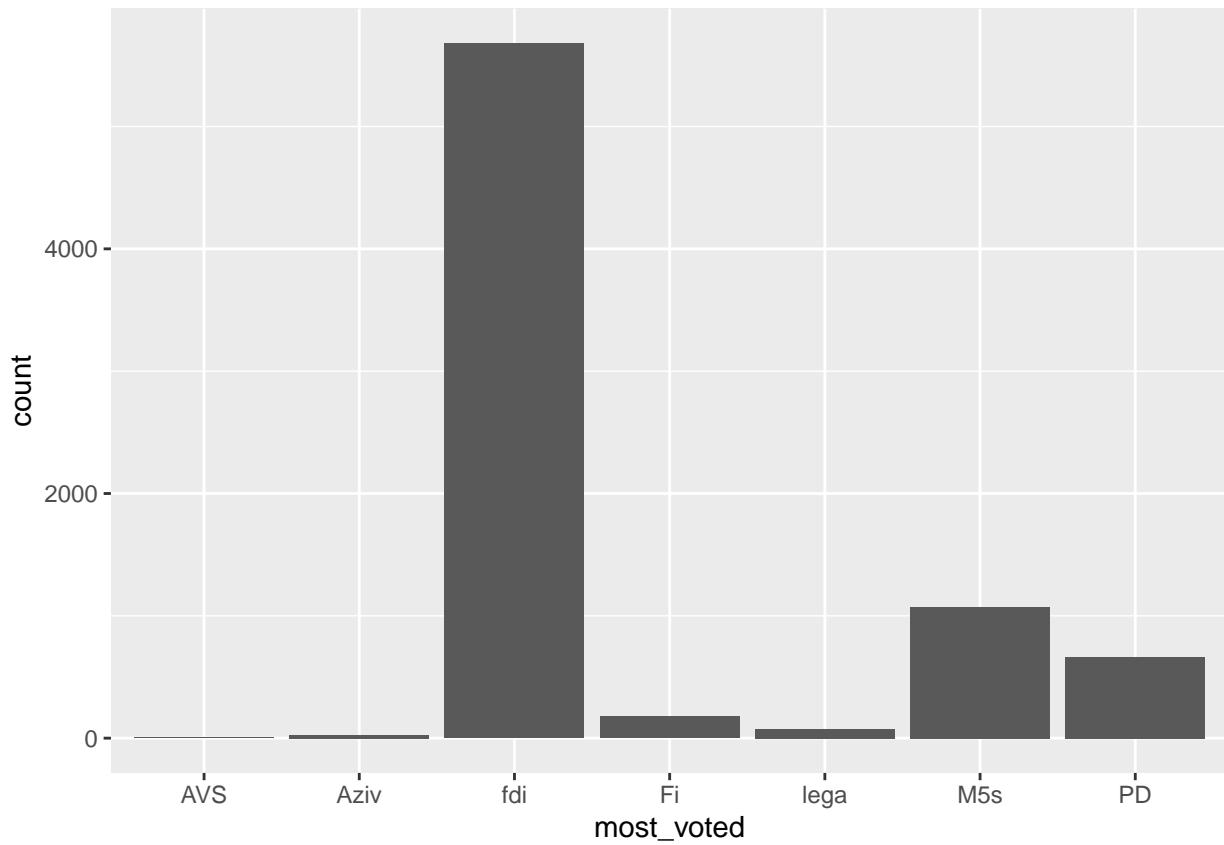
Per i grafici noi usiamo la libreria ggplot2. Secondo me è il modo migliore. Ggplot2 ha un linguaggio un po' criptico. Limitatevi per adesso a copia-incollare i comandi. Per i più audaci in assoluto, gli stessi che hanno scritto il libro su R hanno anche scritto il libro su ggplot2 (QUI) Anche qui, per capire ci vuole tanta pazienza, ma basta cogliere l'idea per adesso. Ad esempio, proviamo a vedere come sono distribuiti i voti al PD.

```
ggplot(data, aes(x=PD)) +  
  geom_density()
```



