Introduction to R & RStudio

Washington University in St. Louis

June 9, 2020 – June 25, 2020

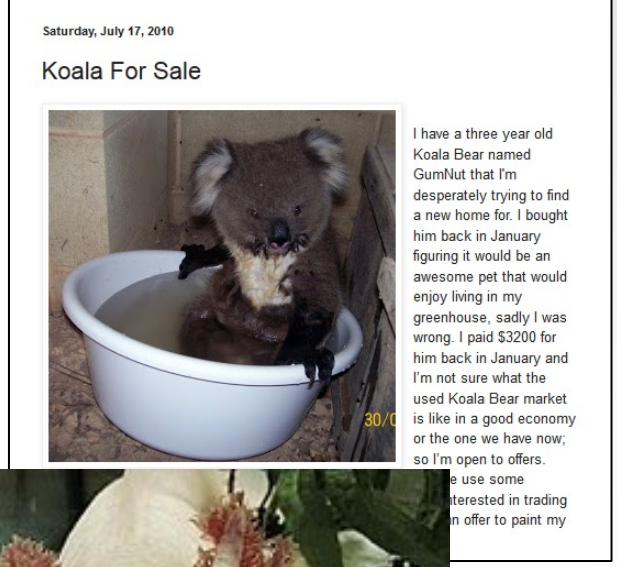
About Us

Kendra Smith – 6th Year
- Cognitive psychology
- GumNut

Emorie Beck – 5th Year
- Personality psychology
- Axolotl

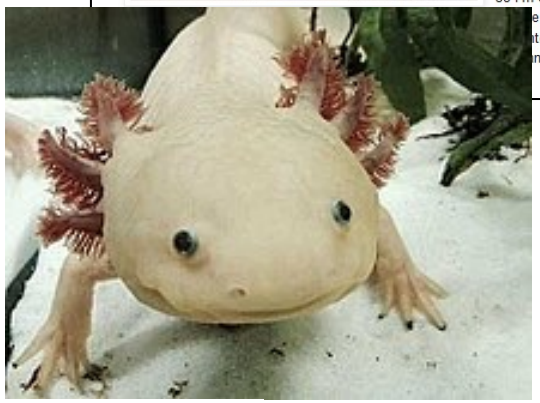
Maddy Frumkin – 4th Year
- Clinical psychology
- Bernese mountain dog

Violet Brown – 3rd Year
- Cognitive psychology
- Gorillas


Saturday, July 17, 2010

Koala For Sale

I have a three year old Koala Bear named GumNut that I'm desperately trying to find a new home for. I bought him back in January figuring it would be an awesome pet that would enjoy living in my greenhouse, sadly I was wrong. I paid $3200 for him back in January and I'm not sure what the used Koala Bear market is like in a good economy or the one we have now; so I'm open to offers.



About YOU

This Workshop

6 sessions

- We will send out slides at the end of each class.
- You will have access to all code we write in class – don't worry if you don't get every single thing down!

Post-its for exercises in RStudio

- Chat = have a question (can PM leader directly if you prefer)
-  = done with exercise

Why R?

FREE!

Open source (user-created packages)

Learning R will help you learn other programming languages

- Example: MATLAB, Python, web programming

Flexible

- Can do anything you want it to do

(Did we mention it's free?)

What is R?

R is a programming language

- Statistics & graphing

RStudio is an environment that makes it easy to actually use R

Ask questions

Make mistakes!

When in doubt, Google

SETTING UP R & RSTUDIO

Installing R & RStudio

1. Install R first

   - **Mac users:**
     1) Go to: http://cran.wustl.edu/ → download R for (Mac) OS X
     2) Choose the correct version of R to install:
     - **R-4.0.0.pkg** will likely be the case for most of you

   - **Windows users**:
     1) Go to: http://cran.wustl.edu/ → download R for Windows
     2) Click on the **base** folder --> click on **Download R 4.0.0 for Windows**

2. Then, install RStudio

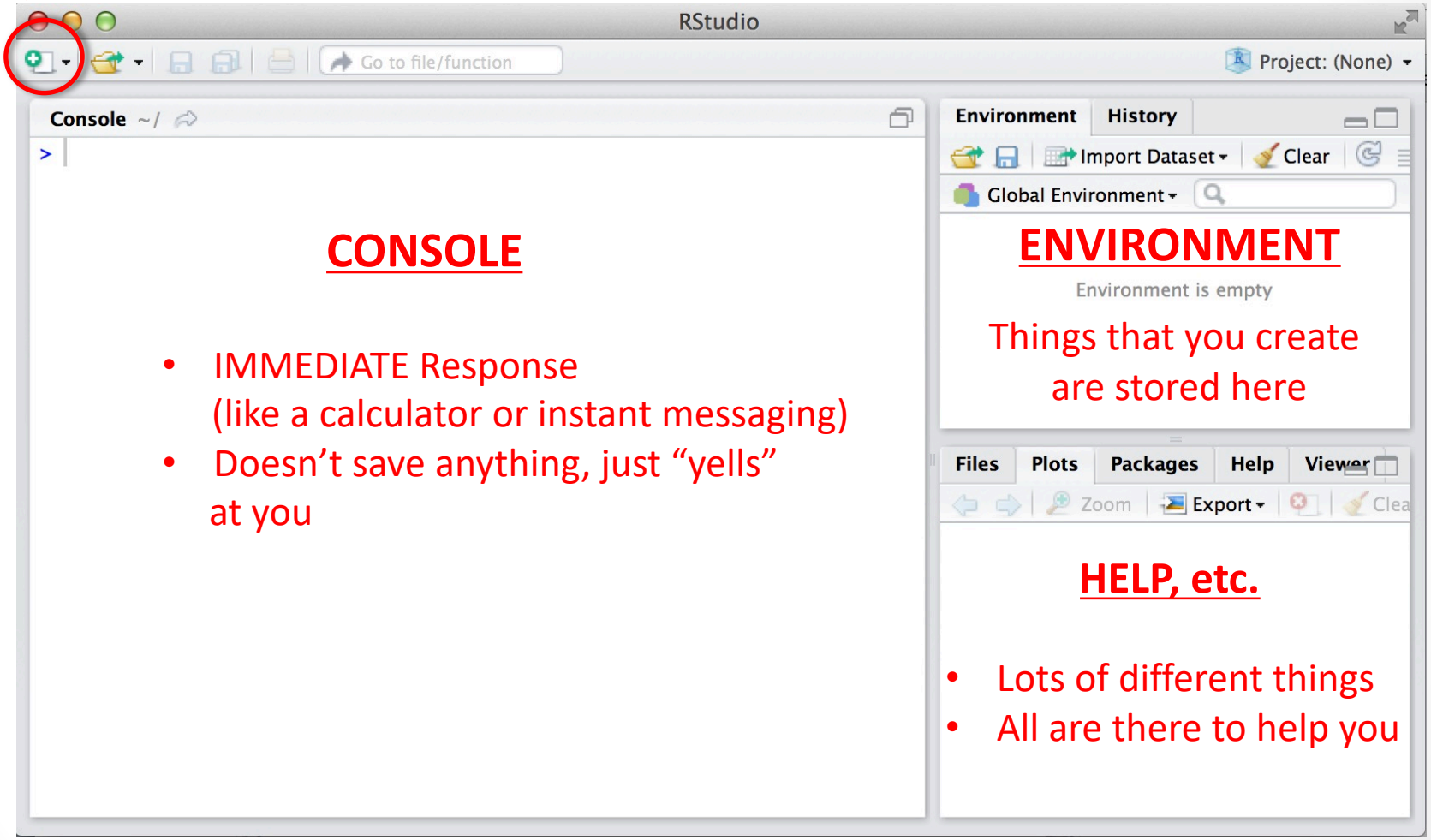
   1) Go to https://www.rstudio.com/products/rstudio/download/

