CLASS 1

Introduction to RStudio

By Anita Kurm

ABOUT ME

- My name is Anita Kurm
- Have been through BSc Cognitive Science - I know the struggle
- Just started MSc in Cognitive Science at AU
- Got a puppy this summer!
- I come from Estonia, but my native language is Russian
- Email: 201608652@post.au.dk

PLEASE ASK

- **It's fine to interrupt!** Your questions and comments are always valid! If you are in doubt, someone else is probably too.

- Our final goal is your understanding – not me getting through my slides. I might be talking too fast, just ask me to slow down

- State your name when you speak :)

ASK MORE AND GIVE FEEDBACK

- Before, during and after the class
- In the break
- In person and via email: 201608652@post.au.dk
- In fb group ‘CogOverflow' – highly recommended :)

WHY PROGRAMMING

- In general:
  - Widely applicable, makes lots of things either possible or easier
  - Develops structured and creative thinking
- For Cognitive Science:
  - Enables us design and run experiments
  - Enables us analyse observations from experiments, even when dealing with ‘big data’
  - Allows us build computational models

ADVANTAGES OVER OTHER STATISTICAL TOOLS

- There are many programs, where you can just drag and drop files with data, press buttons and run pre-made functions
- Coding has way more to offer
  - You get a powerful and flexible toolbox for statistics, experiments and much more
  - You also get a deeper understanding of analysis, as you create it from the scratch and can always change some parts of it

GUIDELINES

- **Google!**
  - Even very experienced programmers constantly look up how to do one or another thing – no-one knows everything!
  - Online communities like StackOverflow are going to be huge help!!!
    - There is always someone with the same problem or question - and solution is most likely there too!
- **Copy-paste but understand**
  - Use code you found online
  - Reuse your own code
  - Share your code or code you found

STUDENTS HELPING STUDENTS

- You are encouraged to help each other!
- Ask each other questions, try to help when you know the answer
- Improves your skill and understanding
- A good teamwork practice
- Makes our class run smoother

