Lecture 1.4 – Intro to R and RStudio

Specific Learning Objectives:

1.1.1 – Understand how to use the command line.

1.1.2 – Understand how to use the help function of R.

1.1.3 – Understand the basic syntax of the R language.

1.1.4 – Execute inbuilt mathematical functions to perform calculations in R.

1.1.5 – Learn how to assign variables.

1.1.6 – Understand the basic syntax of functions in R.

1.1.7 – Open, edit, and save a script in RStudio's editor.

2.2.1 – Create reproducible scripts in R.

Downloading *R* and RStudio

Download *R*:
https://www.r-project.org/

1. Go to https://cloud.r-project.org/
2. Select your operating system.
3. Select the latest release that is "notarized and signed."
4. Save and open the file, follow the instructions to install.

Download RStuido:
https://rstudio.com/products/rstudio/download/

1. Select the RStudio Desktop version.
2. Download, open, and follow instructions to install.
3. Open RStudio to get started!

About *R*

- A high-level interpreted language designed primarily to run statistical tests and visualize data.
  - Based on the language *S*, which is not used anymore.
  - Free, open-source language (meaning anyone can access the base code all the way down to compilers!)
  - Base *R* code is written in C, C++, Fortran, and *R*.
  - Interpreted language: meaning you don't worry about compiling machine code, it happens automatically in the background.
  - *R* installations come with a simple command line and script editor.

About RStudio

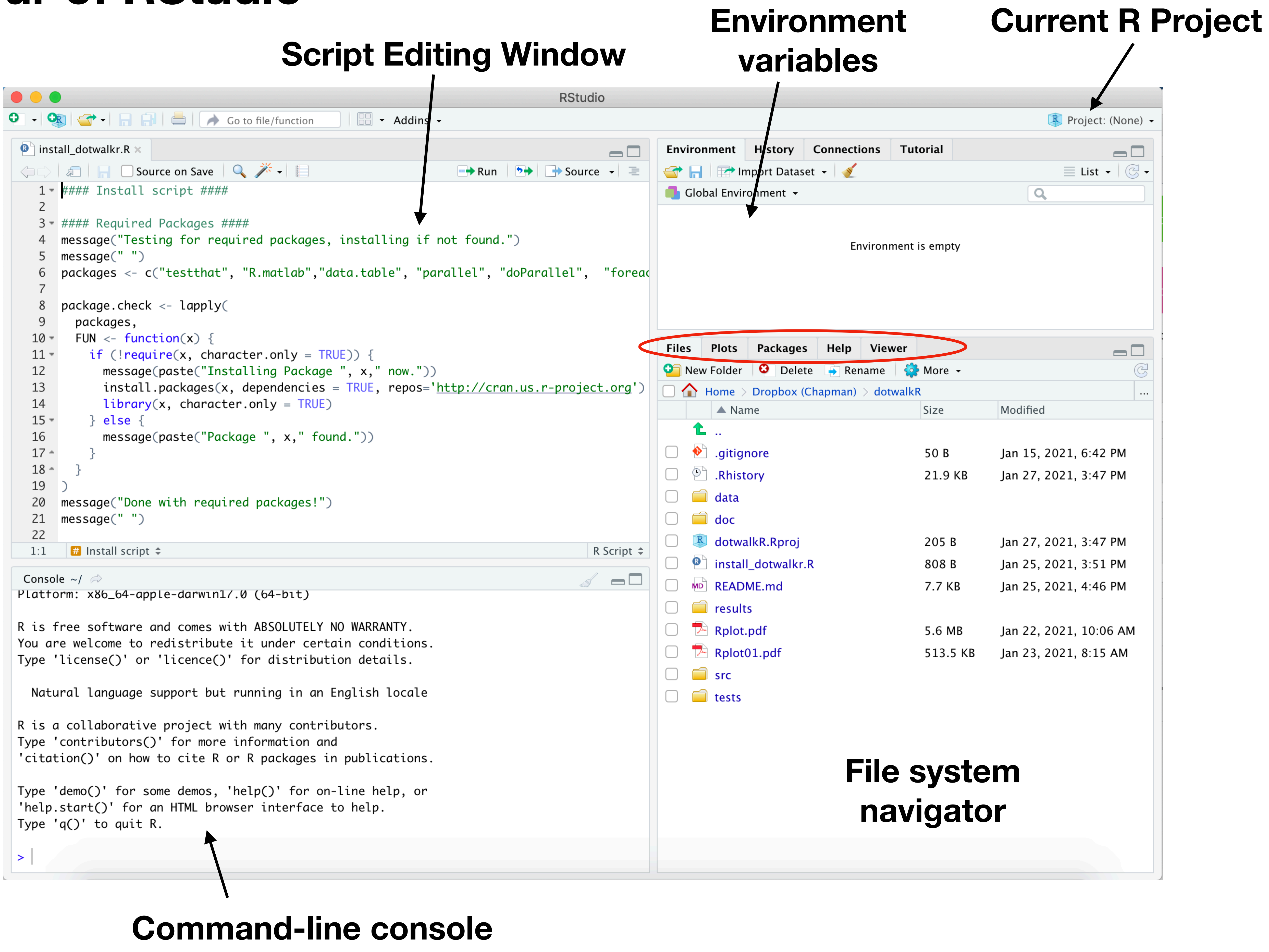
- A Integrated Development Environment (IDE) for the *R* language.

  - RStudio is basically a wrapper with some helpful features! It is not *R* itself, which has to be downloaded separately.

  - Helpful features include:

    - Command-line prompt
    - Help viewer
    - Integrated plot editor
    - Jobs & testing environments
    - Memory management tools
    - File system navigator
    - Package Manager
    - Script Editor
    - Integrated RMarkdown & Knitr manager
    - Version Control panel

