

# Introduzione a R

...rapida e indolore

---

Angela Andreella

13/10/2020

Let's start!

# use...R!

- Avete **installato** R?
- Quanti di voi hanno mai **usato** R o un altro linguaggio di programmazione?

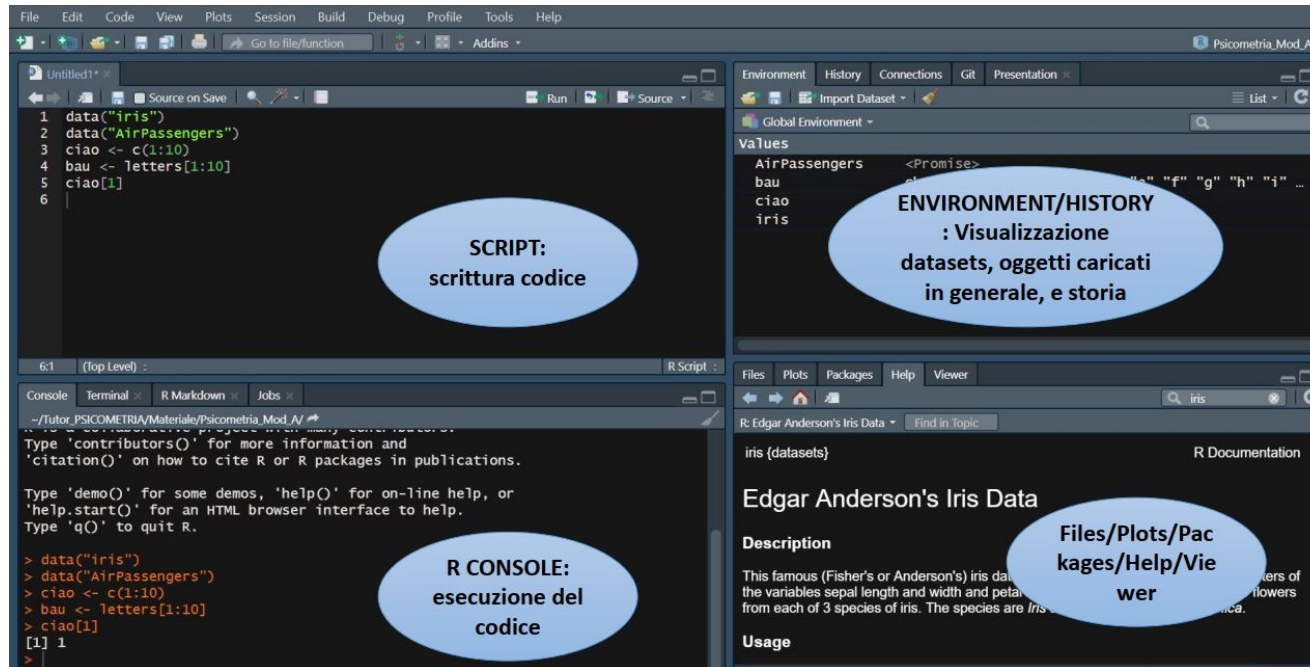
Perchè usare R?

- **Analisi riproducibili**
- **Free, Open-Source**
- In continua **evoluzione** ed espansione
- Large **community**



# use...R!

Prima di tutto.... proviamo R! ma Rstudio è più semplice:



✧ **Attenzione!** R e Rstudio non sono la stessa cosa! R è un linguaggio-ambiente di programmazione, mentre Rstudio è un IDE che rende R più user-friendly

# Pacchetti per tutti!

R fornisce una serie di pacchetti (**packages**) che racchiudono funzioni molto utili e di vario genere. Per esempio con i seguenti comandi:

```
install.packages("xlsx") #Installiamo il pacchetto "xlsx"  
library(xlsx) #Dopo averlo installato lo richiamiamo
```

installiamo il pacchetto `xlsx` e lo richiamiamo nella nostra sessione. Questo pacchetto ci permette di caricare all'interno di R dati da Excel.

→ per ogni cosa che volete fare esiste un pacchetto corrispondente! Se trovate questo errore in futuro:

```
Error in library(dplyr) : there is no package called dplyr
```

significa che dovete installare il pacchetto `dplyr`. 🐼 **Consiglio:** errori/bug cercate su google o [stackoverflow](https://stackoverflow.com)!