   2) Scroll to the bottom, to **"All Installers"**

   3) Click the first link to download the installer for Windows, OR click the second link to download the installer for Mac

R Script

Getting Oriented with RStudio



CONSOLE

- IMMEDIATE Response
  (like a calculator or instant messaging)
- Doesn't save anything, just "yells"
  at you

ENVIRONMENT

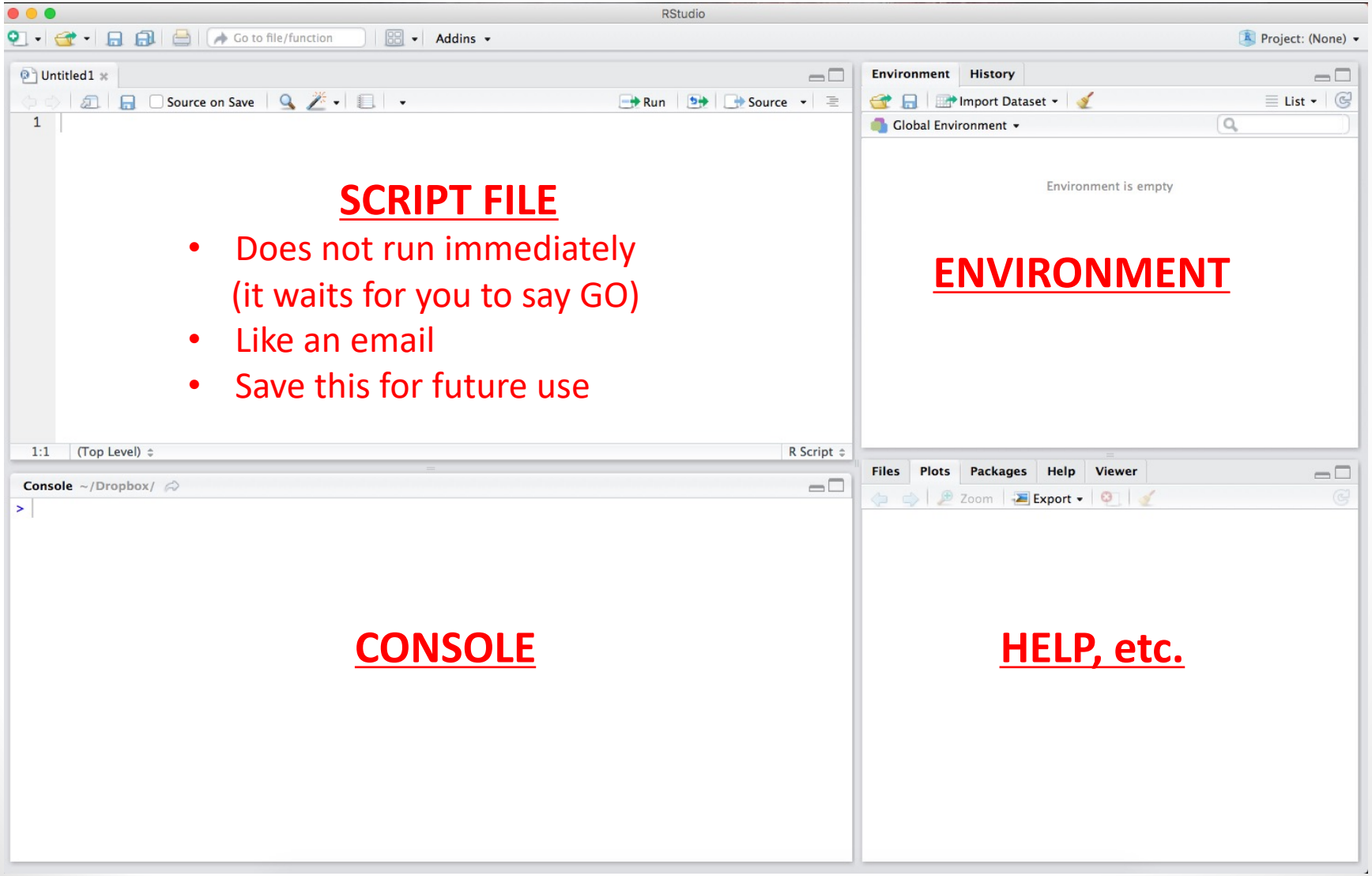
Environment is empty

Things that you create are stored here

HELP, etc.