A Tour of RStudio



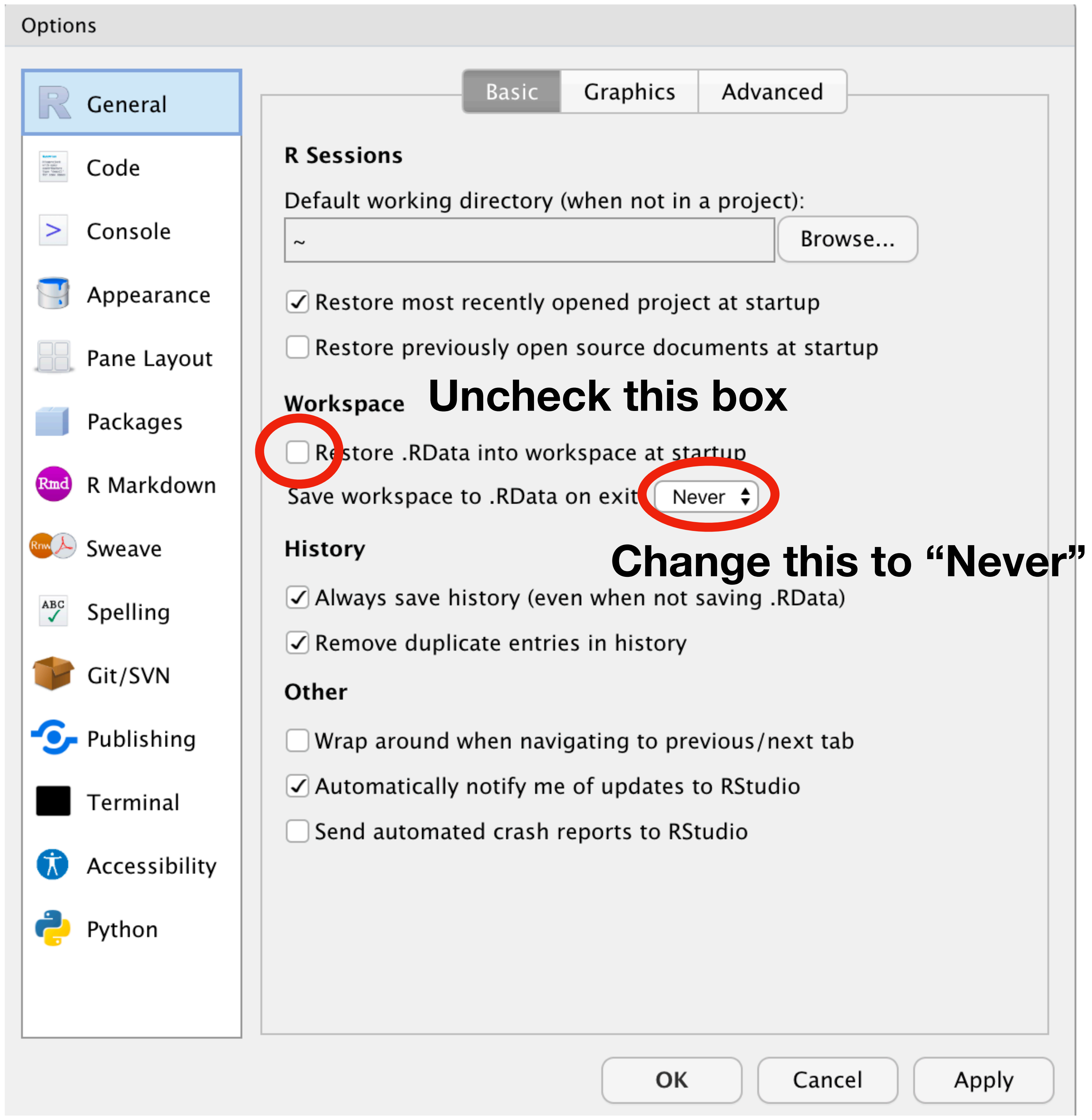
```r
#### Install script ####

#### Required Packages ####
message("Testing for required packages, installing if not found.")
message(" ")
packages <- c("testthat", "R.matlab","data.table", "parallel", "doParallel",  "foreac

package.check <- lapply(
  packages,
  FUN <- function(x) {
    if (!require(x, character.only = TRUE)) {
      message(paste("Installing Package ", x," now."))
      install.packages(x, dependencies = TRUE, repos='http://cran.us.r-project.org')
      library(x, character.only = TRUE)
    } else {
      message(paste("Package ", x," found."))
    }
  }
)
message("Done with required packages!")
message(" ")
```

```
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Size	Modified
..		
.gitignore	50 B	Jan 15, 2021, 6:42 PM
.Rhistory	21.9 KB	Jan 27, 2021, 3:47 PM
data		
doc		
dotwalkR.Rproj	205 B	Jan 27, 2021, 3:47 PM
install_dotwalkr.R	808 B	Jan 25, 2021, 3:51 PM
README.md	7.7 KB	Jan 25, 2021, 4:46 PM
results		
Rplot.pdf	5.6 MB	Jan 22, 2021, 10:06 AM
Rplot01.pdf	513.5 KB	Jan 23, 2021, 8:15 AM
src		
tests		

Adjusting Preferences

- Open RStudio's preferences:
  - On Mac:
    Top bar > RStudio > Preferences…
  - On Windows:
    Top bar > Tools > Options…
- We don't want your workspace saved!!!



Options

R Sessions

Default working directory (when not in a project): ~ Browse...

[✓] Restore most recently opened project at startup

[ ] Restore previously open source documents at startup

Workspace **Uncheck this box**

[ ] Restore .RData into workspace at startup

Save workspace to .RData on exit: Never

Change this to "Never"

History

[✓] Always save history (even when not saving .RData)

[✓] Remove duplicate entries in history

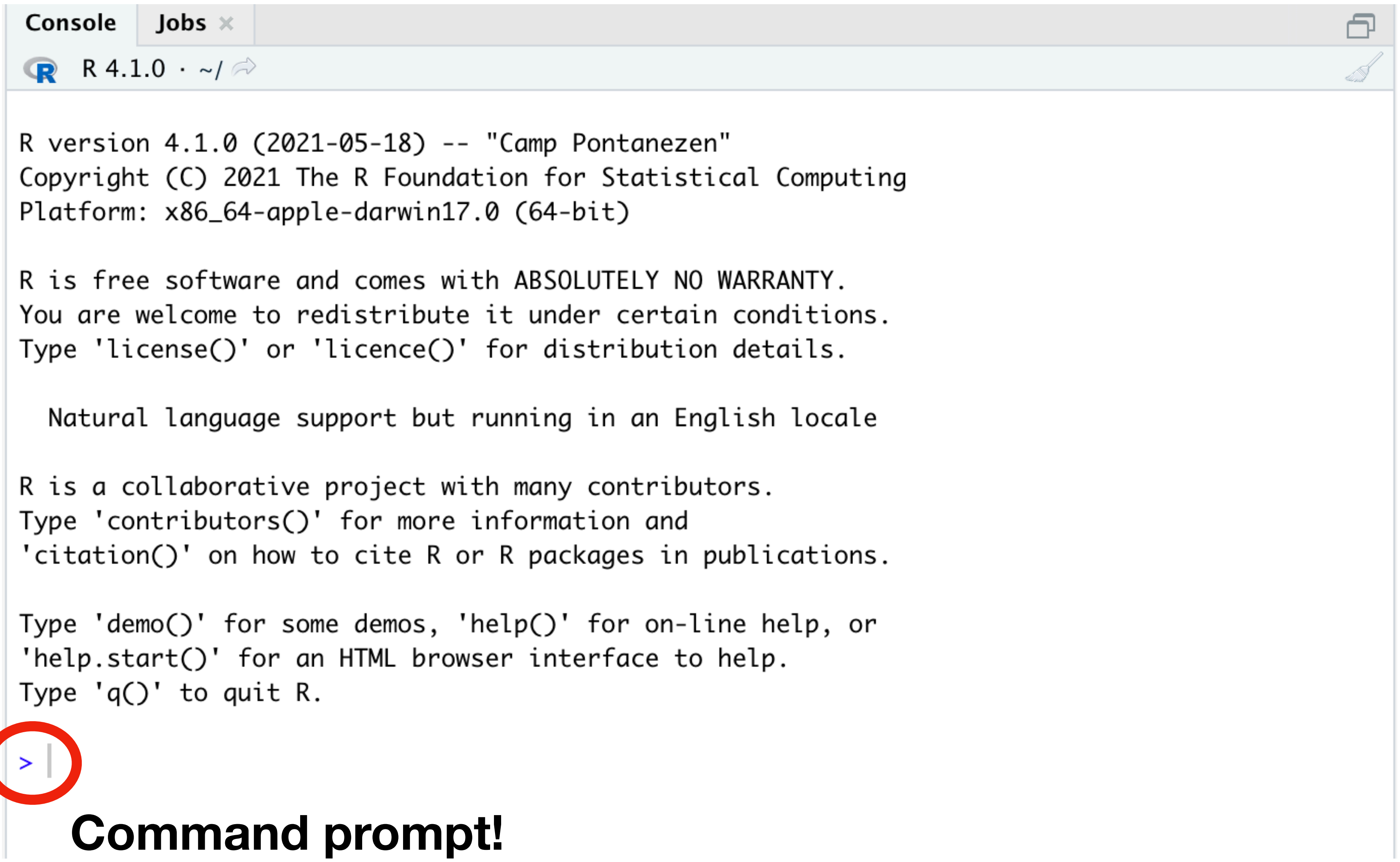
Other

[ ] Wrap around when navigating to previous/next tab

[✓] Automatically notify me of updates to RStudio

[ ] Send automated crash reports to RStudio

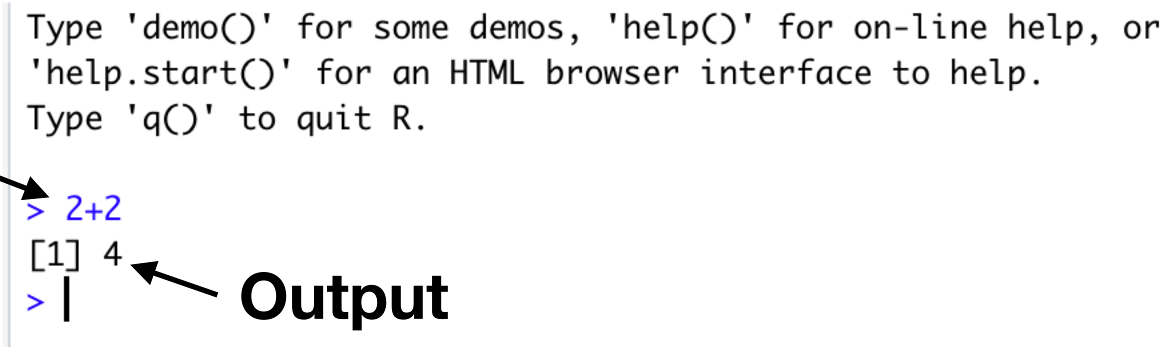
Using Command Line



```
R version 4.1.0 (2021-05-18) -- "Camp Pontanezen"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Command prompt!