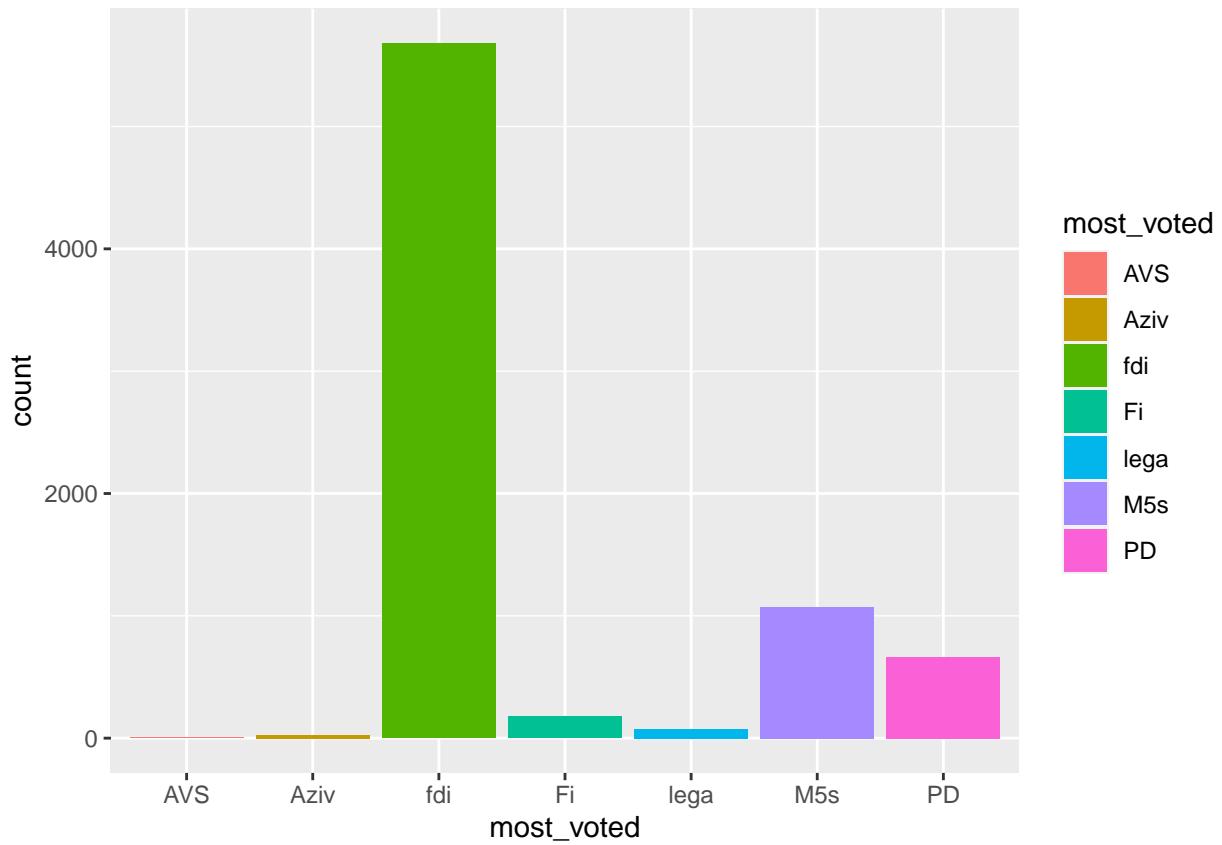
Proviamo a fare un **diagramma a barre** (barplot) sul partito più votato in tutti i comuni

```
ggplot(data, aes(x=most_voted)) +  
  geom_bar()
```