- Lots of different things
- All are there to help you

Getting Oriented with RStudio



How to run code from your .R file

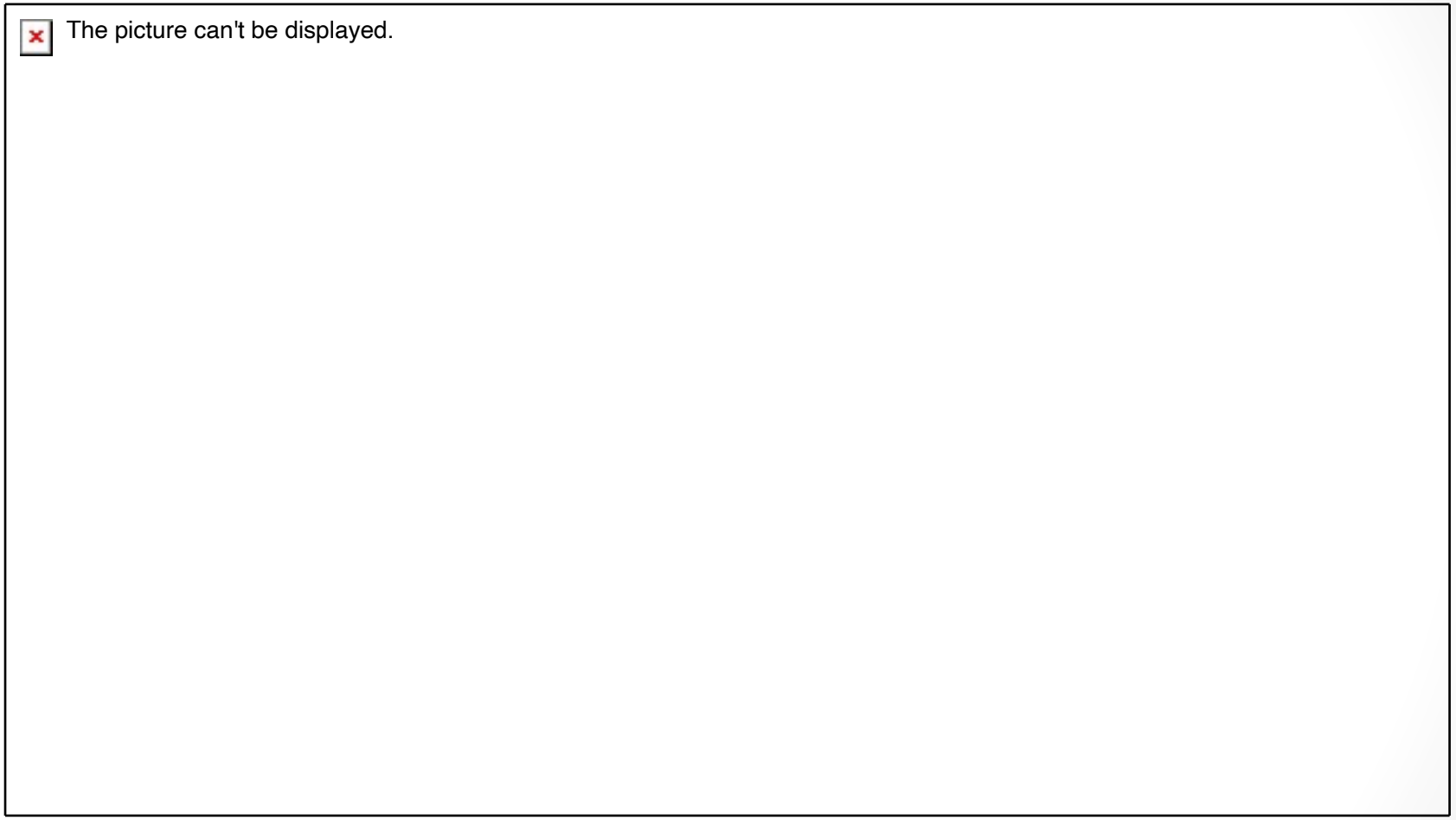
Highlight the lines that you want to run, then…

- Click on **Run** button

OR

- Ctrl + Enter (Windows)
- Command + Enter (Mac)

EXAMPLE 1 IN R



QUESTIONS?
ASK NOW OR ASK IN THE CHAT!

WHAT ARE DATA?

How does R make sense of this?

- What “words” does R use?
- We are going to pick apart this data set, and give you a vocabulary so that you can talk to R
- How can you talk to R about these data in such a way that R understands you?
- **to be clear, your experiment data is in the csv file – you’ll put that spreadsheet into R….and then when you DO…

Data Frames

- Used for storing variables of equal length.
- In Excel, you would call this a “spreadsheet”.
- Many researchers might simply call this “data”.

Data Frames

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Data Frames

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Data Frames

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Data Frames

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Data Frames

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Data Frames

Used for storing data by combining variables of equal length

- `data.frame()`

Exercise

Create a data.frame named `workshop`.

- Variables:
  - `sub: "Subject##"`
  - `bip: T/F`
  - `cty: ##`
  - `pt: "dog" vs. "ant"`

OBJECTS

Object

- A basic concept in (statistical) programming is called an **object**.
- An object allows you to store a value or a thing in R.
- You use the object's name to easily access this value or thing.

Data Frames

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Data Frames

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Object

```
pi
[1] 3.141593
```

Referencing objects

If you want to use an object later on, you have to name it.

