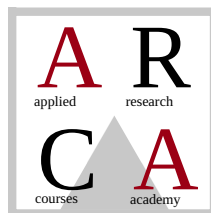


Introduzione a R

Giornata 1



Corsi ARCA - @DPSS

Filippo Gambarota

Primi passi con R

Installazione

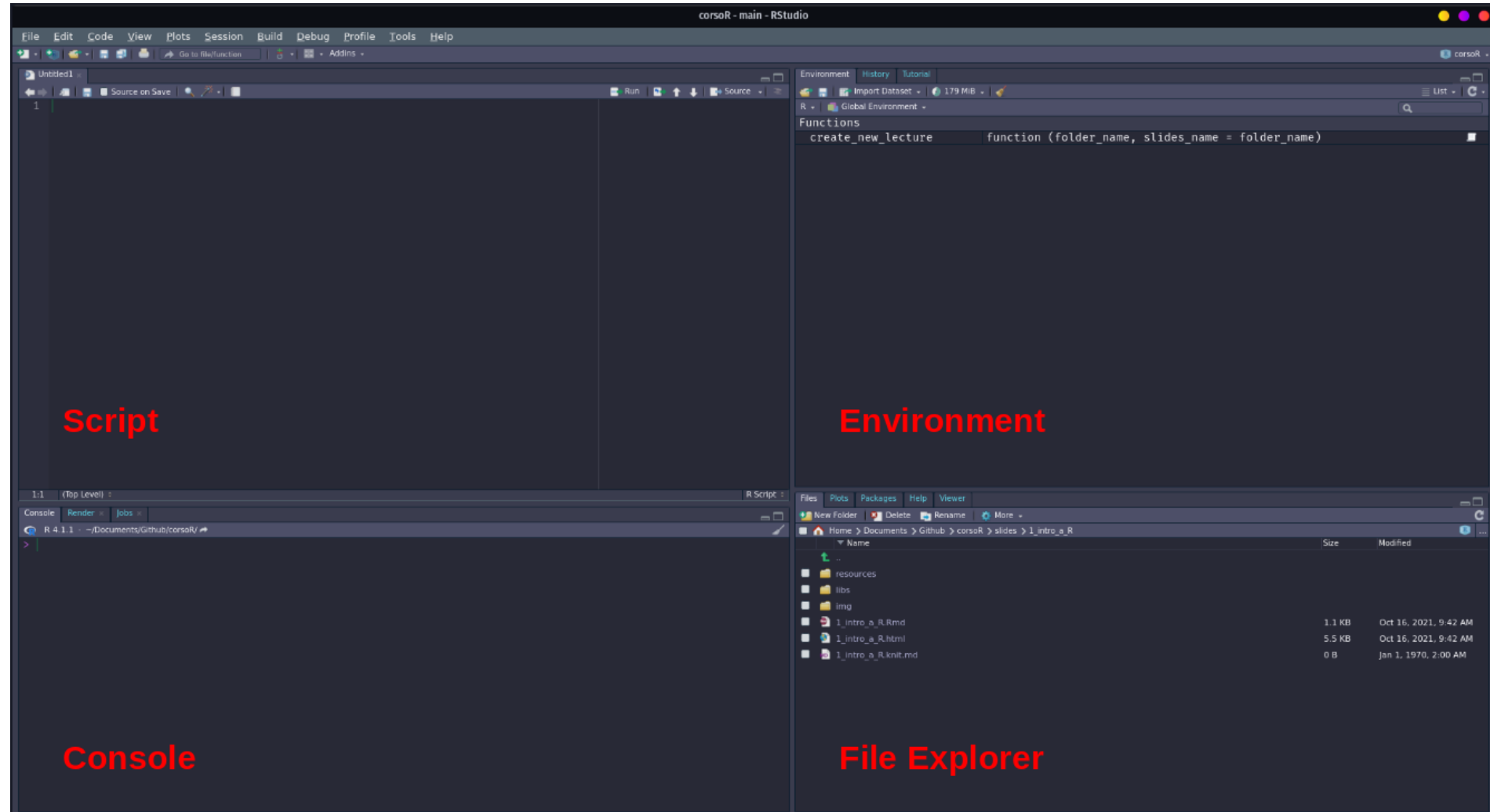
Per l'installazione trovate le indicazioni nella sezione [Installare R e RStudio](#) del libro. In generale i passaggi sono:

- scaricare R e installare **R** per il vostro sistema operativo
- scaricare e installare **RStudio**

Come si presenta R

Console

RStudio



I primi passi in R

R come calcolatrice

In R è possibile effettuare tutte le **operazioni matematiche** e algebriche dalle più semplici alle più avanzate

Funzione	Nome	Esempio
<code>x + y</code>	Addizione	<code>> 5 + 3</code> <code>[1] 8</code>
<code>x - y</code>	Sottrazione	<code>> 7 - 2</code> <code>[1] 5</code>
<code>x * y</code>	Moltiplicazione	<code>> 4 * 3</code> <code>[1] 12</code>
<code>x / y</code>	Divisione	<code>> 8 / 3</code> <code>[1] 2.666667</code>
<code>x %% y</code>	Resto della divisione	<code>> 7 %% 5</code> <code>[1] 2</code>

Operatori matematici

- Importante considerare l'**ordine delle operazioni** analogo alle regole della matematica: $2 \times 3 + 1$ prima 2×3 e poi $+ 1$. Analogamente in R:

```
# Senza parentesi
```

```
2 * 3 + 1
```

```
## [1] 7
```

```
# Con le parentesi
```

```
(2 * 3) + 1
```

```
## [1] 7
```

```
# Con le parentesi forzando un ordine diverso
```

```
2 * (3 + 1)
```

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

Operatori relazionali

Gli operatori relazionali sono molto utili dentro le **funzioni**, per **selezionare elementi dalle strutture dati** (vedremo più avanti) e in generale per **controllare** alcune sezioni del nostro codice:

```
3 > 4
```

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

```
3 >= 3
```

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

```
10 < 100
```

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

```
40 == 40
```

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

```
10 != 50
```

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

Operatori logici

Gli operatori logici permettono di **combinare espressioni relazionali** e ottenere sempre un valore `TRUE` o `FALSE`:

```
3 > 4 & 10 < 100
```

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

```
10 < 100 | 50 > 2
```

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

```
!5 > 4
```

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

R e gli oggetti

R e gli oggetti

| “Everything that exists in R is an object” - John Chambers

Il concetto di **oggetto** è fondamentale in R. Essenzialmente tutto quello che possiamo creare o utilizzare in R come un numero, un vettore, dei caratteri o delle funzioni sono creati come oggetti.

R e le funzioni

“Everything that happen in R is a function call” - John Chambers

Anche il concetto di **funzione** è fondamentale in R. Essenzialmente tutto quello che facciamo è chiamare **funzioni** su oggetti ottenendo un nuovo oggetto o modificando un oggetto esistente

Cosa possiamo usare/creare in R?

- **Numeri**: 100, 20, 6, 5.6 sono tutti numeri interpretati e trattati come tali
- **Stringhe**: "ciao", "1" sono *caratteri* che vengono interpretati letteralmente devono essere dichiarati con `"`
- **Nomi**: ciao, x sono nomi (senza virgolette) e sono utilizzati per essere associati ad un oggetto (variabile, funzione, etc.)
 - **operatori**: sono delle funzioni (e quindi oggetti con un nome associato) che si utilizzano in modo particolare. `3 + 4` in questo caso `+` è un operatore (funzione) che si può usare anche come `+(3, 4)`

R e gli oggetti

- Come creare un oggetto?
- Oggetti e nomi
- Dove viene creato l'oggetto?

Come creare un oggetto?