Un po' di basi

# Operazioni

Per le operazioni di somma, sottrazione, moltiplicazione e divisione si utilizzano  $+$   $-$   $*$   $/$ . Per l'elevazione a potenza si usa:  $**$  o  $^$ .

```
(4+5)*3
```

Gli operatori logici corrispondono invece ai seguenti simboli:

```
& #AND  
| #OR  
> #maggiore  
< #minore  
== #uguale  
!= #diverso  
>= #maggiore uguale  
<= #minore uguale
```

✳ **Attenzione:** I simboli  $=$  e  $<-$  si usano per assegnare un valore a una variabile

```
pluto = "ciao"
```

diverso da  $==$  (operazione logica che definisce l'uguale)!!

# Operazioni

Vediamo qualche esempio insieme

```
4==10  
5!=10  
(4==4) & (10!=1)  
8>=10  
3<20
```

✦ **Attenzione:**

```
TRUE==1
```

```
FALSE==0
```



# Altri comandi fondamentali

## Aiuto?

```
help("+") #cerchiamo aiuto per capire l'uso di +
```

## Cambiare working directory?

```
setwd("il_mio_path")
```

## Vedere cosa c'è nella mia working directory?

```
ls()
```

## Cancellare quello che abbiamo trovato nella working directory?

```
rm(list = ls()) #Cancelliamo tutti gli elementi presenti  
rm(pluto) #cancelliamo l'oggetto pluto
```

# Caricamento dati

Come caricare i dati? dipende... 🤖

Per prima cosa vedete il **formato** del vostro documento (xlsx, csv, xls, txt etc), in generale, vi sarà un comando corrispondente:

```
read.table("path/file_name.txt") #formato txt  
read.csv("path/file_name.csv") #formato csv  
read.xlsx("path/file_name.xlsx") #formato xlsx
```

etc.. se siete indecisi usate l'help, google, stackoverflow o [R-bloggers](#)! 😊

oppure andate su Files → Import Datasets → ...

Proviamo insieme!

Qualche funzione utile:

```
colnames(db)  
rownames(db)
```

Ma le variabili?

# Tipologie di variabili

Carichiamo il file di esempio:

```
DB <- read.csv("Datasets/HappinessAlcoholConsumption.csv")
```

e vediamo qualche esempio di variabile:

```
str(DB) #con questo comando vediamo la struttura del dataset DB  
head(DB) #Vediamo le prime 6 righe/osservazioni  
View(DB) #Vediamo in finestra separata il dataset intero
```

1. Numeric : HappinessScore
2. Factor (Variabili Qualitative): Country
3. Integer: Beer\_PerCapita

ma abbiamo anche character, logical, NA, NaN.

# Tipologie di oggetti (alcuni)

## Vettori

```
pluto <- c(1:100) #creiamo un vettore con valori da 1 a 100
pluto1 <- seq(1,500,5) #creiamo un vettore con valori da 1 a 500 ogni 5 valori
pluto2 <- rep(10,5) #creiamo un vettore dove si ripete il valore 10 5 volte
pluto[3] #richiamiamo l'elemento del vettore pluto che sta nella 3 posizione
pluto[1:3] #richiamiamo l'elemento del vettore pluto che sta nella 1,2,3 posizioni
is.vector(pluto) #verifichiamo che pluto sia un vettore
length(pluto) #dimensione del vettore pluto?
sort(pluto) #ordiniamo pluto
pluto/5 #dividiamo ogni elemento di pluto per 5
```

proviamo insieme!

# Tipologie di oggetti (alcuni)

## Matrici

```
pippo <- matrix(data = 10, nrow = 10, ncol = 5) #Creiamo una matrice  
#con 10 righe e 5 colonne contenente solo 10  
pippo1 <- matrix(data = pluto, nrow = 20, ncol = 5, byrow = T) #mettiamo pluto in  
#pippo! partendo dalle righe (byrow)  
pippo2 <- cbind(pluto,pluto1) #Uniamo i due vettori per colonna  
pippo3 <- rbind(pluto,pluto1) #Uniamo i due vettori per riga  
dim(pippo) #Dimensione di pippo?  
ncol(pippo) #numero di colonne di pippo?  
nrow(pippo) #numero di righe di pippo?  
is.matrix(pippo) #Check che pippo sia una matrice
```

✳ Se non vi ricordate → `help(matrix)`!