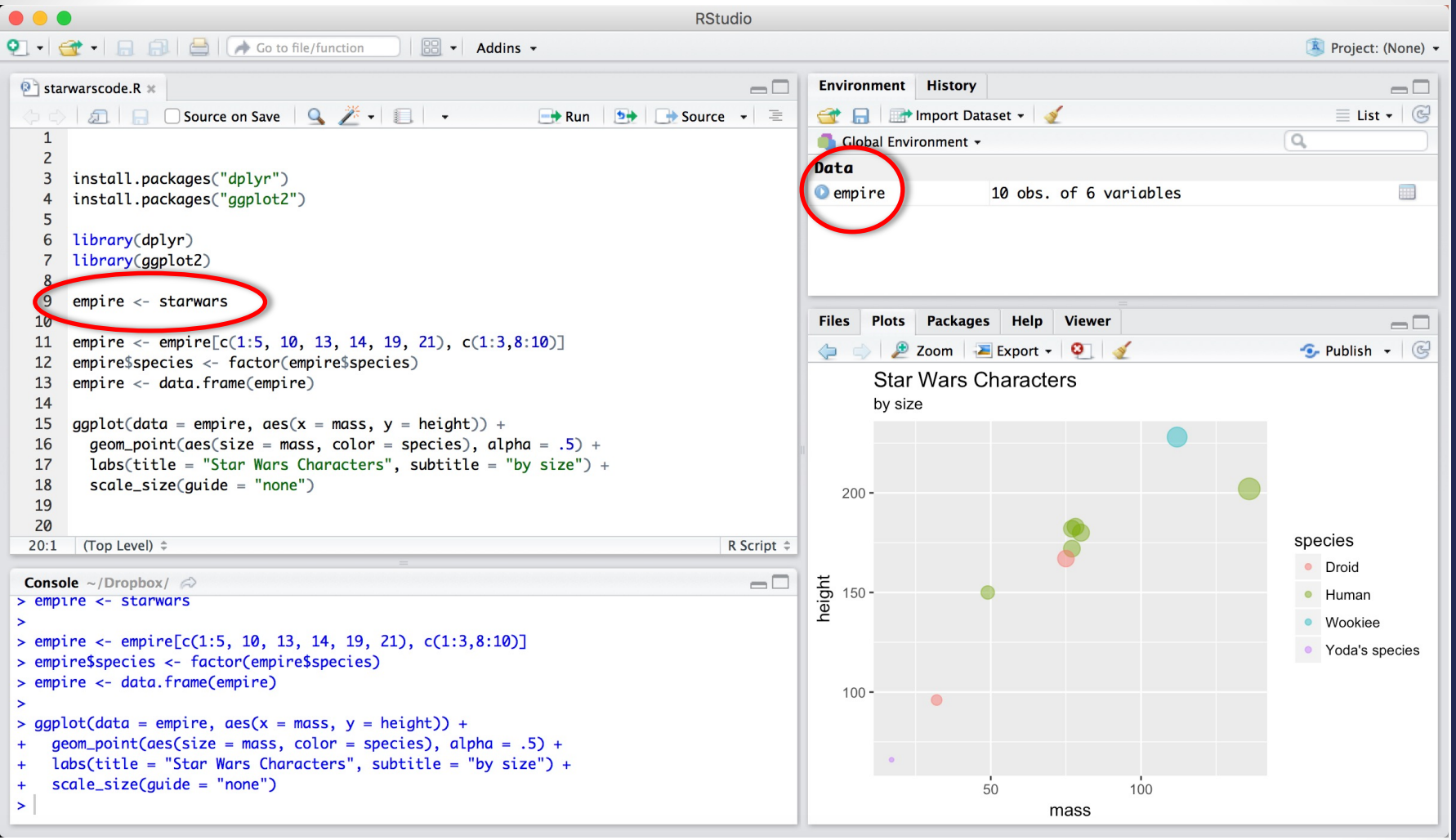
```
my_object = thingtoassign
```

```
my_object <- thingtoassign
```

```
Ex) pi <- 3.141593
```

WE RECOMMEND USING THE SECOND WAY!

(Will become important when we start using functions. Good habit to establish now!)



- **Objects** can be used later!
- You can see them in your **Global Environment.**

Exercise

Now let's try adding together two objects.

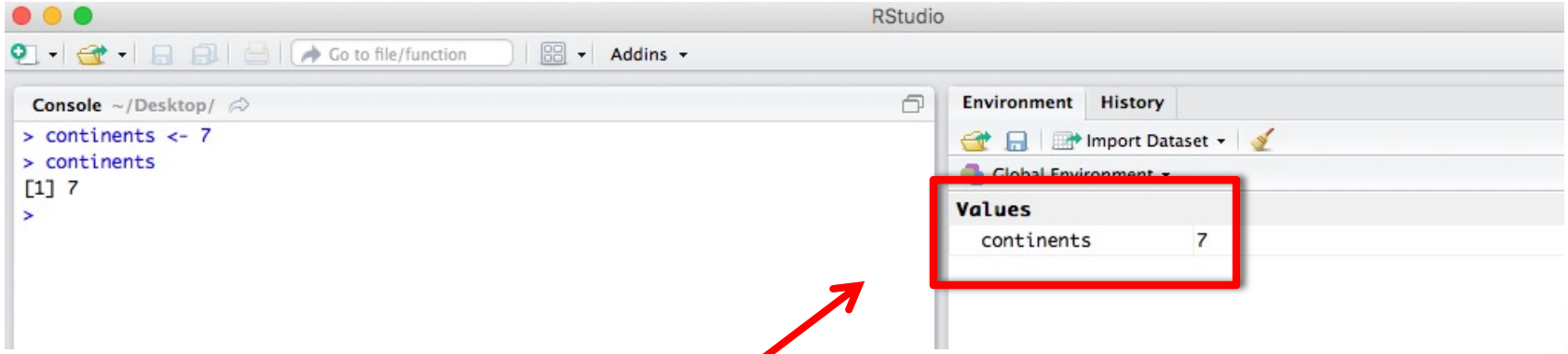
In your **script**:

- Assign the value of `5` to an object called `apples`.
- Assign the value of `2` to an object called `oranges`.
- Give the command `apples + oranges` and find out how much fruit you have!
  - **BONUS: where is your answer stored?**

Exercise

Try creating your own object!

1. In the console, assign the number `7` to `continents`
2. Then type `continents` in the console to print out the assigned value



Look!
Your environment has stored your object!

BONUS: What is the difference between typing in the console or in a script file?

Basic Object Classes

Numeric: Decimals (3.141593)

Integer: Natural numbers (0, 1, 2…)

Character: Text or string characters

- Always inside quotations.
- **Factors** (or categories)

Logical: Boolean values (True or False)

- No quotations.
- 2 possible values: **TRUE** or **FALSE**

Factors

Character objects

- Character strings represent distinct groups.
  - Control, Treatment
  - Dog, Ant

Basic Data Class

To check what data class your object is, you can type `class()` into the console.

```
class(cty)
[1] "numeric"
```

Exercise

Assign values to the different objects

- `lastname`
- `birthyear`
- `schlLevel` (undergrad, grad, postdoc, other)
- `pet` (do you own a pet?)

Check the classes of each of your objects.

Vectors

Group of objects is a vector

Vectors are ONE-DIMENSIONAL

Vectors

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Vectors

	name	height	mass	gender	homeworld	species
1	Luke Skywalker	172	77.0	male	Tatooine	Human
2	C-3PO	167	75.0	*NA*	Tatooine	Droid
3	R2-D2	96	32.0	*NA*	Naboo	Droid
4	Darth Vader	202	136.0	male	Tatooine	Human
5	Leia Organa	150	49.0	female	Alderaan	Human
6	Obi-Wan Kenobi	182	77.0	male	Stewjon	Human
7	Chewbacca	228	112.0	male	Kashyyyk	Wookiee
8	Han Solo	180	80.0	male	Corellia	Human
9	Yoda	66	17.0	male	*NA*	Yoda's species
10	Boba Fett	183	78.2	male	Kamino	Human

Indexing

Having a group of objects is great, but sometimes you only want one or a few of those objects.

How do we ACCESS our data?

Indexing a vector

To index a vector

- `[ ]`

For example, how many cities has the **3rd** person lived in?