- Put your cursor here and enter something!



Input

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 2+2
[1] 4
> |
```

Output

***R* can function as a basic calculator!**

Let’s just make R produce errors

- We’re going to take a quick break to make R produce some errors. I make R errors all the time, constantly, even if it might look like I always know what I’m doing!

- Desensitize yourself to errors, they really aren’t a big deal. You’re not going to break anything.

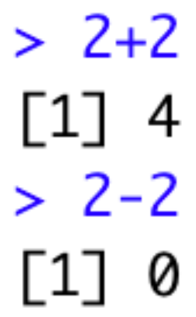
- Feel free to swear at R, I don’t care and neither does it.

- A huge part of learning to code is learning to tolerate being frustrated! It’s ok and totally normal. Just don’t let it consume you, take a break! Do something else! You’ll get it eventually!

R as a calculator

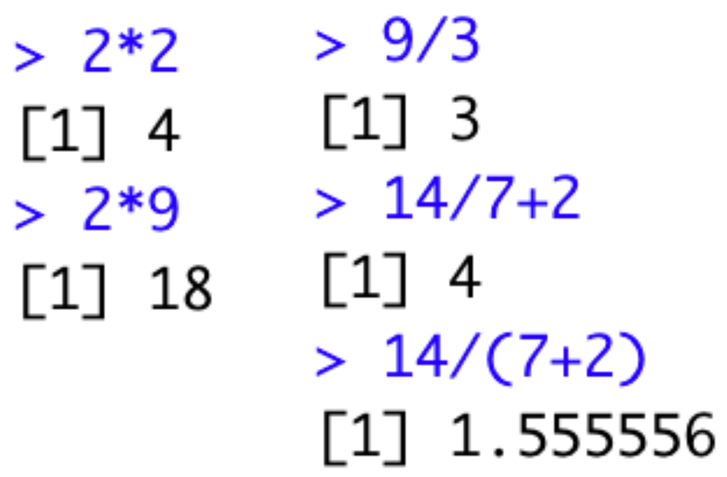
- R can act as a basic and scientific calculator

  - Addition and Subtraction



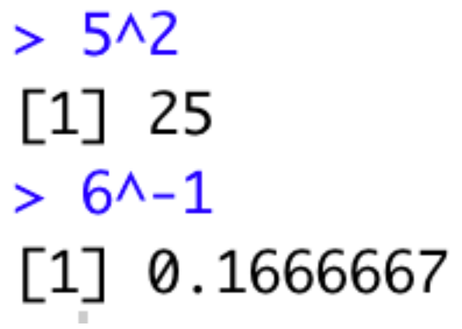
```
> 2+2
[1] 4
> 2-2
[1] 0
```

  - Multiplication and Division

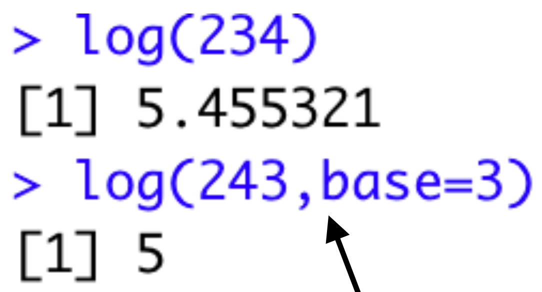


```
> 2*2       > 9/3
[1] 4       [1] 3
> 2*9       > 14/7+2
[1] 18      [1] 4
            > 14/(7+2)
            [1] 1.555556
```

  - Exponents and Logarithms



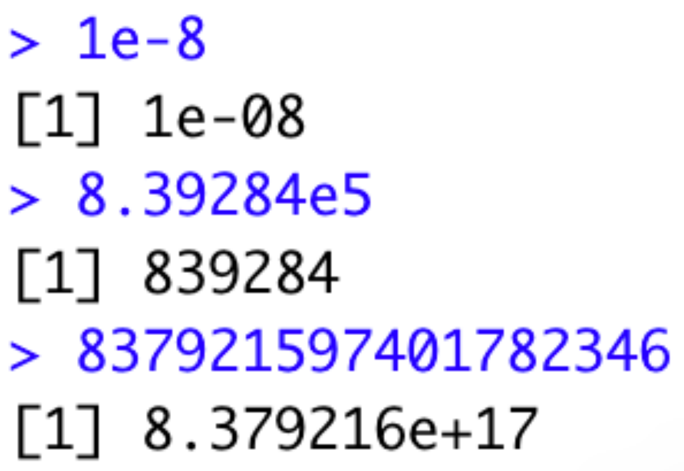
```
> 5^2
[1] 25
> 6^-1
[1] 0.1666667
```



```
> log(234)
[1] 5.455321
> log(243,base=3)
[1] 5
```

It's a function!!

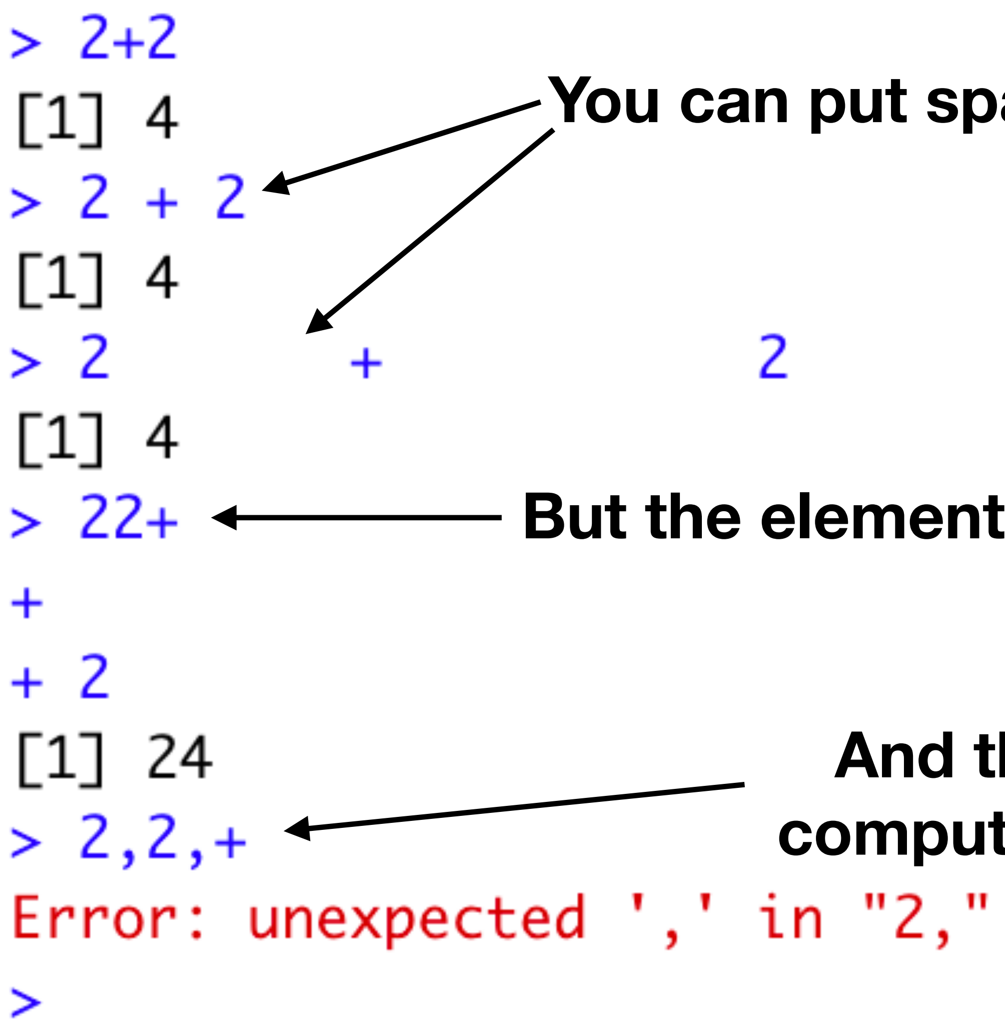
  - e notation