# Tipologie di oggetti (alcuni)

## Data frame

```
is.data.frame(DB)
rick <- data.frame(pippo2[1:20,]) #creiamo un dataframe di nome rick
#che prende le prime 20 osservazioni di pippo2
dim(rick) #Dimensione? nrow, ncol etc come matrix
colnames(rick) #Nomi colonna?
rownames(rick) #Nomi riga?
rownames(rick) <- letters[1:dim(rick)[1]] #rinominiamo le righe di rick
#come le prime 20 lettere dell'alfabeto
rick[10,2] #Richiamiamo l'elemento della 10 riga e 2 colonna
rick$pluto #richiamo la variabile pluto (prima colonna)
```

? Che differenza c'è tra `data.frame` e `matrix`?

Prima di continuare..



1. Domande ?

2. Ma perdo tutto quello che ho caricato/scritto? No ovviamente 😊

```
save(rick, file = "your_path/il_mio_primo_rda.rda") #salviamo l'oggetto  
#rick in il_mio_primo_rda  
load("your_path/il_mio_primo_rda.rda") #e lo ricarico sulla working directory!!
```

Un po' di analisi descrittiva

# Variabili qualitative

## Tabelle di Frequenza

### Frequenze assolute

```
fa_h <- table(DB$Hemisphere)
addmargins(fa_h) #Aggiungi totale alla tabella fa_reg
```

```
##
## both north noth south Sum
##    5    92     4    21  122
```

# Variabili qualitative

## Tabelle di Frequenza

Frequenze relative

```
fa_h/nrow(DB)
```

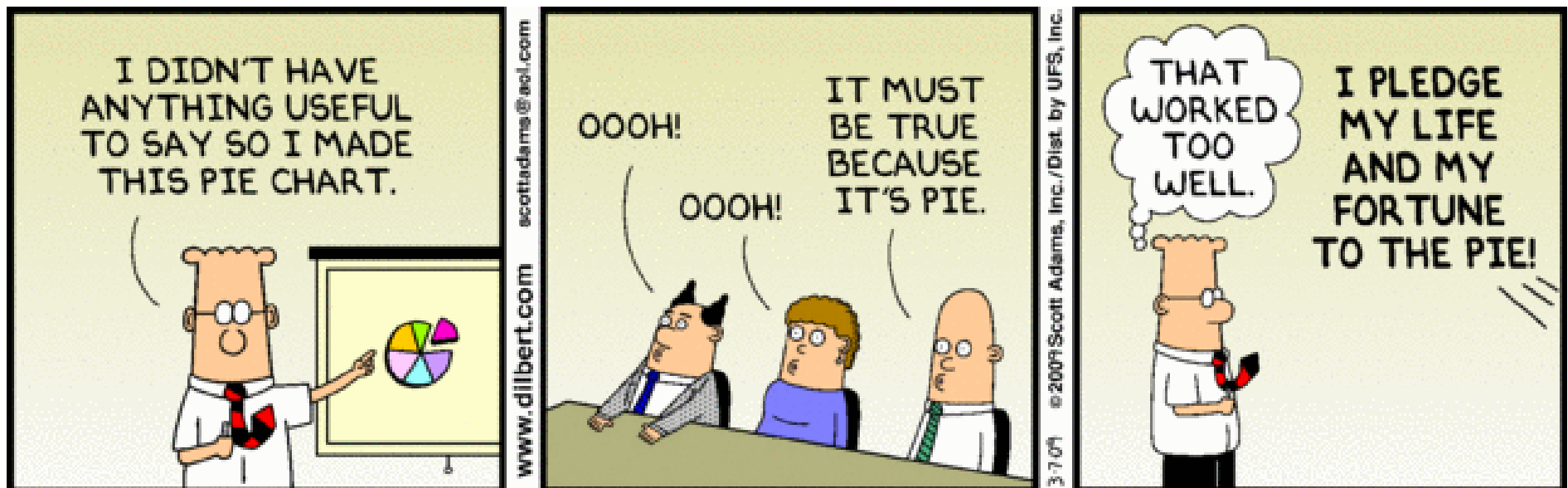
```
##  
##      both      north      noth      south  
## 0.04098361 0.75409836 0.03278689 0.17213115
```

```
fr_h <- prop.table(fa_h)  
addmargins(fr_h) #Aggiungi totale alla tabella fr_reg
```

```
##  
##      both      north      noth      south      Sum  
## 0.04098361 0.75409836 0.03278689 0.17213115 1.00000000
```