- `cty[3]`

Indexing

You can select multiple objects within a vector

To select objects that are sequential (in a row):

- `cty[3:5]`
- You can think of `:` as “through”
  - `[3:5]` = “three ***through*** five”

To select objects that are not in a row:

- `cty[c(1,2,5)]`

Indexing

You can go crazy and combine these!

- `cty[c(1:3, 5)]`

Indexing

Data frames can be indexed just like vectors

EXCEPT: data frames have **2** dimensions

- Rows and columns

Indexing data frames

`data.frame[rows, columns]`

`Ex: empire[1:6,5]`

```
> empire
                name height  mass gender homeworld          species
1  Luke Skywalker    172  77.0   male  Tatooine            Human
2           C-3PO    167  75.0   <NA>  Tatooine            Droid
3           R2-D2     96  32.0   <NA>     Naboo            Droid
4     Darth Vader    202 136.0   male  Tatooine            Human
5     Leia Organa    150  49.0 female  Alderaan            Human
6  Obi-Wan Kenobi    182  77.0   male   Stewjon            Human
7       Chewbacca    228 112.0   male  Kashyyyk          Wookiee
8        Han Solo    180  80.0   male  Corellia            Human
9            Yoda     66  17.0   male      <NA> Yoda's species
10      Boba Fett    183  78.2   male    Kamino            Human
```

Indexing data frames

```
empire[1:6,]
```

```
**If you want all of something, leave it blank.
```

```
Console ~/Dropbox/
> empire[1:6,]
             name height mass gender homeworld species
1 Luke Skywalker    172   77   male  Tatooine   Human
2          C-3PO    167   75   <NA>  Tatooine   Droid
3          R2-D2     96   32   <NA>     Naboo   Droid
4    Darth Vader    202  136   male  Tatooine   Human
5    Leia Organa    150   49 female  Alderaan   Human
6 Obi-Wan Kenobi    182   77   male   Stewjon   Human
```

Finding Your Data

Most of the time, your data are stored in a data.frame…

…BUT you're not doing something with the entire data.frame.

In a data.frame, our **columns** have names—we can use these names instead of memorizing what the column number is!

You can access just one column at a time by using $

Finding Your Data

$ means you are accessing just one column within your data.frame

Console ~/Dropbox/

```
> empire
                name height  mass gender homeworld        species
1  Luke Skywalker    172  77.0   male  Tatooine          Human
2           C-3PO    167  75.0   <NA>  Tatooine          Droid
3           R2-D2     96  32.0   <NA>     Naboo          Droid
4     Darth Vader    202 136.0   male  Tatooine          Human
5     Leia Organa    150  49.0 female  Alderaan          Human
6  Obi-Wan Kenobi    182  77.0   male   Stewjon          Human
7       Chewbacca    228 112.0   male  Kashyyyk        Wookiee
8        Han Solo    180  80.0   male  Corellia          Human
9            Yoda     66  17.0   male      <NA> Yoda's species
10      Boba Fett    183  78.2   male    Kamino          Human
```

Console ~/Dropbox/

```
> empire$height
 [1] 172 167  96 202 150 182 228 180  66 183
```

Indexing (again)

Let's say you want to know how tall R2-D2 is…

Option 1: [row #, column #]

```
empire[3, 2]
```

Option 2: [row #, column name]

```
empire[3, "height"]
```

Option 3: data.frame$column[item#]

```
empire$height[3]
```

A NOTE ON ACCESSING DATA

Sometimes (*like now!*), objects can exist in two places in your environment:

- **Values**
  - You'll see **pt** here, too!
- **Data**
  - Click on the arrow next to **workshop** → you'll see **personType**

Once your data have a "home," always use their "address". We refer to this as "calling" a variable from your data frame.

Ex) If I wanted to use my "dog vs. ant person" variable, I could type:

`pt` – like yelling KENDRA (loudly) in the middle of a busy street to try and get a hold of my friend. Any random person named Kendra (and there are a LOT of them) could walk up.

`workshop$personType` – like knocking on my friend Kendra's front door if I want to talk to her.

ACTING ON VARIABLES

Acting on variables

In the previous section, we talked about objects and data sets.

Now let’s do something with them!

Actions

- Operators
- Functions

Operators

An **operator** is a simple calculation.

Operators

+	addition
-	subtraction
*	multiplication
/	division
^	taking powers

Order of Operations

Important note: Order of operations matters!

```
(8-4)/2
[1] 2
```

```
8-(4/2)
[1] 6
```

PEMDAS, anyone?

Example

1. Take 14, add 4, and using parentheses, multiply the whole thing by 18.

2. Take 14 and add the product of 4 and 18 (no parentheses).

Example

```
(14 + 4) * 18
[1] 324

14 + 4 * 18
[1] 86
```

Logical Operators

==	equality
!=	inequality
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Logical Operators

- Return a value of **TRUE** or **FALSE**

```
workshop$personType == “ant”
```

Example

- Test whether `cities` is greater than 2.

Logical Operators

Has anyone in our data.frame lived in more than 2 cities?

```
workshop$cities > 2
```

Has anyone in our data.frame lived in exactly 2 cities?

```
workshop$cities == 2
```