```
> 1e-8
[1] 1e-08
> 8.39284e5
[1] 839284
> 837921597401782346
[1] 8.379216e+17
```

Syntax in R

- Syntax is the set of rules that define what various combinations of symbols mean.
    - Each computing language has a different set of rules of how you can arrange symbols to tell the computer what to do.



```
> 2+2
[1] 4
> 2 + 2
[1] 4
> 2       +             2
[1] 4
> 22+
+
+ 2
[1] 24
> 2,2,+
Error: unexpected ',' in "2,"
>
```

You can put spaces and tabs in between elements

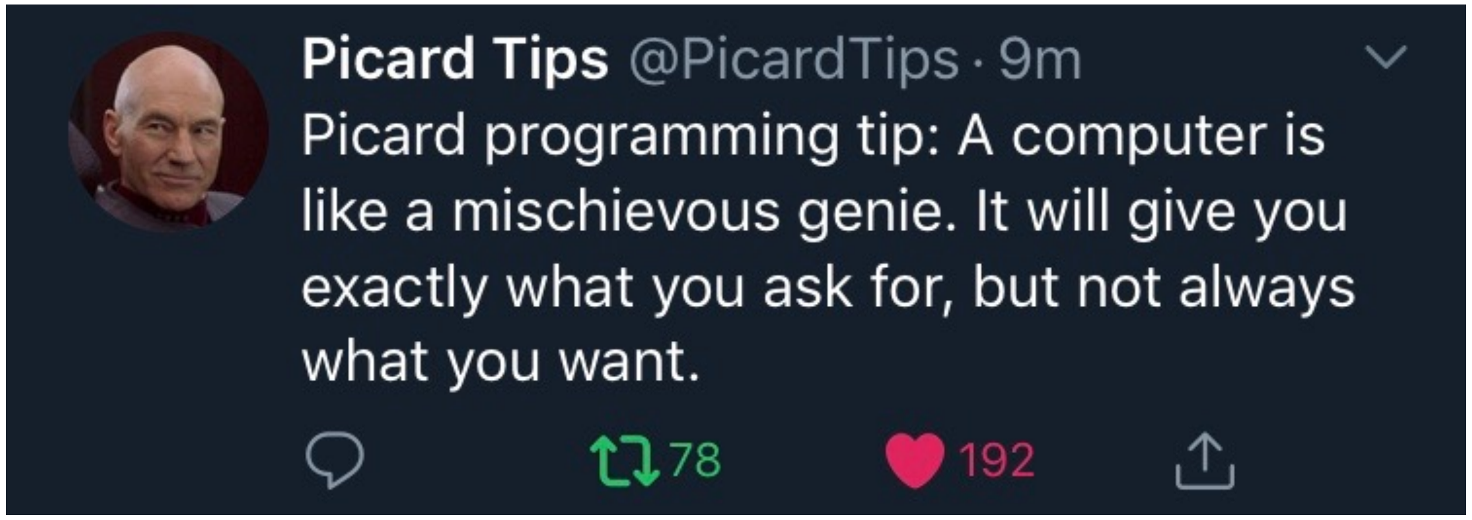
But the elements need to be in the correct order!

And they must make sense to the computer (no commas allowed here!)

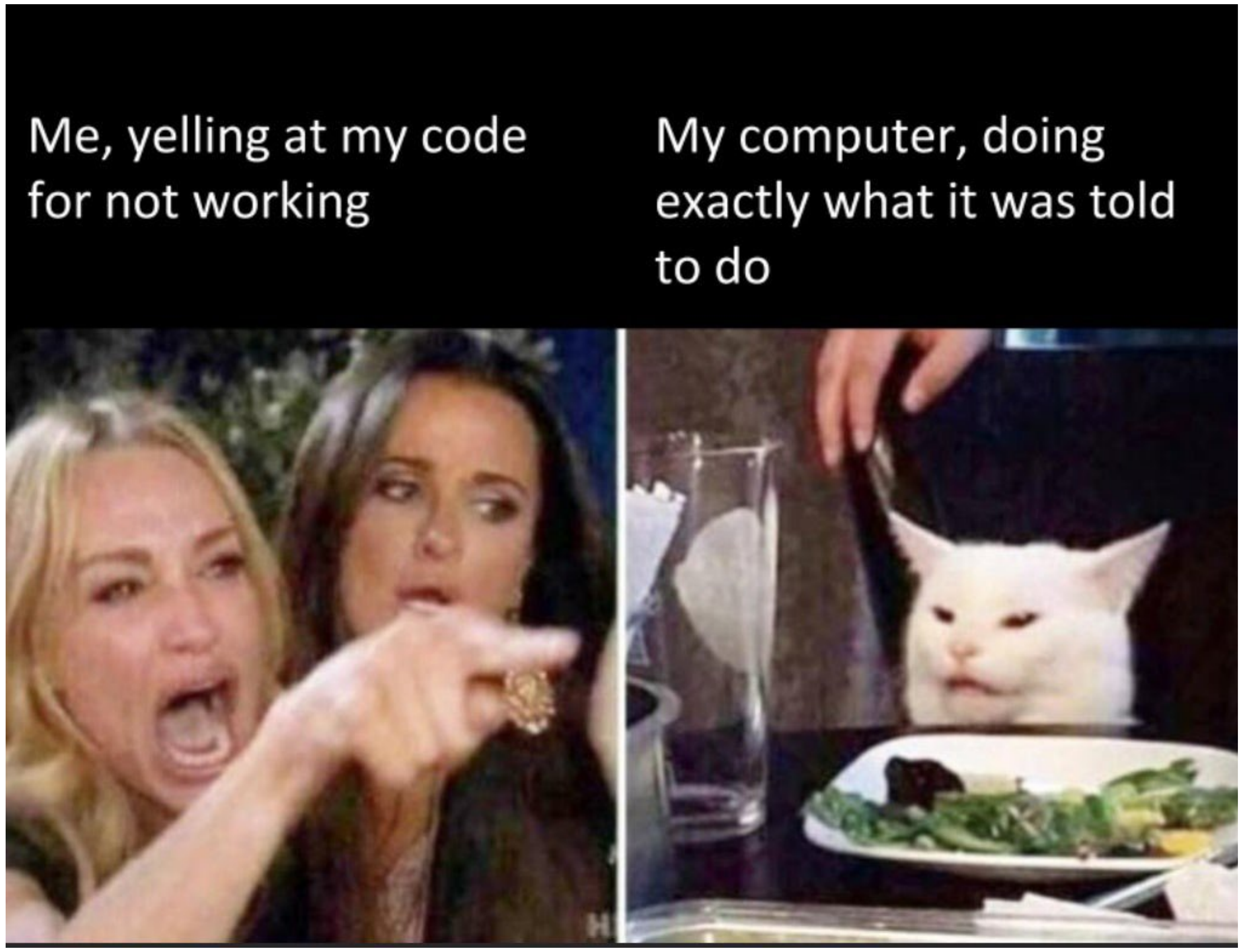
A lot of your errors, especially at first, will be syntax errors. This is ok, just fix it and try again!

Some tips for using *R*

- **Remember that computers do EXACTLY what you tell them to, but that's often not what you THINK you are telling them to do!**


Picard Tips @PicardTips · 9m
Picard programming tip: A computer is like a mischievous genie. It will give you exactly what you ask for, but not always what you want.
78 192

- **Computers really only do what you tell them to do. It's always your fault. (It's always my fault too.)**


Me, yelling at my code for not working
My computer, doing exactly what it was told to do

A lot of your errors, especially at first, will be syntax errors. This is ok, just fix it and try again!

Check Your Understanding

Have R calculate the square root of 164956 using the `sqrt()` function. What is the correct output?

Correct answer

a) `[1] 406.1478`

b) `[1] 12.8841`

c) `Error: unexpected input in "√"`

d) `Error in sqrt(164956, 2) : 2 arguments passed to 'sqrt' which requires 1`

Error in c: I used the sign √164956, even though I understand what this means, R doesn't understand so returns an error.

Error in d: I put too many numbers (or arguments) in the `sqrt()` function so it's telling you you need one argument only!

Assigning variables and objects

- You'll notice that when you add 2+2, R will give you an output, but it doesn't save or "store" that value anywhere. (Look at your environment, it is empty!)

- Assigning variables (objects): two methods