# Variabili qualitative

## Grafici

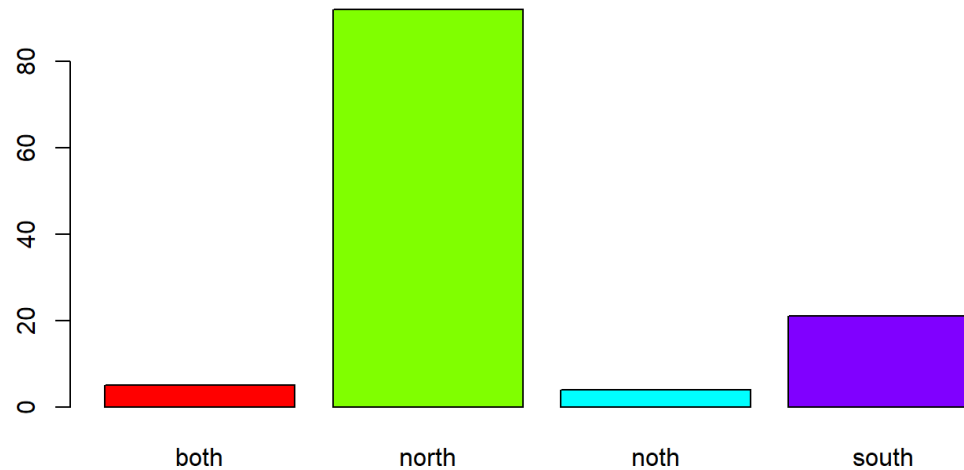


# Variabili qualitative

## Grafici

### Barplot

```
barplot(fa_h, col = rainbow(length(levels(DB$Hemisphere))))
```



# Variabili qualitative

## Grafici

Grafico a torta

```
pie(fa_h, col = rainbow(length(levels(DB$Hemisphere))))
```



# Variabili quantitative

## Tabelle di Frequenza e altro

Per le tabelle di frequenza assolute e relative valgono gli stessi comandi usati per la variabile qualitativa Hemisphere.

Con le variabili quantitative abbiamo anche:

```
min(DB$HappinessScore)
```

```
## [1] 3.069
```

```
max(DB$HappinessScore)
```

```
## [1] 7.526
```



# Variabili quantitative

## Tabelle di Frequenza e altro

```
summary(DB$HappinessScore)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  3.069   4.528   5.542   5.525   6.477   7.526
```

```
mean(DB$HappinessScore)
```

```
## [1] 5.524828
```

```
median(DB$HappinessScore)
```

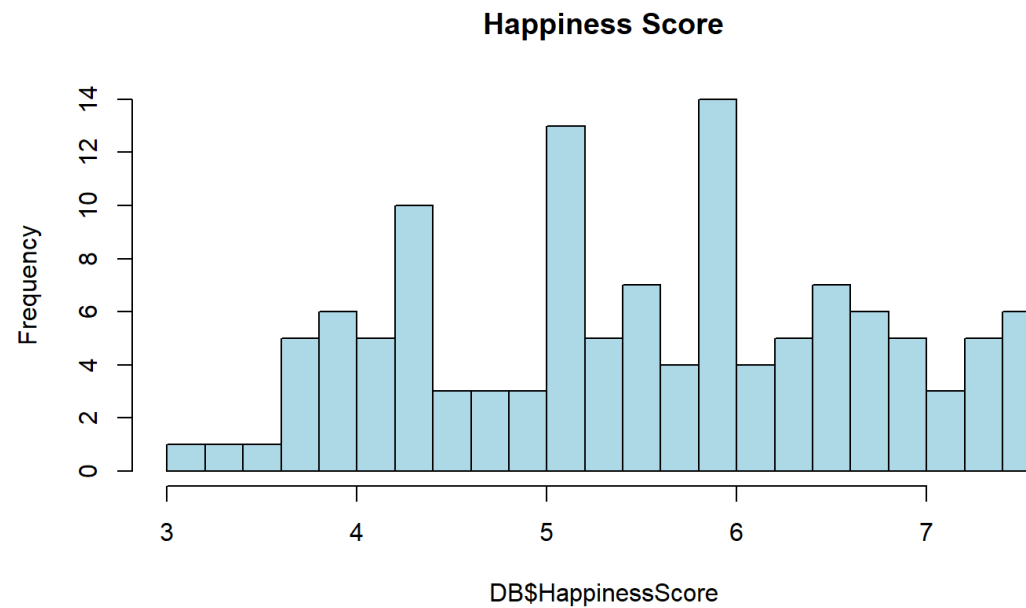
```
## [1] 5.542
```

# Variabili quantitative

## Grafici

### Istogramma

```
hist(DB$HappinessScore, breaks = 20, col = "lightblue", main = "Happiness Score") #breaks decide gli intervalli!
```