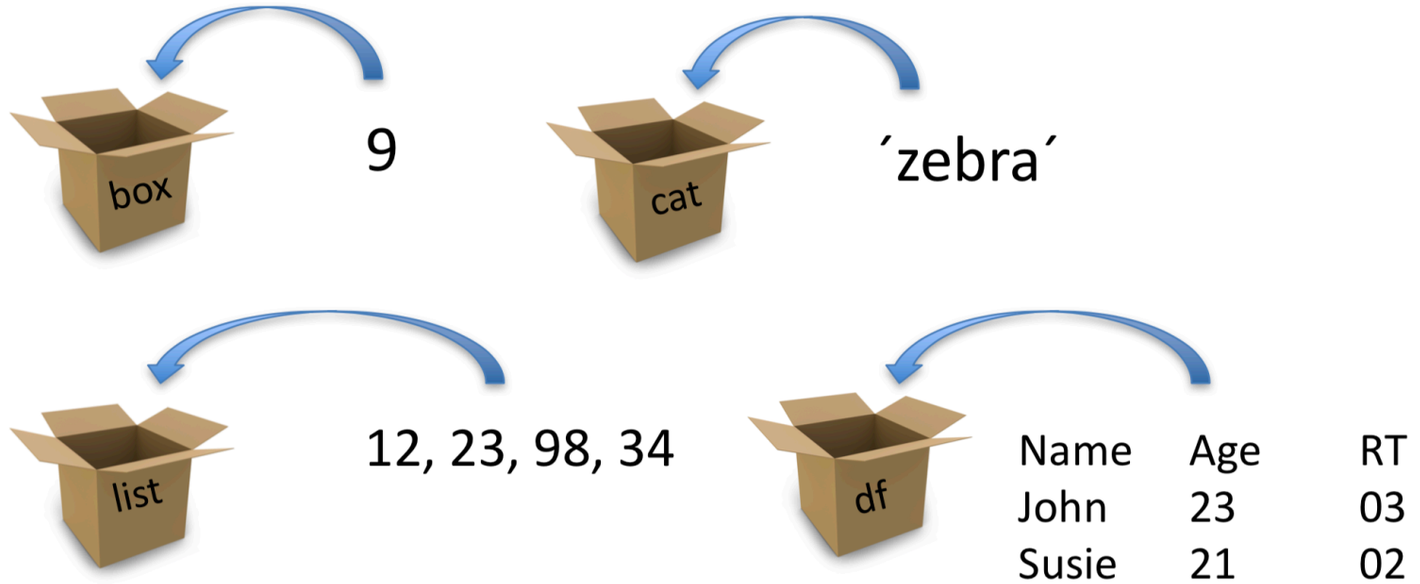
R AND RSTUDIO

- R: Programming language for statistical programming
  - Free and popular – well-supported online
  - Open-source – so many packages, so many opportunities!
  - Powerful and flexible
- RStudio: an intuitive interface for R

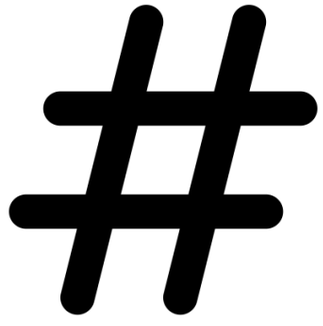
LIVE CODING

Identifier ⟵ Value(s)

variable



NICE CODE



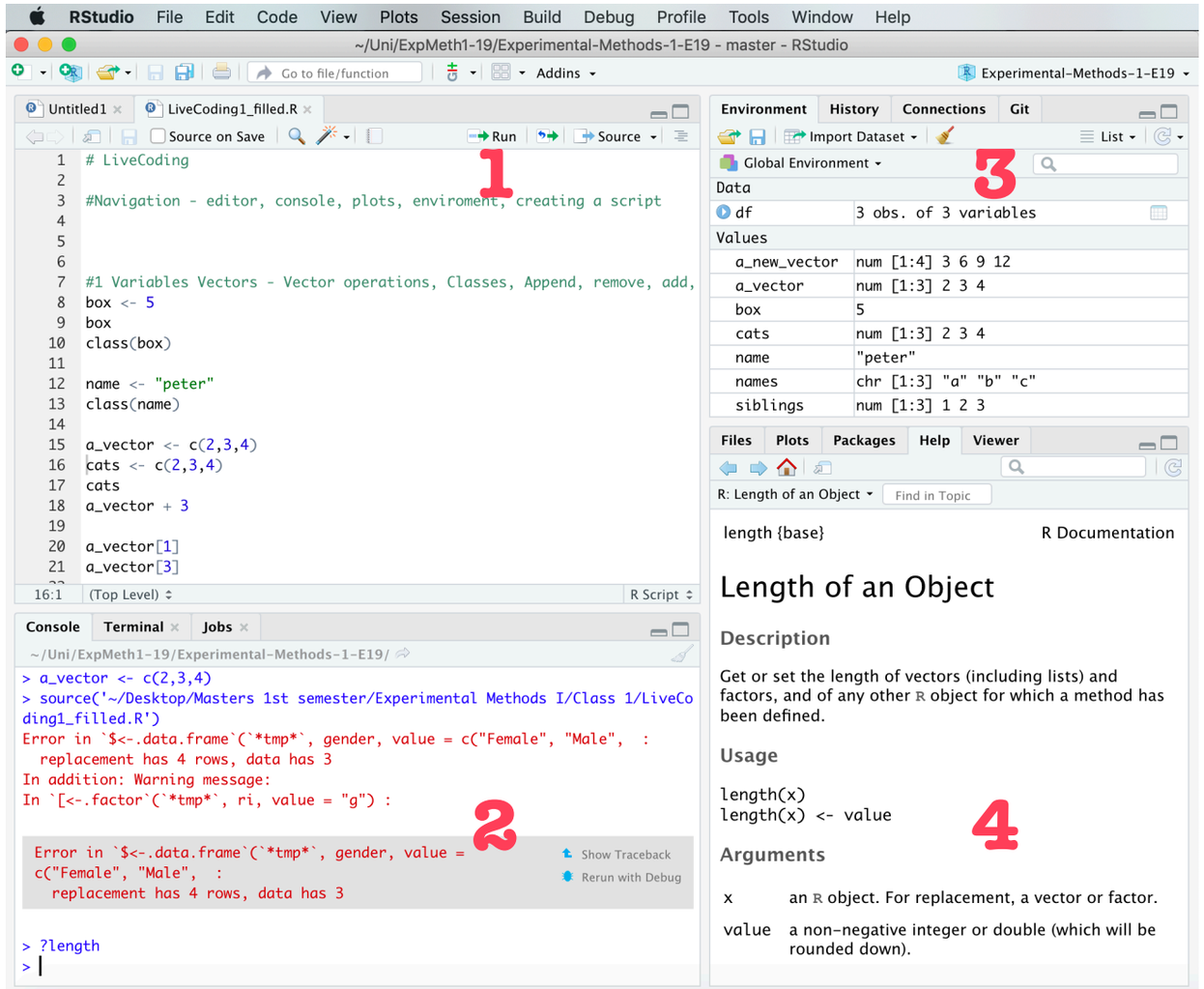
- Use # to comment your code
  - Makes your code more understandable to others and most importantly to YOU!
  - A very convenient way to keep notes about your code
- Space your code
  - R does not care, but you should!
  - Easier to read and spot errors
- Name your variables with meaningful names
- Be consistent