```
> a <- 2
```

- Assign arrow (less than sign and dash)

```
> b = 3
```

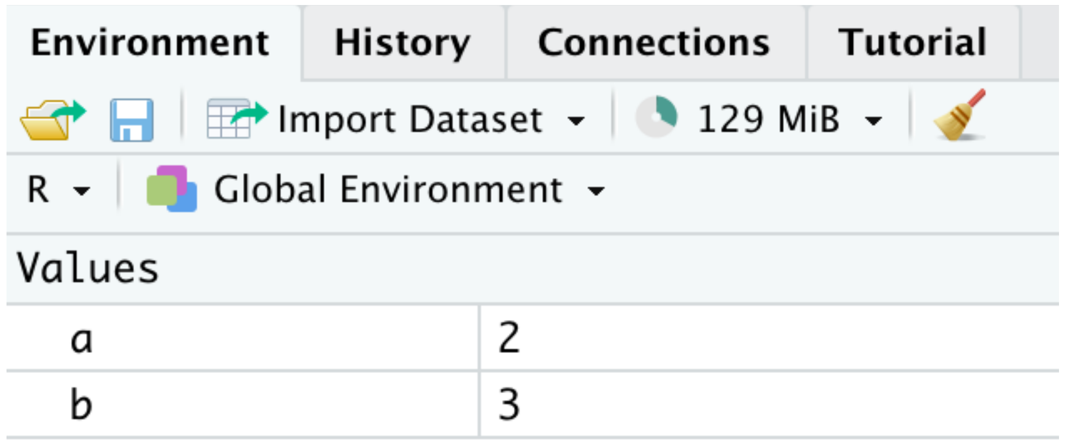
- Equal sign

For historical reasons, this is the preferred method for variables…

…and this is the preferred method for arguments inside functions.

Your environment tells you what's in R's memory! This is empty when R restarts!!

- Assigned variables appear in your environment!



Values

a	2
b	3

Check Your Understanding

Syntax is important here! Using a space between the < and - will change the meaning of the command.

Try these: do they all have the same result? Why or why not?

a) `x<-2`   b) `x <- 2`   c) `x< - 2`

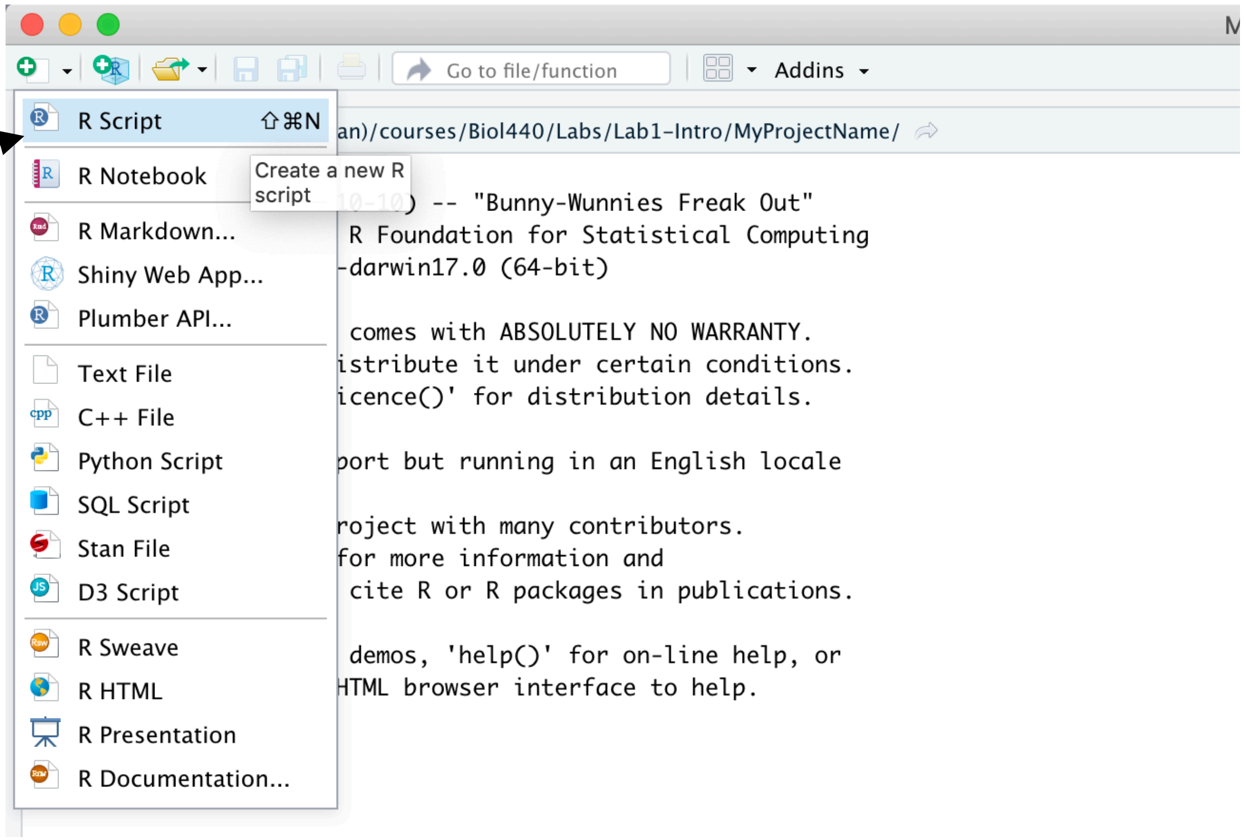
Does it matter which way the arrow points?

Try these: which work and which produce errors?

d) `7 -> j`   b) `7 <- j`   c) `j -> 7`

Your First R Script

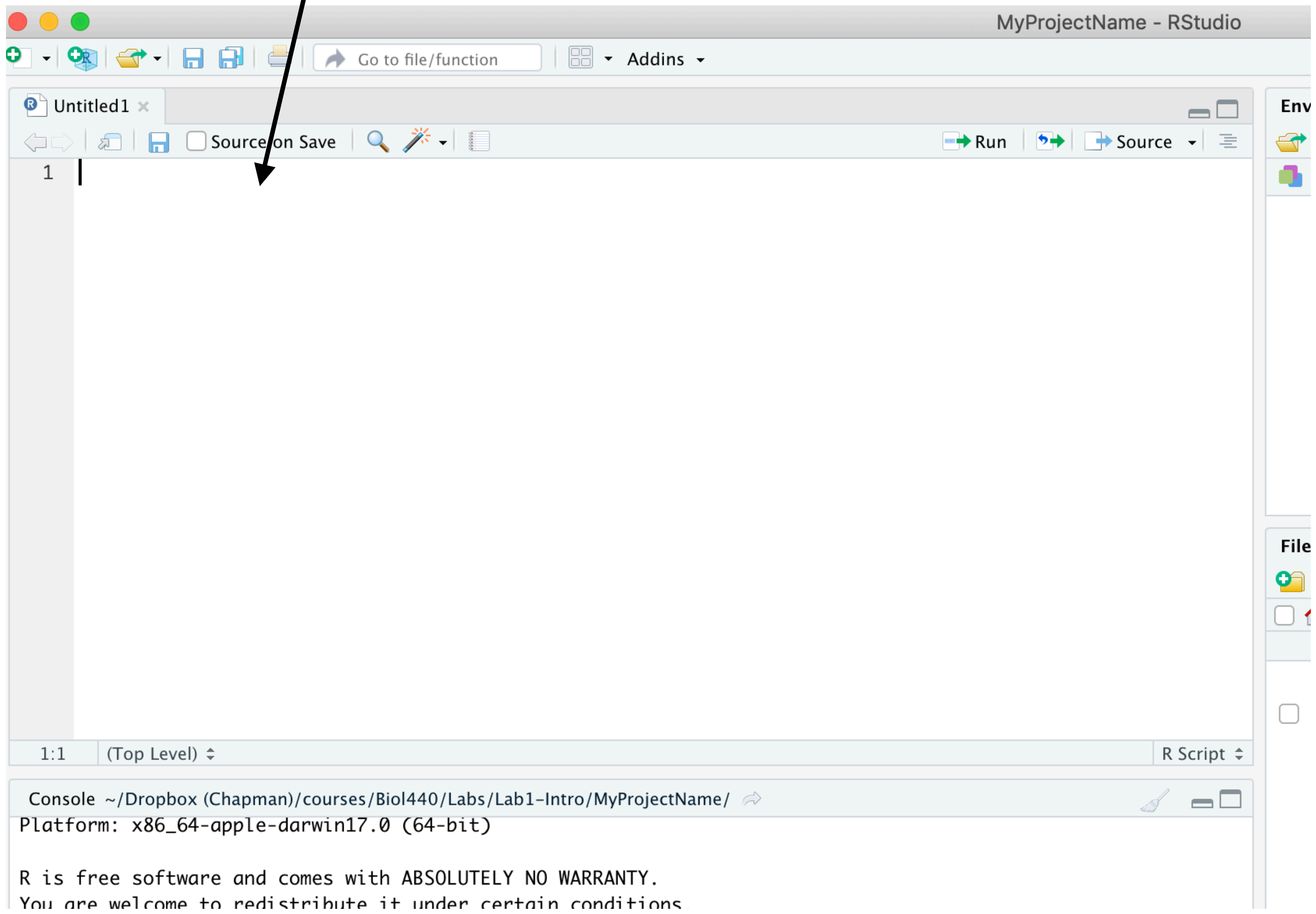
Click on "New File" tab the in upper left and select "R Script".



an)/courses/Biol440/Labs/Lab1-Intro/MyProjectName/