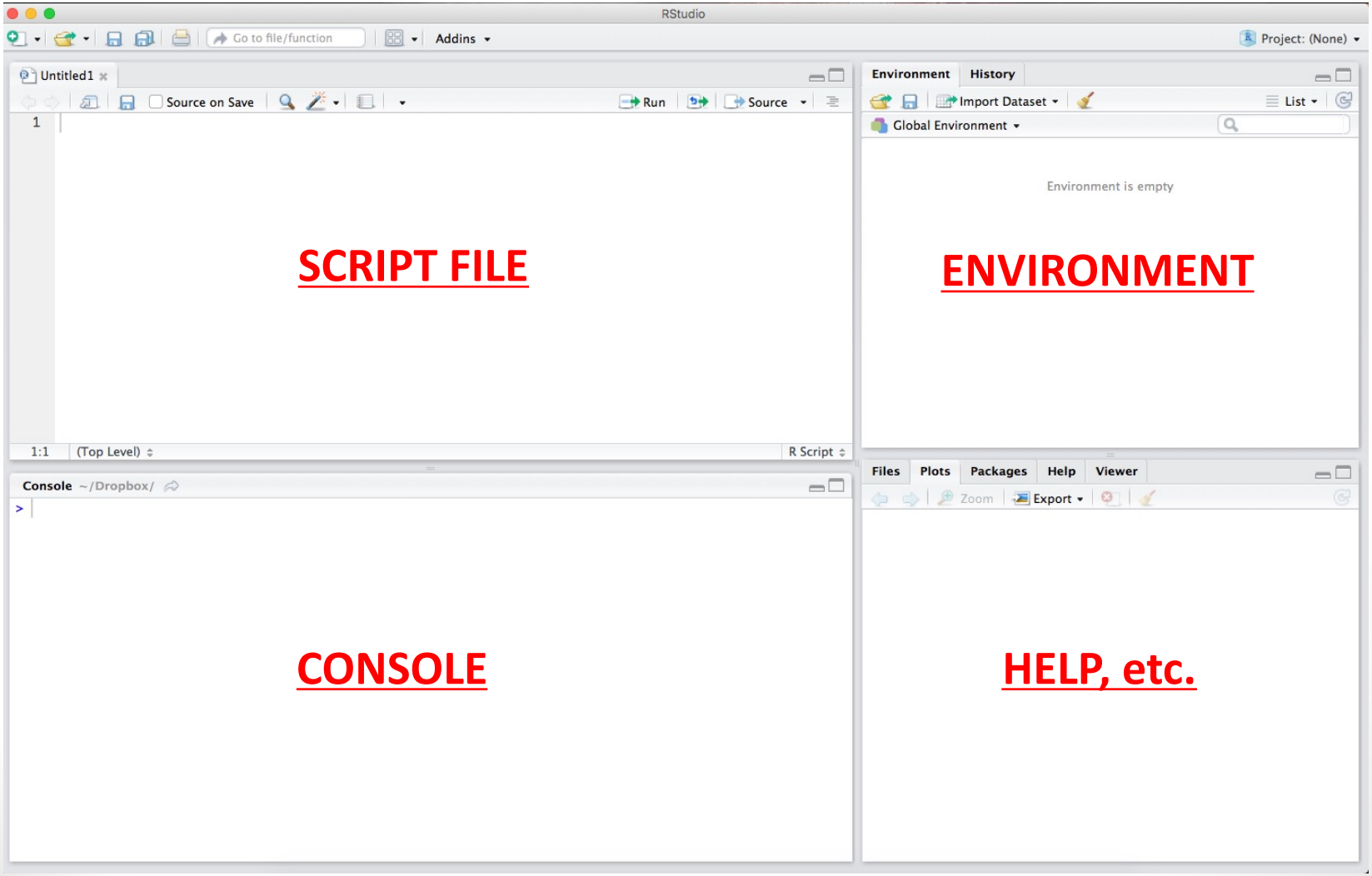
```
workshop$cities  = 2
```

→ WARNING: a SINGLE equals sign will change your data!

RECAP

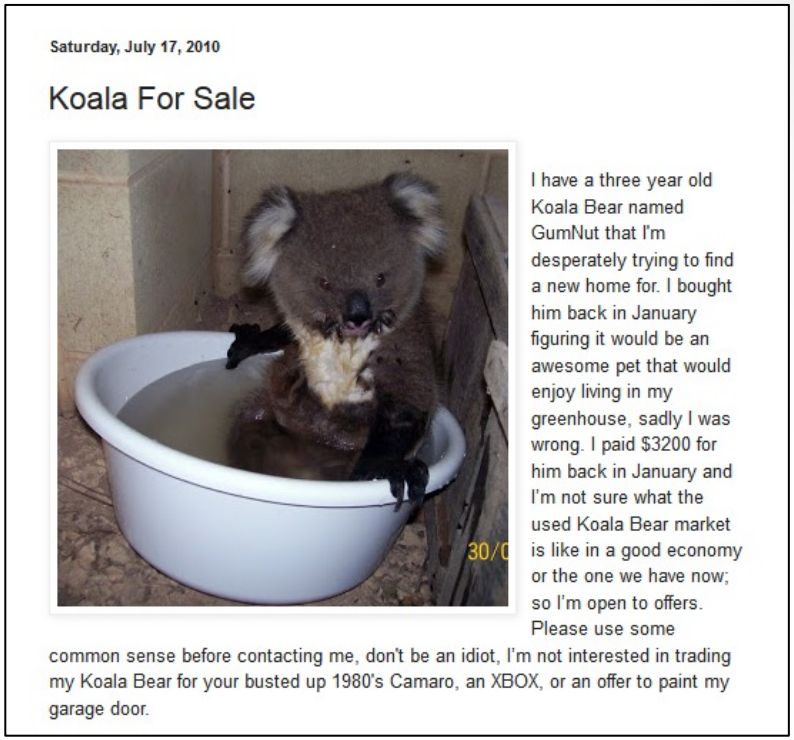
1) What is the difference between typing in the **console** vs. a **script file**?

- **Console** = immediate feedback; "yells" at you
- **Script** = need to tell it to run; can save for future use/editing



2) Consider the following piece of code:

```
bestanimal <- "GumNut"
```


Saturday, July 17, 2010

Koala For Sale

I have a three year old Koala Bear named GumNut that I'm desperately trying to find a new home for. I bought him back in January figuring it would be an awesome pet that would enjoy living in my greenhouse, sadly I was wrong. I paid $3200 for him back in January and I'm not sure what the used Koala Bear market is like in a good economy or the one we have now; so I'm open to offers. Please use some common sense before contacting me, don't be an idiot, I'm not interested in trading my Koala Bear for your busted up 1980's Camaro, an XBOX, or an offer to paint my garage door.

- What does this code do?
  - **Creates an object**
  - **Assigns a value to the object**
- What class is the object?
  - **Character**
- How can we verify this?
  - `class(bestanimal)`
- Characters look different from numeric, integer, and boolean values. What are these differences?
  - **Numeric, integer, and boolean values show up as `blue`**
  - **Numeric, integer, and boolean values don't need quotation marks**
- What happens if we type in `bestanimal` to our console?
  - **It will print "GumNut"**

3) How would you read this code?

`age >= 25`

- Age is greater than or equal to 25

`anxiety_score != 99`

- Anxiety score is not equal to 99

`data.frame[1:5,]`

- All columns for rows 1 through 5 of our data frame

Bonus Qs!

- If we ran any of the above, would we be able to access the result later without re-running that code?
  - No! Need to assign it to something.
- What kind of output would the first two pieces of code generate?
  - Logical operators return vectors of `TRUE` / `FALSE` values.