Power tip: Use shortcuts! For fast '<– ' try:

alt + '-' (windows)

⌥ + '-' (mac)

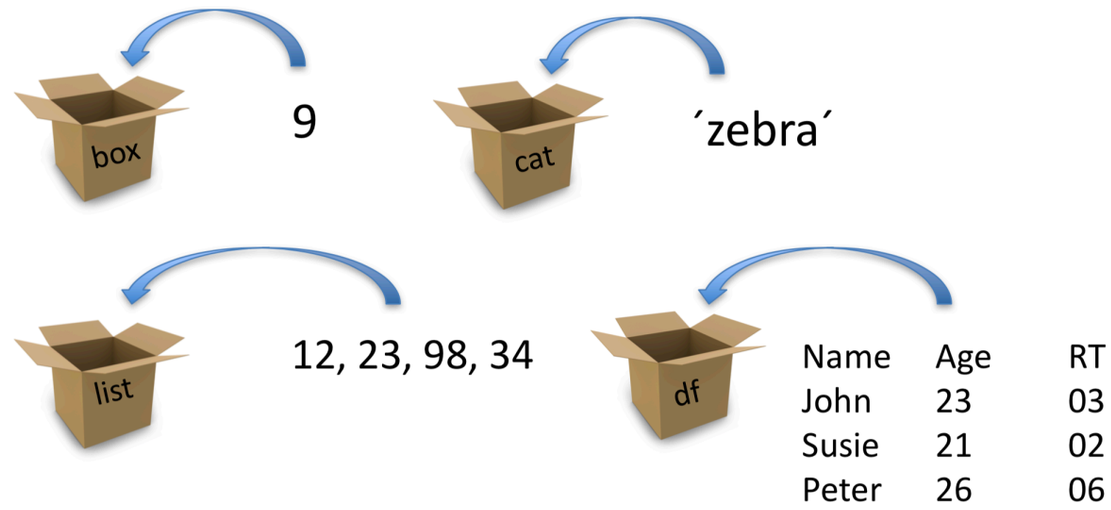
NOTES FOR LIVE CODING: RSTUDIO INTERFACE



- 1. Editor - a 'notepad', where you write down and edit your code and notes
- 2. Console - where your code is executed and outputs are shown
- 3. Tabs for Environment, History, Git…
- 4. Tabs for Graphs, Help…