```
) -- "Bunny-Wunnies Freak Out"
 R Foundation for Statistical Computing
-darwin17.0 (64-bit)

 comes with ABSOLUTELY NO WARRANTY.
istribute it under certain conditions.
icence()' for distribution details.

port but running in an English locale

roject with many contributors.
for more information and
 cite R or R packages in publications.

 demos, 'help()' for on-line help, or
HTML browser interface to help.
```

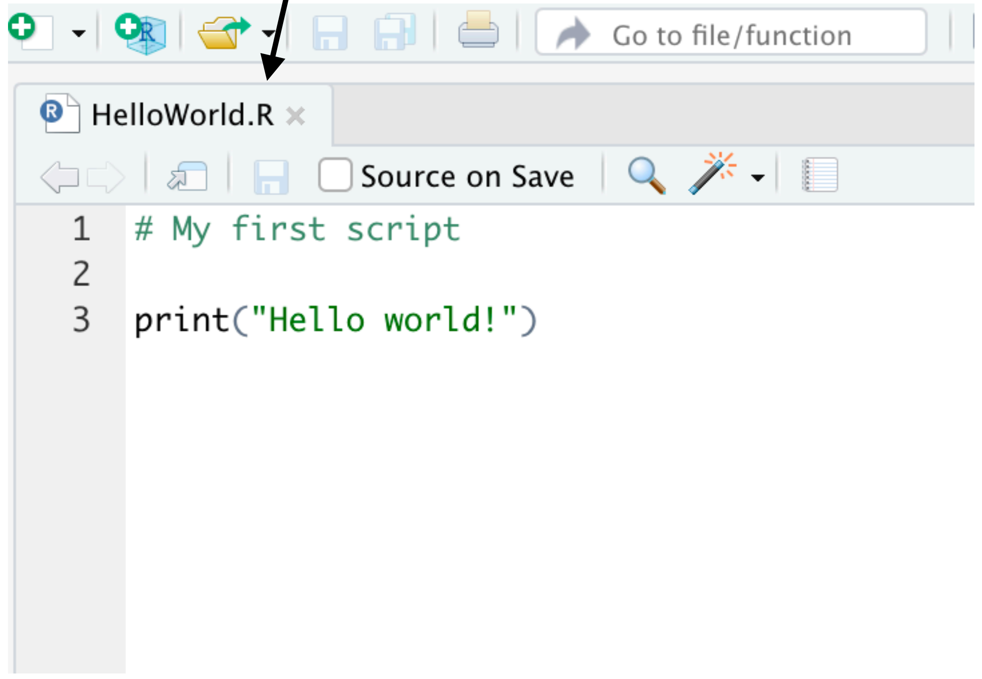
This will open a window in the script editor. You're ready to code!



Console ~/Dropbox (Chapman)/courses/Biol440/Labs/Lab1-Intro/MyProjectName/

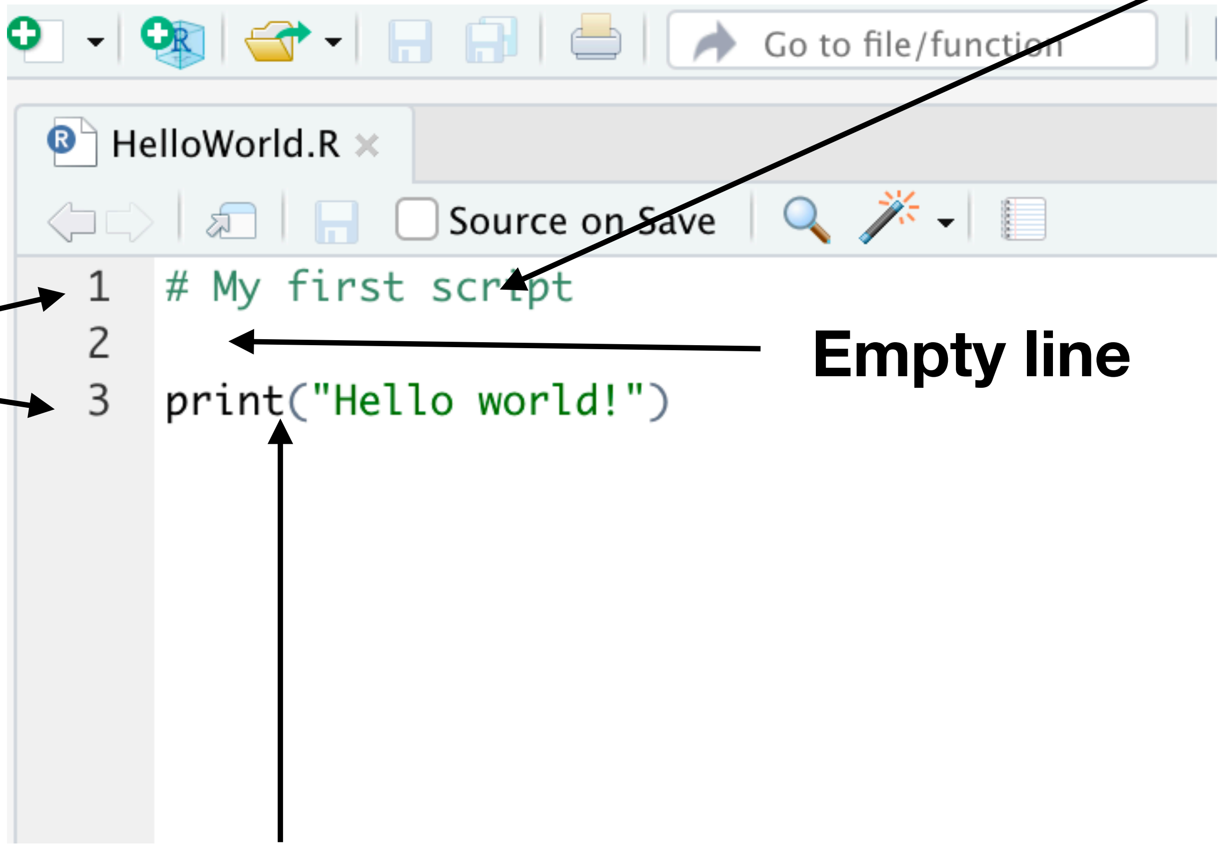
```
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions
```

Be sure to save your script with a name that ends in .R