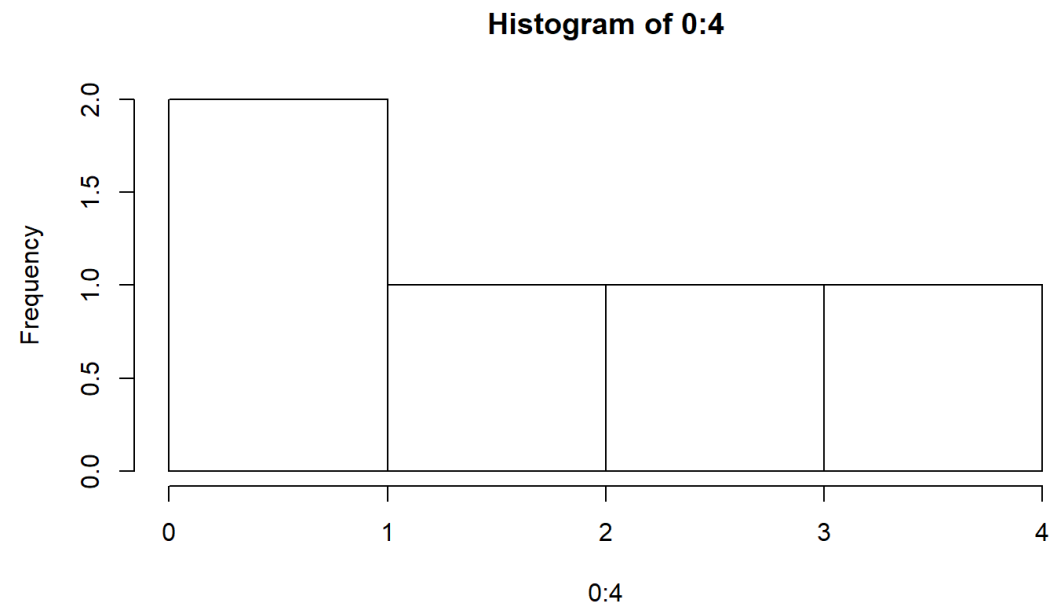


# Variabili quantitative

## Grafici

✦ **Attenzione** agli intervalli!

```
hist_plot <-hist(0:4)
```



```
hist_plot
```

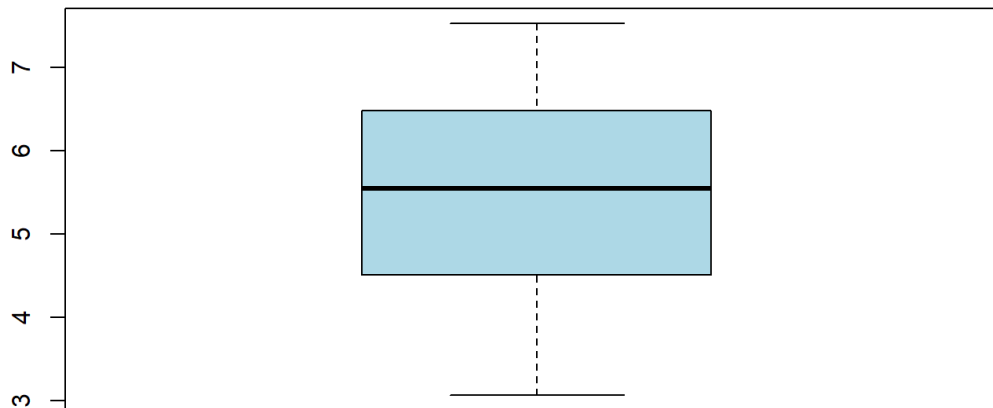
```
## $breaks  
## [1] 0 1 2 3 4  
##  
## $counts
```

# Variabili quantitative

## Grafici




### Boxplot

```
boxplot(DB$HappinessScore,col = "lightblue")
```



Vuoi diventare un supeR useR?

# Cosa fare?

- Allenarsi 
- Risolvere i bug da soli 
- Documentarsi :
  - Google, [R-bloggers](#), ...
  - [R Cookbook](#)
  - [R in a Nutshell](#)
  - [Learning R](#)
  - [An Introduction to Statistical Learning: with Applications in R](#)

Trovate tutto il materiale di questa lezione [qui!](#)