NOTES FOR LIVE CODING: VARIABLES

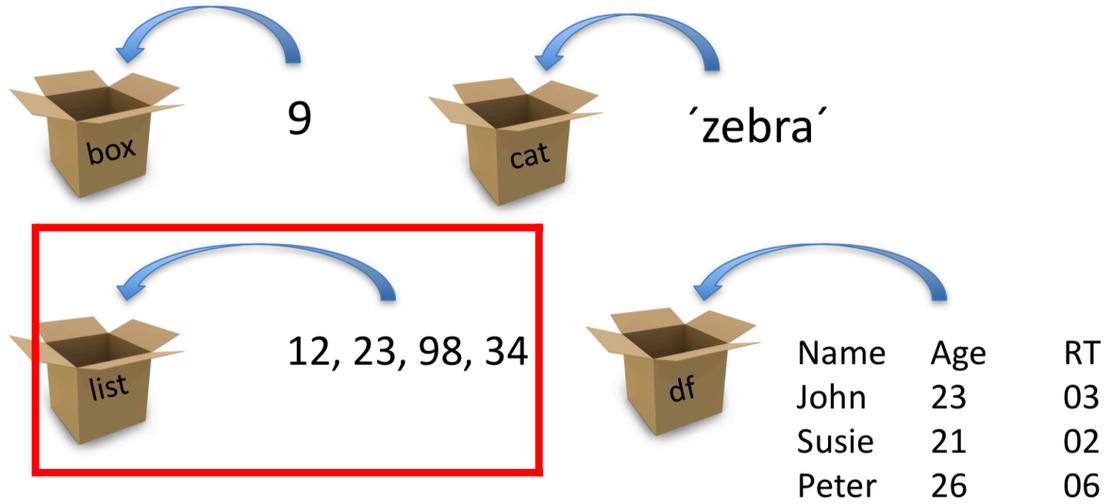
- Variables – objects that store data, whose value can be changed according to your needs
- Unique name you give to a variable (or any other object) – identifier
- Example of a variable: `box <- 9`
- All numbers and characters (letters, words, text) by themselves are constants
- R needs to see quotation marks " " to understand that letters and words are characters, and not commands
- The type of constants inside of your object defines its `class()` and `typeof()`



- Credit and more info: https://www.datamentor.io/r-programming/variable-constant/

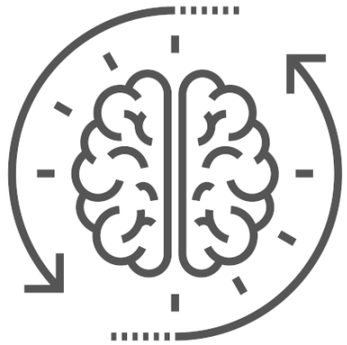
NOTES FOR LIVE CODING: VECTORS

- Vector is a basic data structure in R, which contains several elements
- Vectors can be created using the `c()` function
- The number of elements in the vector can be checked using `length()`
- The order of elements in the vector can be used to access separate elements:
  - e.g. to see only the second element of the vector named 'a_vector' we write `a_vector[2]`



- Credit and more info: https://www.datamentor.io/r-programming/vector/

EXERCISE 1



1. Create a vector of the names of people sitting near you (at least 5), make it a variable by giving the vector a name

2. Similarly make another vector with a guess of how many siblings they each have, name this vector too

3. Add 2 to both vectors what happens and why?

4. Check the class of both vectors, what does it tell you?

5. How many siblings do people have in total?

6. What is the product of person 1 and 2's number of siblings?

Commands you will need:

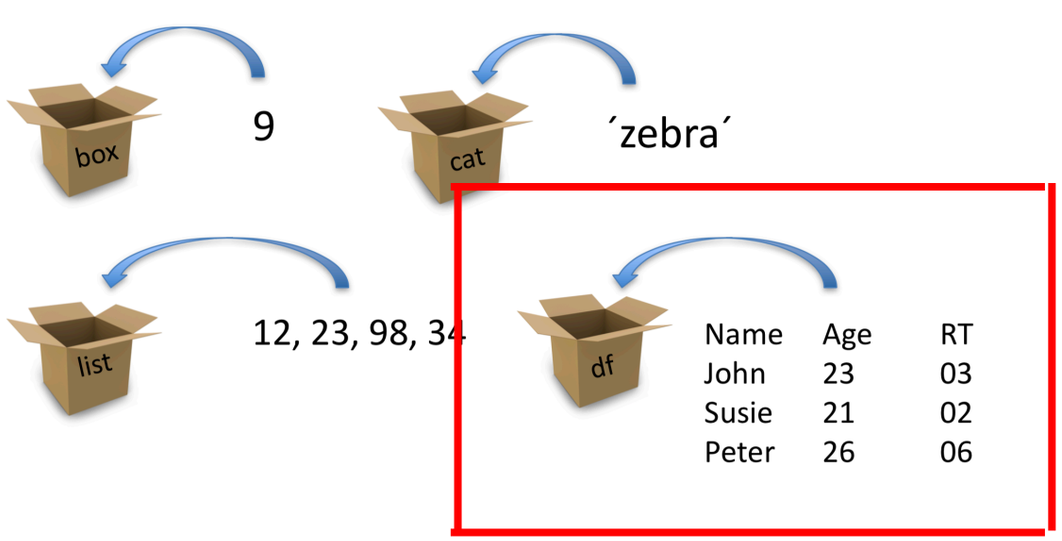
```
c()
sum()
class()
```

Extra:

- *Why does the following code result in an error?* `name <- Peter`
- *Add one new element to the vector with the number of siblings, the new element has to be a word.*
- *What happens if you try to multiply this vector by 2 now? Why?*
- *Remove the new word from the vector and try again! *hint: check the class()*

NOTES FOR LIVE CODING: DATA FRAMES

- Data frame is a two dimensional data structure in R
- Can be created using `data.frame()`
- We can use `[row index,column index]` to access certain values in the df, e.g. value in the 5th row and 2nd column in a data frame named df: `df[5,2]`
- A whole row can be accessed by leaving column index empty (and vice versa): the whole 5th row can be accessed by `df[5,]`
- A column can be accessed by leaving row index empty. Also by using the name of the column.
- To add a new vector as a new row into the data frame, it has to be the same length as other rows in the df, and its elements need to be of the same kind as values in other rows



- Credit and more info: https://www.datamentor.io/r-programming/data-frame/

NOTES FOR LIVE CODING: DATA FRAMES

- More about accessing columns:

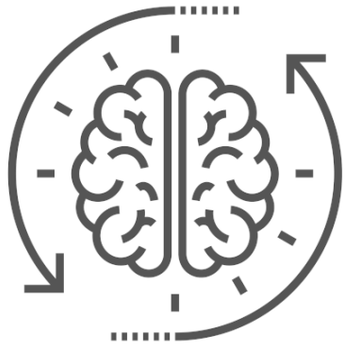
We can use either `[`, `[[` or `$` operator to access columns of data frame.

```
> x["Name"]
Name
1 John
2 Dora
> x$Name
[1] "John" "Dora"
> x[["Name"]]
[1] "John" "Dora"
> x[[3]]
[1] "John" "Dora"
```

Accessing with `[[` or `$` is similar. However, it differs for `[` in that, indexing with `[` will return us a data frame but the other two will reduce it into a vector.

- Credit and more info: https://www.datamentor.io/r-programming/data-frame/

EXERCISE 2



1. Create a dataframe with the previous vectors
2. Add gender to the dataframe
3. Add a new person to the dataframe

**hint: make sure names in the dataframe are as.character()*

4. What is the mean number of siblings?

**hint: make sure the number of siblings in the dataframe is as.numeric()*

5. Ask people how many siblings they have and put the actual numbers as a separate column in your data frame

6. Make a column with numbers showing how much you were 'off'

7. Comment your code

New functions and operators:

`data.frame()`

`$`

`rbind()`

`mean()`

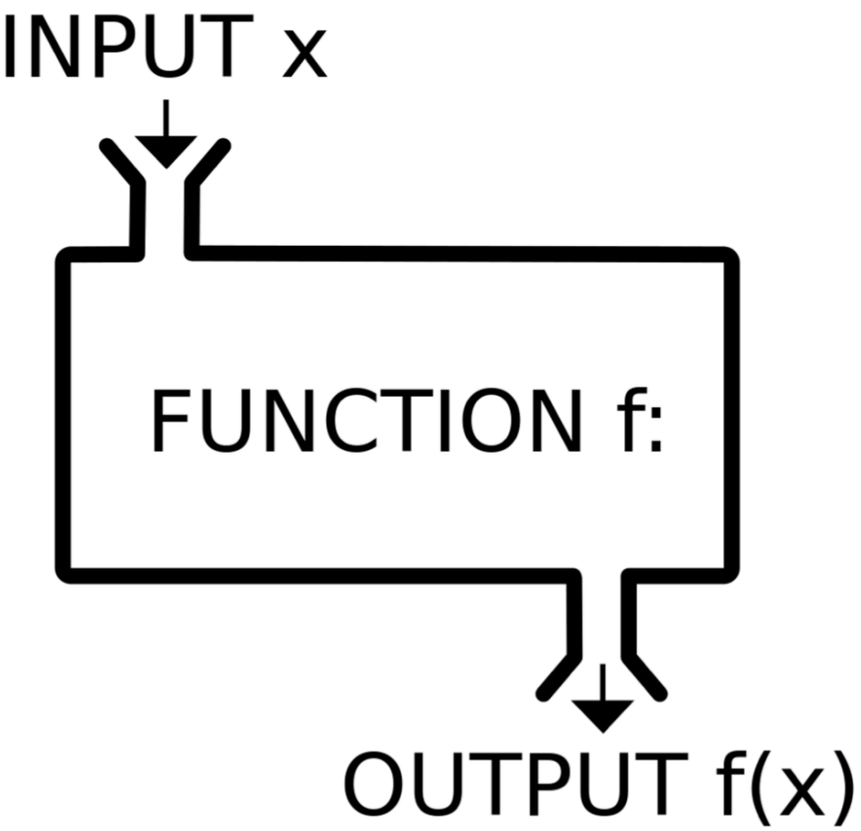
`as.factor()`

`as.character()`

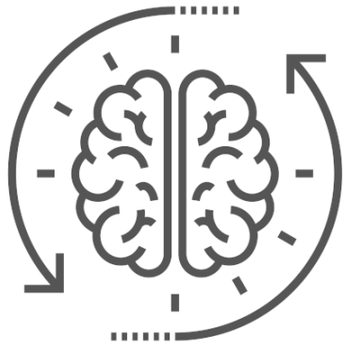
`as.numeric()`

FUNCTIONS

- Almost everything in R is done through functions
- Built-in functions coming with R and in every single package you install
- Users can define their own functions (more advanced stuff)



EXERCISE 3



1. Who has 3 siblings?

2. How many people have more than 2 siblings?

3. Create a subset of the data containing only people you guessed right

4. Use '?' to find what these functions do: round(); length(); unique(); mean()

Extra: Try to use these functions

Functions:

```
install.packages()
library()
subset()
round()
length()
unique()
mean()
```

ERROR AND WARNING MESSAGES

What is the mean of c(3, 4, 'Cat')? What is the warning about? * *hint: use class()*

```
> mean(c(3,4,'cat'))
[1] NA
Warning message:
In mean.default(c(3, 4, "cat")) :
  argument is not numeric or logical: returning NA
```

What is the problem in the following code:

```
105  names <- c("Peter", "Natalie", "Maya")
106  n_pets <- c(1,3,8)
107  pet_frame <- data.frame(names=names n_pets=n_pets)
```

Error messages can be sadly uninformative:

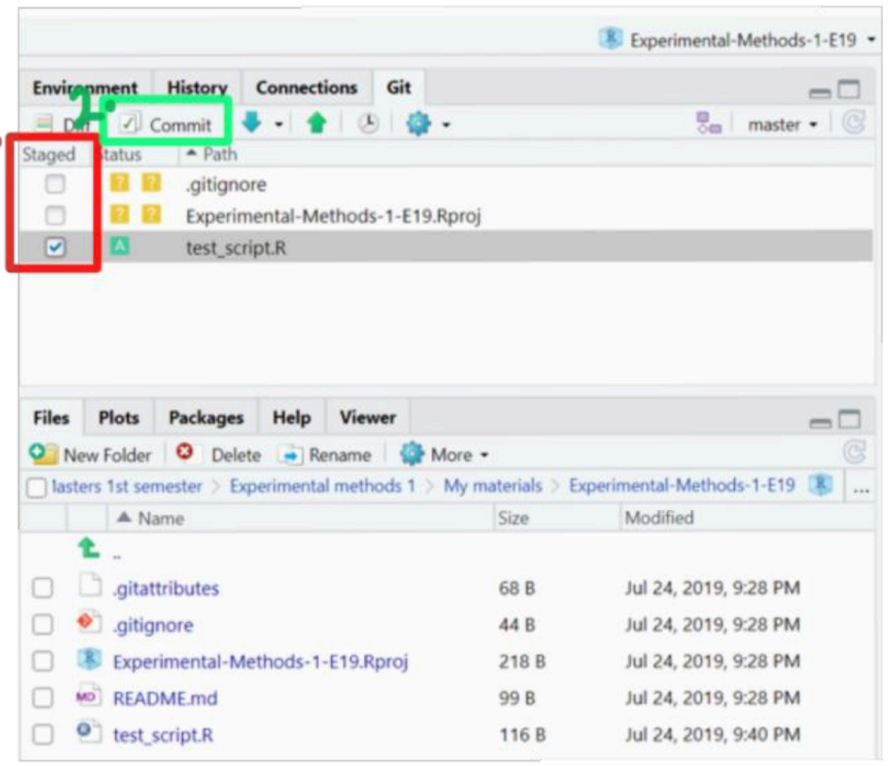
```
> pet_frame <- data.frame(names=names n_pets=n_pets)
Error: unexpected symbol in "pet_frame <- data.frame(names=names n_pets"
```

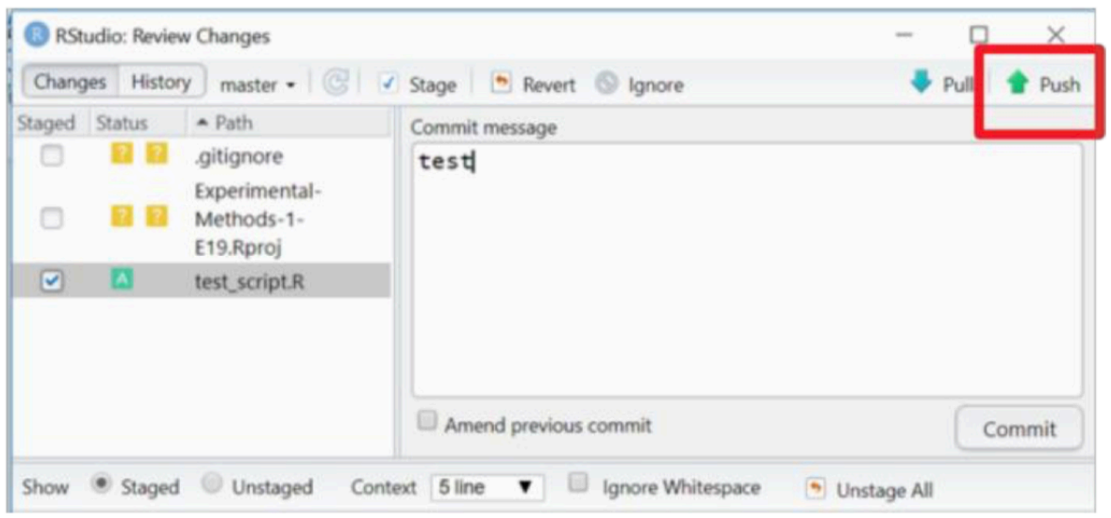
You can always google if you are in doubt

It gets better with time - the more errors you make, the more you learn how to deal with them

GITHUB (LIVE DEMO)

- Will be used to link your code (and everything necessary to run it) in your portfolio





GETTING HELP

- Use '?' or '??' or help() in R
- Check the book and previous scripts (GitHub)
- Use the internet: google, stackoverflow etc.
- Use the Facebook group CogOverflow
- Cheat sheets
- Study café on Tuesdays
- Contact me :)
- Nice R tutorials can be found here