```r
# My first script

print("Hello world!")
```

Your First R Script



```
1 # My first script
2
3 print("Hello world!")
```

Line numbers

- Useful for referencing specific bits of code
- Will show up in error messages

Empty line

A function!

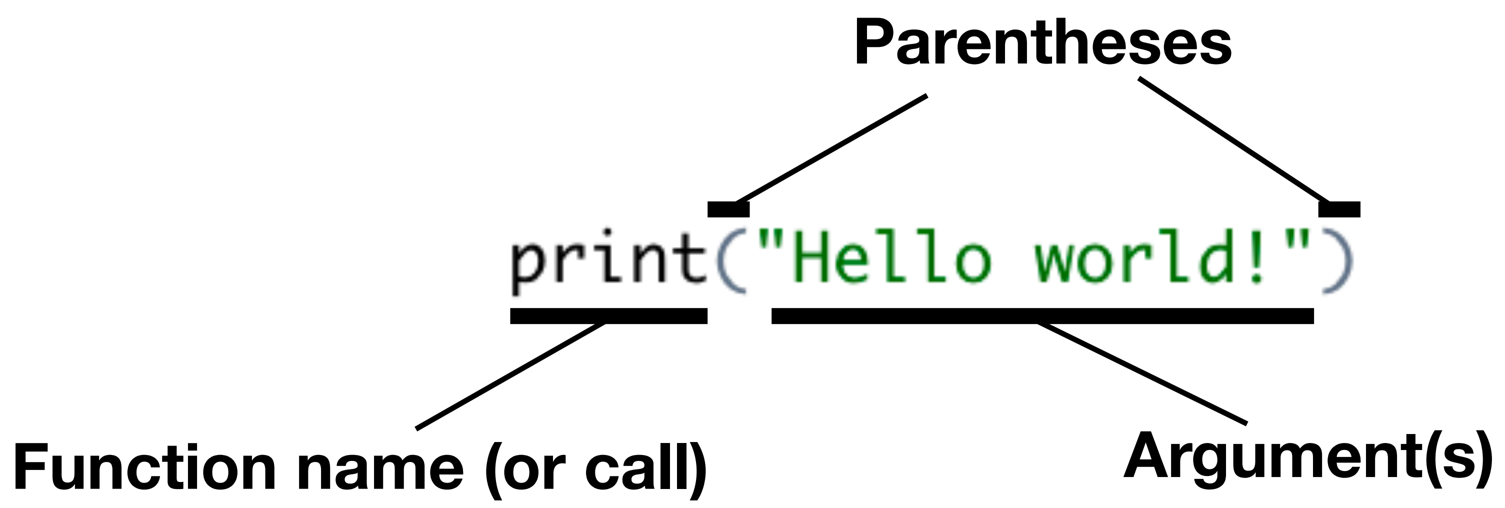
A "comment"

- Anything after a hash symbol #
- Skipped by R (not evaluated)
- Appears in green in script editor
- Really useful for documenting work

Functions in R

- Functions reference other bits of code that you can run by “calling” it (written by someone else or you)

- Syntax of a function in *R*:



```
print("Hello world!")
```

- **Function name (or call)**: the name of the function you desire to use, or how to “call” the function.

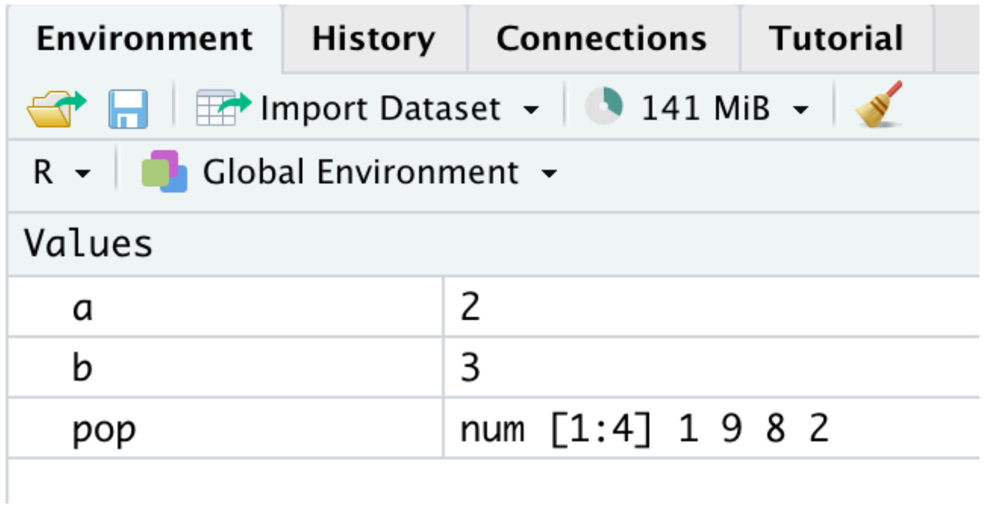
- **Parentheses**: Parentheses after the call is how R knows you want to call a function. You MUST put parentheses, even if the function takes no arguments, or else R will think you are trying to recall a variable!

- **Arguments**: options that you’d like to pass to the function.

Functions in R

- Use a function to create a vector:

```
> pop <- c(1, 9, 8, 2)
```



Values

a	2
b	3
pop	num [1:4] 1 9 8 2

- Use another function to calculate the mean:

```
> mean(pop)
[1] 5
```

- You can nest functions within each other:

```
> mean(c(1, 9, 8, 2))
[1] 5
```

How many arguments a function can take is determined by the function!

- Functions can have many arguments, separated by commas:

arg 1 **arg 2** **arg 3**

```
> blep <- seq(from = 1, to = 10, by = 0.5)
```

Creates a vector that is a sequence of numbers from 1 to 10 by 0.5

R has lazy evaluation!