RECAP

Intro to R: Day 2

Thursday, June 11, 2020

How to run code from your .R file

Highlight the lines that you want to run, then…

- Click on **Run** button

OR

- Ctrl + Enter (Windows)
- Command + Enter (Mac)

3.) You are given the following vectors:

```
nom <- c("Snape", "Harry", "Ron", "Draco",
         "Luna", "Sprout")
house <- c("Slyth", "Gryff", "Gryff", "Slyth",
           "Raven", "Huffle")
role <- c("prof", "student", "student",
          "student", "student", "prof")
```

- Since all three vectors are the same length and correspond to the same thing, make a new object so that they're all in one place and stay together (hint, the new object should be 2d) Rename the first column from nom to Name.
- Using only your new 2d object, use a logical operator to find out who is in "Gryff"
- Using only your new 2d object, use a logical operator to find out who is not a professor

ACTING ON VARIABLES: FUNCTIONS

Functions

- Sometimes, you want to do more than add or multiply variables.

- To perform more complicated actions, use *functions.*
  - Functions are commands that describe, manipulate or analyze objects.

Functions have three parts

1. Function name
   - *Ex:* `log(10)`

     `[1] 2.302`
2. Arguments
   - *Ex:* `log(10)`

     `[1] 2.302`
3. Output
   - *Ex:* `log(10)`

     `[1] 2.302`

Each function has one and only one name.

Functions have three parts

1. Function name
   - *Ex:* `log(10)`
     `[1] 2.302`

2. Arguments
   - *Ex:* `log(10)`
     `[1] 2.302`

3. Output
   - *Ex:* `log(10)`
     `[1] 2.302`

One argument is always specified: the input. This is the object that the function acts on.

Other arguments control **how** the function acts. For example, do you want the natural log? Or log base 10?

Each function has defaults for its arguments. You should know what those are and how to change them.

Functions have three parts

1. Function name
   - *Ex:* `log(10)`
     `[1] 2.302`
2. Arguments
   - *Ex:* `log(10)`
     `[1] 2.302`
3. Output
   - *Ex:* `log(10)`
     `[1] 2.302`

Output can be a:
- number/integer
- a TRUE/FALSE statement
- a character value
- all of the above

Output can be a:
- single value
- vector
- data frame
- matrix
- list

You can store the output by assigning it to another object!

Mathematical functions

`sqrt()`	square root
`round()`	round a number
`log()`	logarithm
`exp()`	exponentiation
`abs()`	absolute value

Example

1. Find the `square root` of `85`.
2. Take the `log` of `100`.

Example

```
sqrt(85)
[1] 9.219544
log(100)
[1] 4.60517
```

Functions you'll use a lot!

`c()` – combine or concatenate

`class()` – check the class of an object

`factor()` – change a character vector into a factor vector (is there meaning? Ex: Treatment vs. Control, Male vs. Female, Session 1 vs. Session 2)

`table()` – really nice for getting quick counts (Ex: how many males and females are there?)

Exercise

Get the `mean()` of everyone's # of `cities` lived in our `workshop` data.frame.

What is the name of the function?

What is the input argument?

What is the output?

Multiple arguments

Most functions take more than one argument.

Separate arguments with commas.

```
round (x = 2.30467, digits = 3)
[1] 2.305
```

Number that needs to be rounded.

Multiple arguments

Most functions take more than one argument.

Separate arguments with commas.

```
round (x = 2.30467, digits = 3)
[1] 2.305
```

Number of digits to round to.

Arguments have Names

Most arguments in functions have names.

USE THE NAMES!!!

```
round (x = 2.30467, digits = 3)
[1] 2.305
```

Order of arguments

This is a bad idea!

Note: You don't have to name the argument.

- If you *do not* use the names:
  - Arguments MUST go in the right order.
  - You cannot skip arguments.
    - `round (2.30467, 3)`
      `[1] 2.305`
    - `round (3, 2.30467)`
      `[1] 3` INCORRECT!
- If you name the arguments, you can put them in any order that you want and you can skip some.
    - `round (x = 2.30467, digits = 3)`
      `[1] 2.305`
    - `round (digits = 3, x = 2.30467)`
      `[1] 2.305`

Exercise

1. Use the `seq()` function to list numbers `0` to `100`.

   Arguments:

   - `from` = starting value of sequence
   - `to` = end value of sequence

2. Use the `seq()` function to list numbers `0` to `100`, by intervals of `10`.

   Arguments:

   - `from` = starting value of sequence
   - `to` = end value of sequence
   - `by` = increment of the sequence

Exercise

1. Use the `seq()` function to list numbers `0` to `100`

   - `seq(from = 0, to = 100)`

2. Use the `seq()` function to list numbers `0` to `100`, by intervals of `10`

   - `seq(from = 0, to = 100, by = 10)`

RStudio

starwarscode.R

```
1
2
3  install.packages("dplyr")
4  install.packages("ggplot2")
5
6  library(dplyr)
7  library(ggplot2)
8
9  empire <- starwars
10
11 empire <- empire[c(1:5, 10, 13, 14, 19, 21), c(1:3,8:10)]
12 empire$species <- factor(empire$species)
13 empire <- data.frame(empire)
14
15 ggplot(data = empire, aes(x = mass, y = height)) +
16   geom_point(aes(size = mass, color = species), alpha = .5) +
17   labs(title = "Star Wars Characters", subtitle = "by size") +
18   scale_size(guide = "none")
19
20
```

20:1 (Top Level) R Script

Console ~/Dropbox/

```
> empire <- starwars
>
> empire <- empire[c(1:5, 10, 13, 14, 19, 21), c(1:3,8:10)]
> empire$species <- factor(empire$species)
> empire <- data.frame(empire)
>
> ggplot(data = empire, aes(x = mass, y = height)) +
+   geom_point(aes(size = mass, color = species), alpha = .5) +
+   labs(title = "Star Wars Characters", subtitle = "by size") +
+   scale_size(guide = "none")
>
```