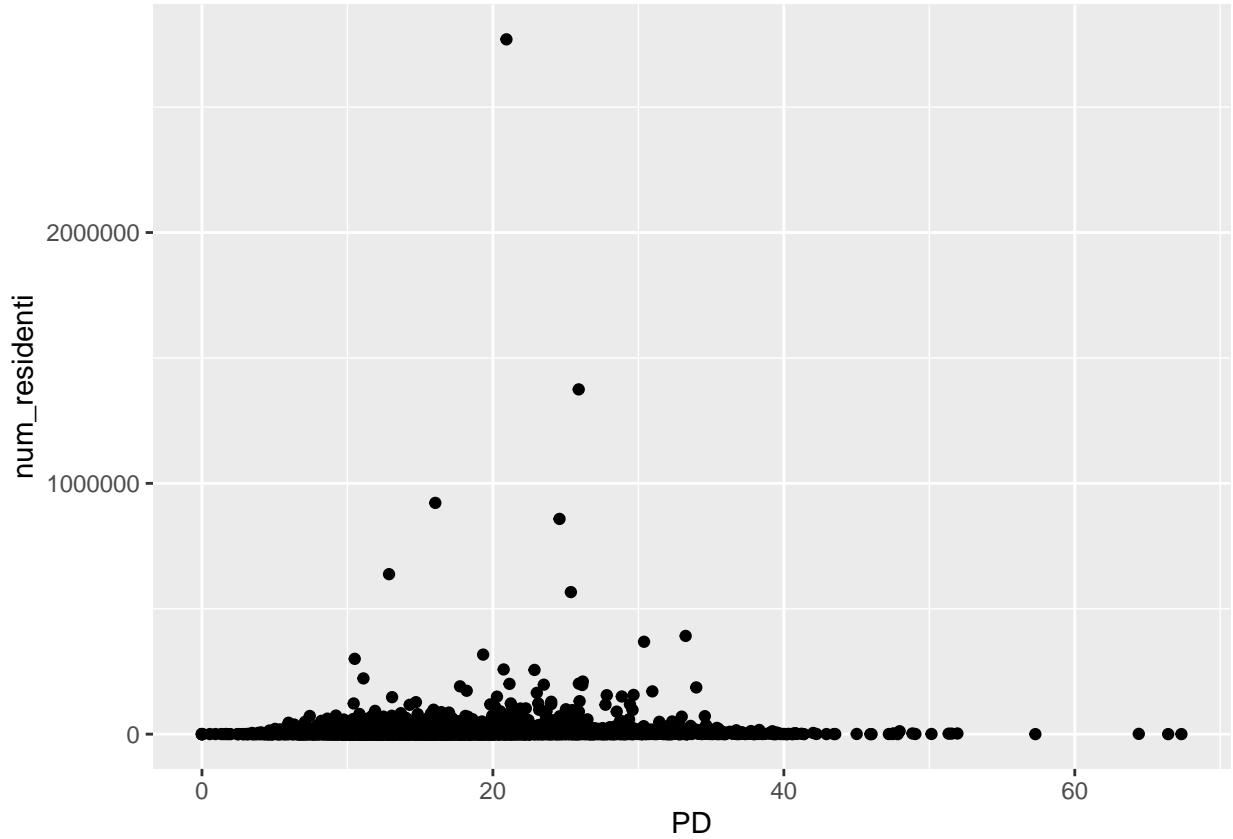
Se le volessi colorate queste barre, scriverei così

```
ggplot(data, aes(x=most_voted))+
  geom_bar(aes(fill=most_voted))
```



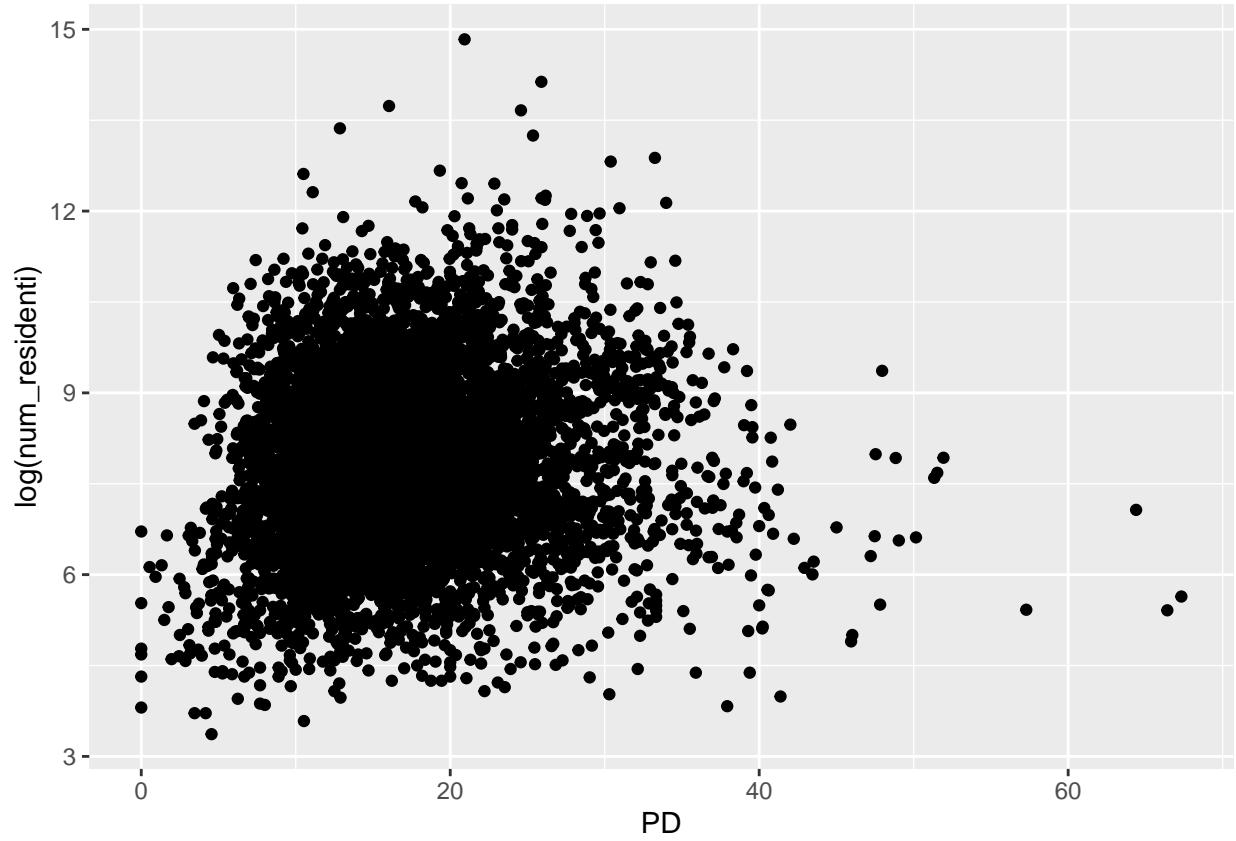
Vogliamo provare a vedere se c'è una correlazione fra il numero di abitanti in un comune ed il fatto che il pd sia il partito più votato. Proviamo con geom_point()

```
ggplot(data, aes(x=PD, y=num_residenti))+
  geom_point()
```