```
> blep <- seq(1, 10, 0.5)
```

Functions in R

Check your understanding!

What happens if you enter the numbers into mean without the **c( )**? Or like this:

```
> mean(1, 9, 8, 2)
```

Why does it do this?

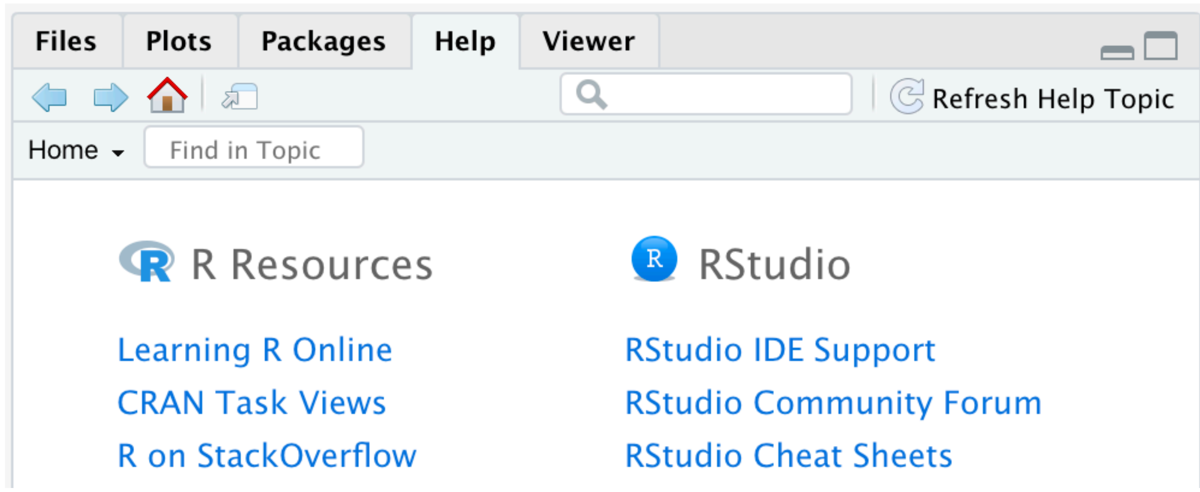
When things go sideways… use help!

- Help documentation exists for each function in R.
  - Help viewer is in the lower right-hand corner of RStudio as a tab.
  - Access help documentation for a function:

```
> ?mean
```

  - Do a keyword search:

```
> ??mean
```

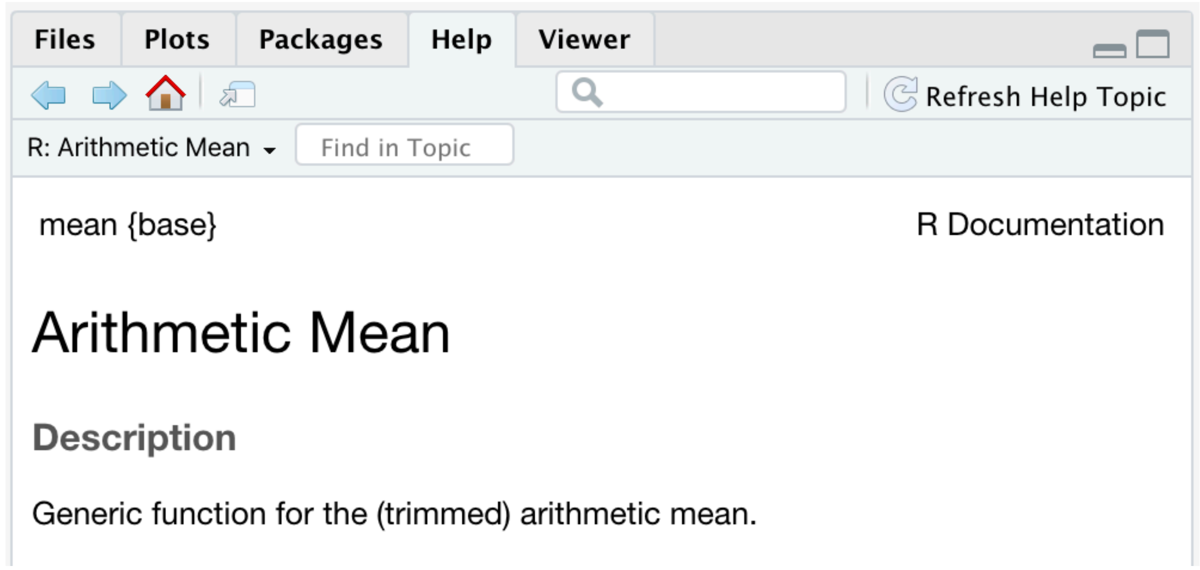


R Resources

Learning R Online
CRAN Task Views
R on StackOverflow

RStudio

RStudio IDE Support
RStudio Community Forum
RStudio Cheat Sheets

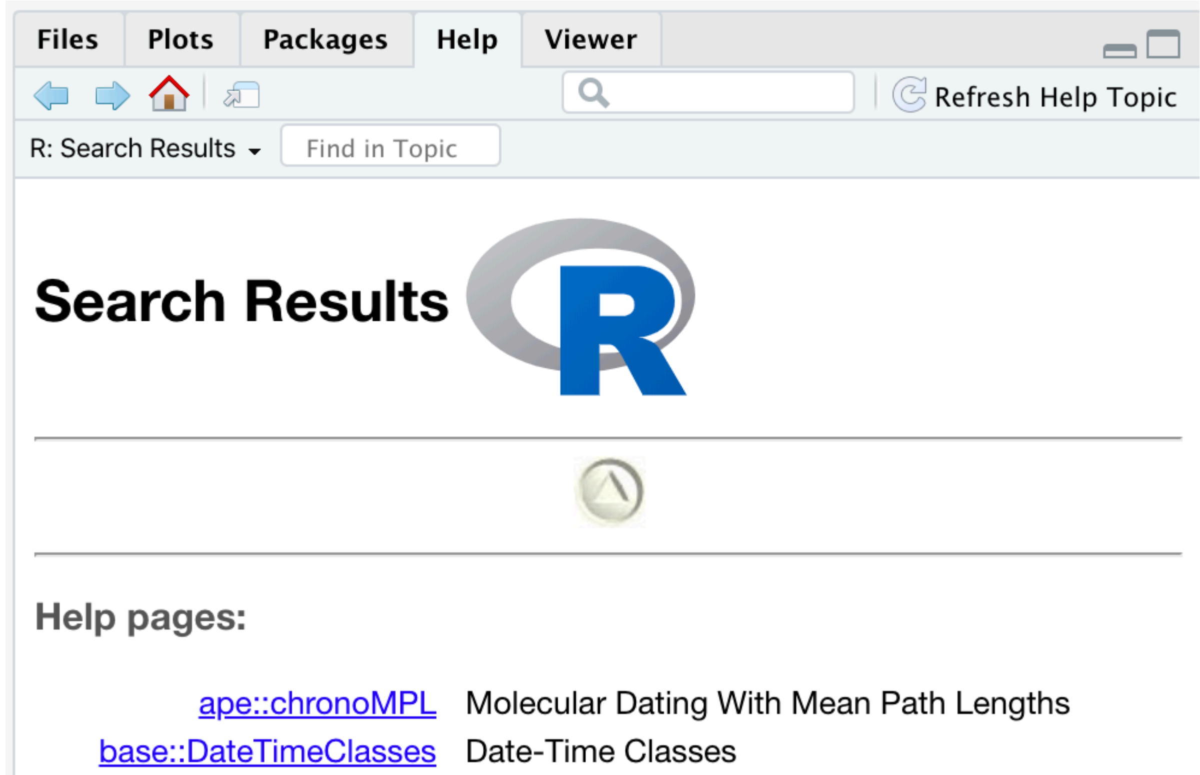


mean {base} R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.



Search Results

Help pages:

ape::chronoMPL	Molecular Dating With Mean Path Lengths
base::DateTimeClasses	Date-Time Classes

Functions in R

Check your understanding!

What happens if you enter the numbers into mean without the **c( )**? Or like this:

```
> mean(1, 9, 8, 2)
```

Why does it do this?

Why would the mean not calculate correctly based on the help documentation for **mean( )**?

More R Tips

- Use scripts to document code used during lecture! Make comments about what does and doesn't work, why errors occur, etc.

- Avoid using command line when possible, put it in a script!

- Make sure to write code that's evaluated from top to bottom without errors!

- You will probably get pretty frustrated. That's fine, just put it down and try again later.

Action Items

1. **Complete Assignment 1.5**

2. **Read Davies Chapters 3, 4, and 6 for next time.**