La creazione di un oggetto avviene tramite il comando `<-` oppure `=` in questo modo: `nome`
`<- oggetto`:

```
x
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

```
10 # questo non è un oggetto, non è salvato
```

```
## [1] 10
```

```
x <- 10 # ora il valore numerico 10 è associato al nome "x"
```

```
x
```

```
## [1] 10
```

Convenzioni vs regole

Ci sono alcune cose da considerare quando si scrive codice ed in particolare si creano oggetti:

- **alcune modalità sono errate** --> R ci fornisce un messaggio di errore
- **alcune modalità sono sconsigliate** --> funziona tutto ma ci potrebbero essere problemi
- **alcune modalità sono stilisticamente errate** --> funziona tutto, nessun problema ma... anche l'occhio vuole la sua parte

Oggetti e nomi

Il nome di un oggetto è importante sia per l'utente che per il software stesso:

```
1 <- 10 # errore  
  
_ciao <- 10 # errore  
  
mean <- 10 # possibile ma pericoloso  
  
`1` <- 10 # con i backticks si può usare qualsiasi nome ma poco pratico
```

```
## Error: <text>:4:2: unexpected symbol  
## 3:  
## 4: _ciao  
##      ^
```

```
my_obj <- 10  
  
my.obj <- 10  
  
My_obj <- 10 # attenzione a maiuscole e minuscole
```

Oggetti e nomi (proibiti)

In R ci sono anche dei nomi non solo sconsigliati ma proprio **proibiti** che nonostante siano sintatticamente corretti, non possono essere usati (per ovvie ragioni):

```
mean <- 10 # ok ma sconsigliato
```

```
function <- 10
```

```
## Error: <text>:4:10: unexpected assignment
```

```
## 3:
```

```
## 4: function <-
```

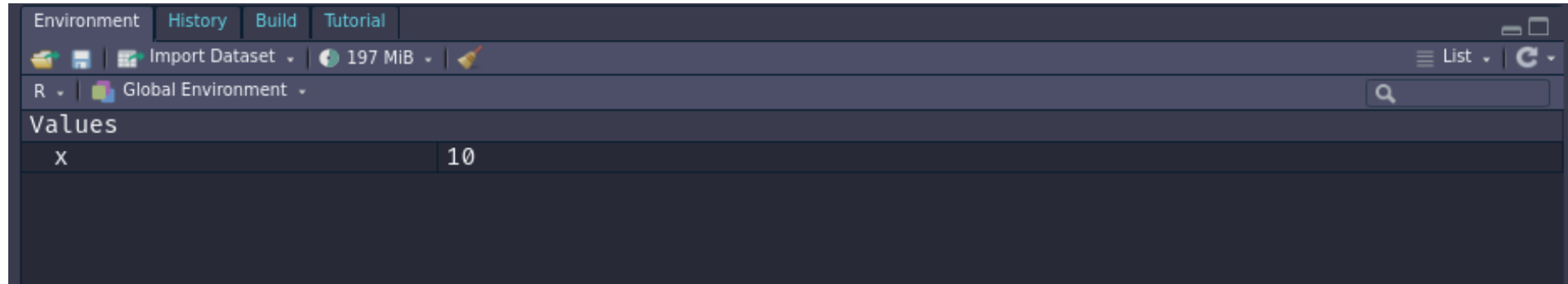
```
##           ^
```

```
TRUE <- 4
```

```
## Error in TRUE <- 4: invalid (do_set) left-hand side to assignment
```

```
T <- 2 # attenzione
```

Dove viene creato l'oggetto?



Non solo numeri (anticipazione)

Non solo numeri (anticipazione)

In R possiamo usare oltre ai numeri (in senso matematico) anche le **stringhe** ovvero parole, lettere interpretate così come sono:

```
"ciao" # stringa formata da 5 caratteri
```

```
## [1] "ciao"
```

```
x <- "ciao" # associa la stringa ad un oggetto
```

```
x + 1 # operazioni matematiche con stringhe (ha senso?)
```

```
## Error in x + 1: non-numeric argument to binary operator
```

```
x == "ciao"
```

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

```
x > 10
```

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

Formula Syntax (extra, but useful)

- In R vedrete spesso l'utilizzo dell'operatore `~` per fare grafici, statistiche descrittive, modelli lineari etc. L'utilizzo di `y ~ x` permette di creare del codice R che non viene eseguito subito ma può essere eseguito successivamente in un ambiente specifico.
- è l'unico caso dove nomi non assegnati possono essere utilizzati senza errori
- questo tipo di programmazione si chiama **non-standard evaluation** perchè appunto non funziona come il solito codice R

```
y # y non esiste e quindi ho un errore
```

```
## Error in eval(expr, envir, enclos): object 'y' not found
```

```
y ~ x # usando ~ non ho errori perchè il codice non viene eseguito
```

```
## y ~ x  
## <environment: 0x55cd6bc86b80>
```


Formula Syntax (extra, but useful)

Le formule vengono utilizzate in tantissimi contesti.

Per fare modelli di regressione

```
# un modello lineare -> dipendente ~ indipendenti  
lm(y ~ x1 + x2)
```

Per fare aggregare un dataset

```
# per aggregare un dataset (vedremo più avanti :) )  
aggregate(y ~ x, data = data, FUN = mean)
```

Per fare grafici