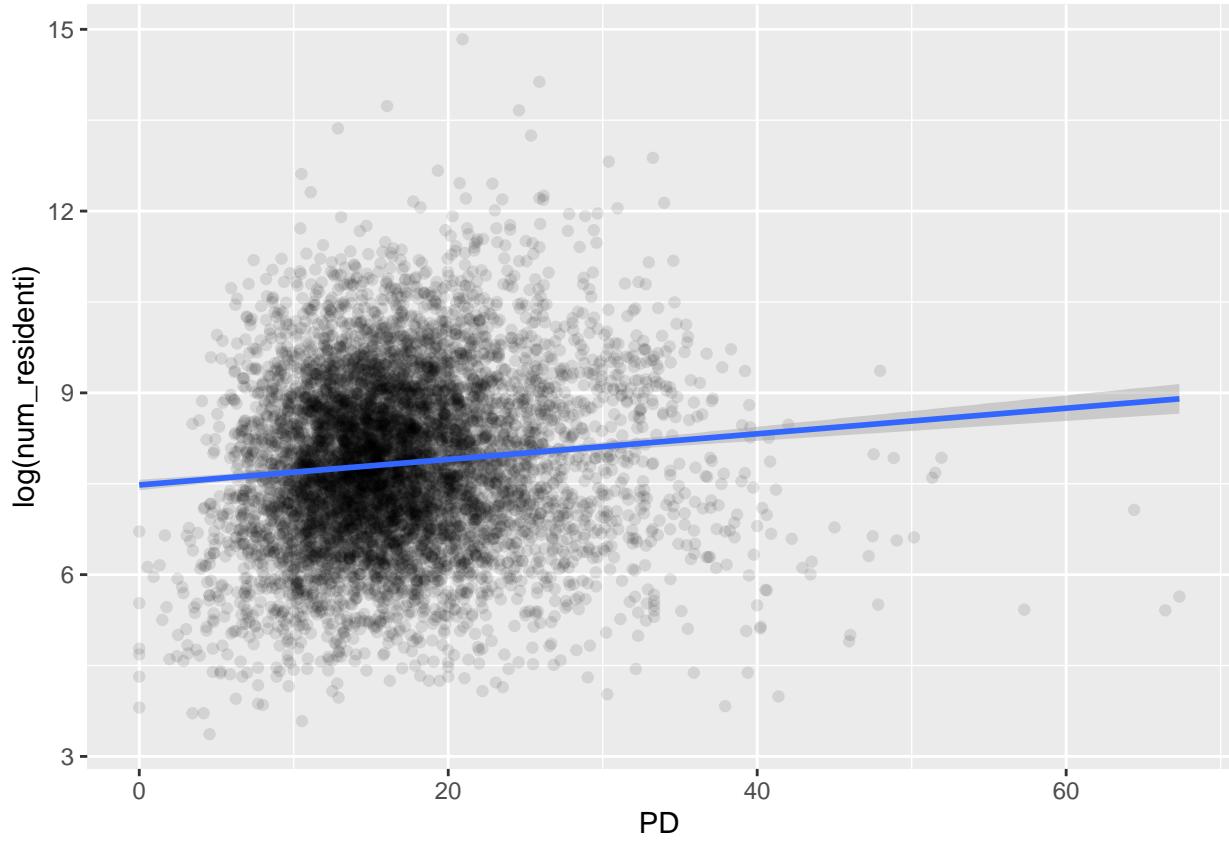
Così non si capisce molto. Posso provare a prendere il logaritmo della variabile numero di residenti.

```
ggplot(data, aes(x=PD, y=log(num_residenti)))+  
  geom_point()
```



Già meglio! Ma se volessi provare a vedere una retta di regressione lì in mezzo?

```
ggplot(data, aes(x=PD, y=log(num_residenti)))+  
  geom_point(alpha=1/10)+  
  geom_smooth(method="lm")  
  
## `geom_smooth()` using formula = 'y ~ x'
```



Carino!

La regressione!

Finiamo la nostra carrellata con qualche regressione. Qui faremo solo regressioni lineari. Supponiamo di voler provare a costruire un modello che spiega il voto al movimento cinque stelle. Facciamo quindi una serie di regressioni lineari, che si fanno col comando “lm”. Il primo argomento di “lm” è “data=nomemiodataset” (nel nostro caso sfortunato il nostro dataset si chiama proprio data). Poi mettete la variabile dipendente, una tilde ~ (Alt+126) e poi le variabili indipendenti. Salvate tutto il risultato in un oggetto e osservate i risultati della vostra regressione mettendo l’oggetto in una funzione screenreg(). Partiamo da un modello celeberrimo: “i cinque stelle partito del sud”

```
regr1 = lm(data=data, M5S ~ macroarea)
screenreg(regr1)
```

```
##
## =====
##                               Model 1
## -----
## (Intercept)            12.60 ***
##                         (0.19)
## macroareaNorth-East -6.37 ***
##                         (0.25)
## macroareaNorth-West -5.30 ***
```

```

##                               (0.21)
## macroareaSouth            10.87 ***
##                               (0.22)
## -----
## R^2                         0.63
## Adj. R^2                     0.63
## Num. obs.                   7700
## -----
## *** p < 0.001; ** p < 0.01; * p < 0.05

```

Domandina: qual è la categoria di riferimento?

Adesso proviamo un modello più sofisticato, che tiene anche in conto la perifericità

```

regr2 = lm(data=data, M5S ~ macroarea + areeint)
screenreg(regr2)

```

```

##
## -----
##                               Model 1
## -----
## (Intercept)             14.77 ***
##                               (0.45)
## macroareaNorth-East   -6.59 ***
##                               (0.24)
## macroareaNorth-West   -5.65 ***
##                               (0.21)
## macroareaSouth          11.45 ***
##                               (0.21)
## areeintB                  1.30
##                               (0.84)
## areeintC                 -1.23 **
##                               (0.43)
## areeintD                 -2.40 ***
##                               (0.44)
## areeintE                 -3.83 ***
##                               (0.44)
## areeintF                 -6.24 ***
##                               (0.52)
## -----
## R^2                         0.65
## Adj. R^2                     0.65
## Num. obs.                   7700
## -----
## *** p < 0.001; ** p < 0.01; * p < 0.05

```

Infine, proviamo anche a metterci dentro il numero di residenti, che è una variabile continua, e la superficie del comune, che è anch'essa una variabile continua.

```

regr3 = lm(data=data, M5S ~ macroarea + areeint + num_residenti + superficie)
screenreg(regr3)

```

```
##
```

```

## =====
##          Model 1
## -----
## (Intercept)      13.44 ***
##                   (0.51)
## macroareaNorth-East -6.51 ***
##                   (0.24)
## macroareaNorth-West -5.47 ***
##                   (0.22)
## macroareaSouth    11.51 ***
##                   (0.21)
## areeintB         2.09 *
##                   (0.85)
## areeintC         -0.16
##                   (0.47)
## areeintD         -1.35 **
##                   (0.48)
## areeintE         -2.81 ***
##                   (0.48)
## areeintF         -5.23 ***
##                   (0.55)
## num_residenti   0.00 ***
##                   (0.00)
## superficie      0.00 **
##                   (0.00)
## -----
## R^2              0.65
## Adj. R^2        0.65
## Num. obs.       7700
## -----
## *** p < 0.001; ** p < 0.01; * p < 0.05

```

Qui termina la nostra lezione. Di nuovo: non vi spaventate. R è difficile. Adesso sapete che esiste. Se un domani vorrete (o dovrete) impararlo il mio consiglio è di essere pazienti e non arrabbiarvi. Dopo un po' ce la si fa e ne vale la pena.

Per qualunque cosa non esitate a scrivermi a g.scaduto2@campus.unimib.it

Ciao!



WHO WOULD WIN?

MULTIBILLION
COMPANIES THAT RULED THE
STATISTICAL PROGRAMMING
MARKET FOR DECADES



SOME WEIRD
LETTER AND A SNAKE???



Using RStudio on Windows



Using RStudio on Mac



PS: questo intero documento è stato scritto su RStudio.