```
# per fare grafici  
boxplot(y ~ x, data = data)
```

Formula Syntax (extra, but useful)

In generale, ogni volta che usate delle variabili *unquoted* (senza virgolette) e queste non sono dichiarate nell'ambiente, state probabilmente usando la **non-standard evaluation** e c'è una formula da qualche parte 😊

Ambiente di lavoro

Ambiente di lavoro

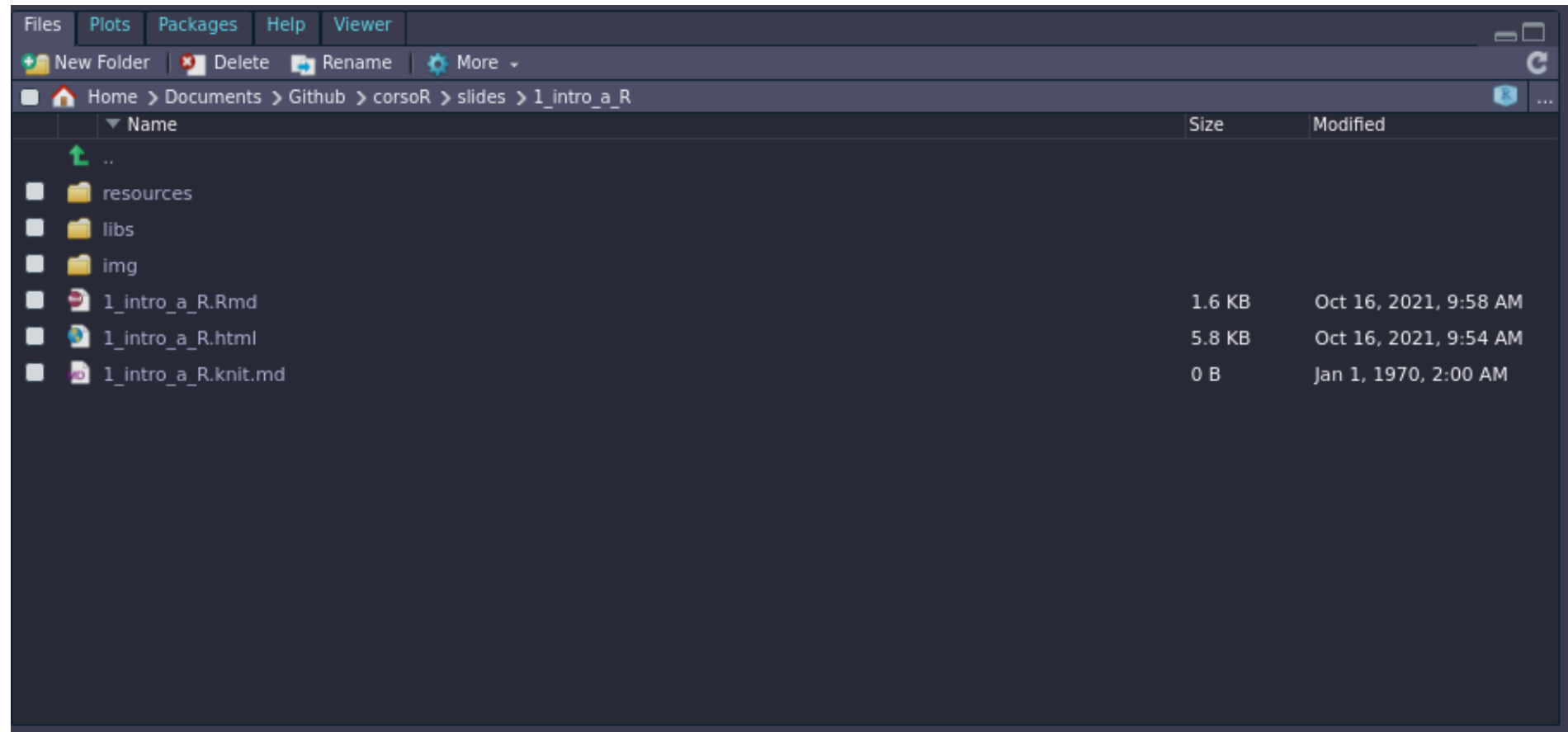
- **Environment**
- **Working directory**
- **Packages**

Environment

Il **working environment** è la vostra *scrivania* quando lavorate in R. Contiene tutti gli oggetti (variabili) creati durante la sessione di lavoro.

Working Directory

La working directory è la posizione (cartella) sul vostro PC dove R sta lavorando e nella quale R si aspetta di trovare i vostri file, se non specificato altrimenti



Packages

In R è possibile installare e caricare pacchetti aggiuntivi che non fanno altro che rendere disponibili librerie di funzioni create da altri utenti. Per utilizzare un pacchetto:

- Installare il pacchetto con `install.packages("nomepacchetto")`
- Caricare il pacchetto con `library(nomepacchetto)`

Packages



The screenshot shows the RStudio interface with the 'Packages' tab selected. The 'User Library' section is expanded, displaying a list of installed packages. Each row includes a checkbox, the package name, a description, the version number, and two circular status icons (one grey, one white).

	Name	Description	Version		
User Library					
<input type="checkbox"/>	abind	Combine Multidimensional Arrays	1.4-5		
<input type="checkbox"/>	anytime	Anything to 'POSIXct' or 'Date' Converter	0.3.9		
<input type="checkbox"/>	arrayhelpers	Convenience Functions for Arrays	1.1-0		
<input type="checkbox"/>	AsioHeaders	'Asio' C++ Header Files	1.16.1-1		
<input type="checkbox"/>	askpass	Safe Password Entry for R, Git, and SSH	1.1		
<input type="checkbox"/>	assertthat	Easy Pre and Post Assertions	0.2.1		
<input type="checkbox"/>	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1		
<input type="checkbox"/>	base64enc	Tools for base64 encoding	0.1-3		
<input type="checkbox"/>	bayesplot	Plotting for Bayesian Models	1.8.1		
<input type="checkbox"/>	BayesRS	Bayes Factors for Hierarchical Linear Models with Continuous Predictors	0.1.3		
<input type="checkbox"/>	bayestestR	Understand and Describe Bayesian Models and Posterior Distributions	0.11.0		
<input type="checkbox"/>	BH	Boost C++ Header Files	1.75.0-0		
<input type="checkbox"/>	binom	Binomial Confidence Intervals For Several Parameterizations	1.1-1		
<input type="checkbox"/>	bitops	Bitwise Operations	1.0-7		
<input type="checkbox"/>	blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.2		
<input type="checkbox"/>	bookdown	Authoring Books and Technical Documents with R Markdown	0.24		
<input type="checkbox"/>	bridgesampling	Bridge Sampling for Marginal Likelihoods and Bayes Factors	1.1-2		
<input type="checkbox"/>	brio	Basic R Input Output	1.1.2		
<input type="checkbox"/>	brms	Bayesian Regression Models using 'Stan'	2.16.1		
<input type="checkbox"/>	Brobdingnag	Very Large Numbers in R	1.2-6		
<input type="checkbox"/>	broom	Convert Statistical Objects into Tidy Tibbles	0.7.9		

Come lavorare in R

Scrivere e organizzare script

- Lo script è un file di testo dove il codice viene salvato e può essere lanciato in successione
- Nello script è possibile combinare **codice** e **commenti**

R Projects

Gli `R projects` sono una feature implementata in R Studio per organizzare una cartella di lavoro

- permettono di impostare la **working directory** in automatico
- permettono di usare **relative path** invece che **absolute path**
- rendono più **riproducibile** e **trasportabile** il progetto
- permettono un **veloce accesso** ad un determinato progetto

**Come risolvere i problemi nella
~~vita~~ in R**

Come risolvere i problemi ~~nella vita~~ in R

In R gli errori sono:

- inevitabili
- parte del codice stesso
- educativi

Resta solo da capire come affrontarli

R ed errori

Ci sono diversi livelli di **allerta** quando scriviamo codice:

- **messaggi**: la funzione ci restituisce qualcosa che è utile sapere, ma tutto liscio
- **warnings**: la funzione ci informa di qualcosa di *potenzialmente* problematico, ma (circa) tutto liscio
- **error**: la funzione non solo ci informa di un **errore** ma le operazioni richieste non sono state eseguite

Come risolvere un errore?

- capire il messaggio
- leggere la documentazione della funzione
- cercare il messaggio su Google
- chiedere aiuto nei forum dedicati

Come risolvere un errore?

- Ogni funzione ha una pagina di documentazione accessibile con `?nomefunzione` o `??nomefunzione`
- Possiamo cercare anche la documentazione del pacchetto
- Possiamo cercare su Google il nome della funzione o l'eventuale messaggio che riceviamo

Come NON fare una domanda

Hi I'm trying to plot my data using ggplot2 but I'm getting this error I've never seen before... I was searching online and couldn't find anything useful.

This is my code:

```
##PACKAGES
library(ggplot2)
library(viridis)
library(hrbrthemes)

##IMPORT DATA
library(readxl)
TOC_LANG <- read_excel("PHD/LAB/LAB
RESULTS/Jul-2021- TOC/TOC- LAN and GROVE
AV/vario016_TOC_MOD.xlsx",
sheet = "R", range = "B3:E55")

#IMPORT DATA
library(readxl)
TOC_PORTGA <- read_excel("PHD/LAB/LAB
RESULTS/Jul-2021- TOC/TOC- LAN and GROVE
AV/vario016_TOC_MOD.xlsx",
sheet = "R", range = "A59:E105")
View(TOC_LANG)
```


Hi I'm trying to plot my data using ggplot2 but I'm getting this error I've never seen before... I was searching online and couldn't find anything useful.

This is my code:

```
##PACKAGES
library(ggplot2)
library(viridis)
library(hrbrthemes)

##IMPORT DATA
library(readxl)
TOC_LANG <- read_excel("PHD/LAB/LAB
RESULTS/Jul-2021- TOC/TOC- LAN and GROVE
AV/vario016_TOC_MOD.xlsx",
sheet = "R", range = "B3:E55")

#IMPORT DATA
library(readxl)
TOC_PORTGA <- read_excel("PHD/LAB/LAB
RESULTS/Jul-2021- TOC/TOC- LAN and GROVE
AV/vario016_TOC_MOD.xlsx",
sheet = "R", range = "A59:E105")
View(TOC_LANG)
```



Come NON fare una domanda

- Non riporta il **messaggio di errore**!
- Usa un **percorso assoluto**! Come posso riprodurre il suo problema?