Environment History

Import Dataset List

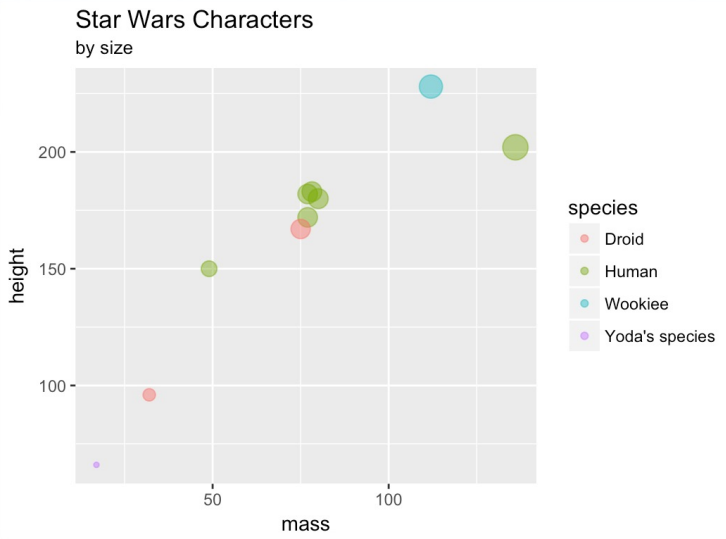
Global Environment

Data

empire	10 obs. of 6 variables

Files Plots Packages Help Viewer

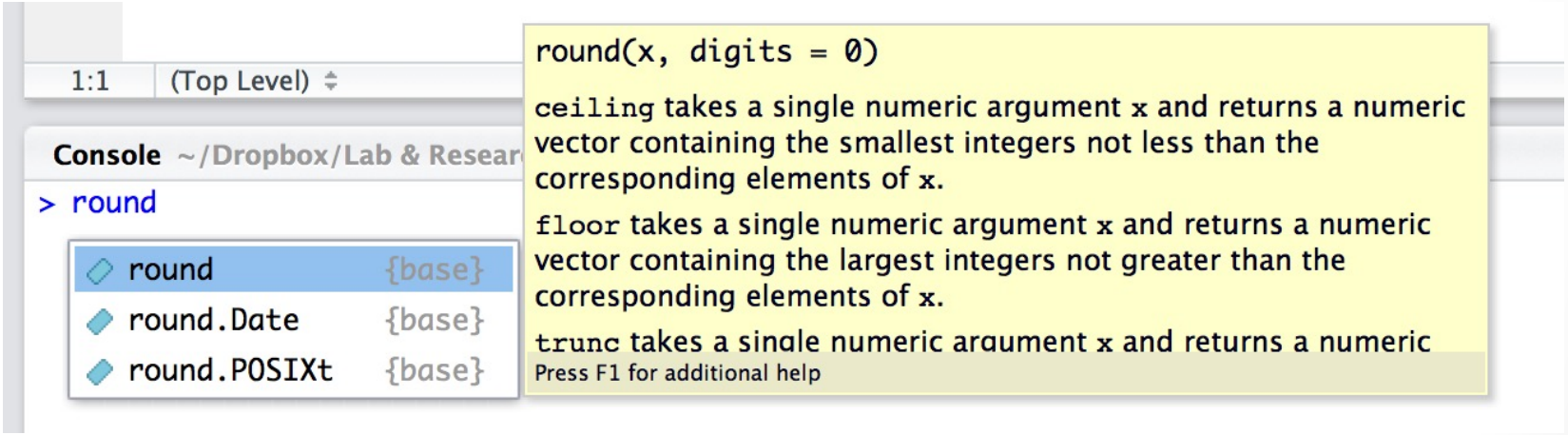
Zoom Export Publish



Great, but how do I know what the arguments are for a function?

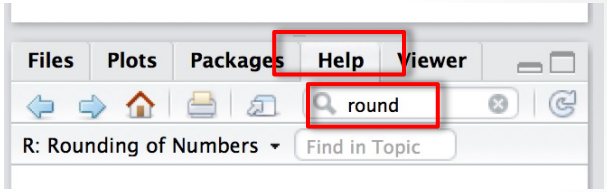
Two ways:

1) In RStudio, press the **tab** key to see names of arguments and descriptions.

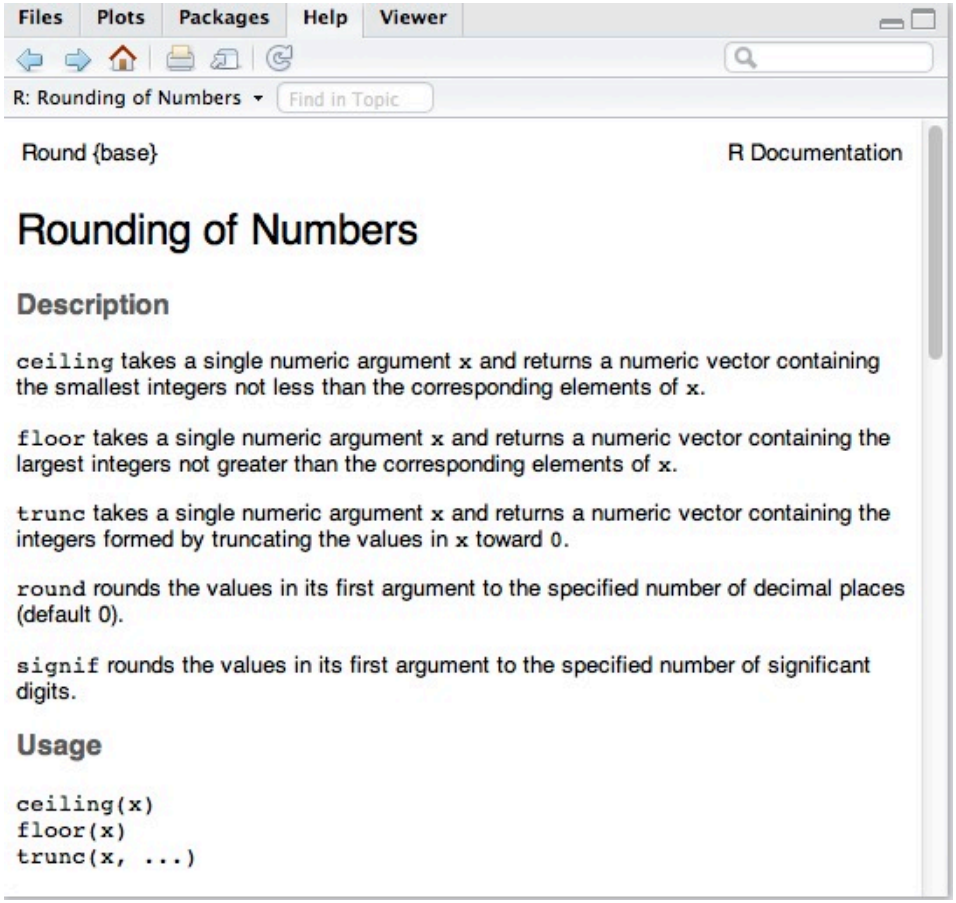


2) Look in the R documentation!

- Go to Help tab



- Or just type `?round` into the console



You try!

Type code to look up the R documentation for the correlation function, `cor`

```
?cor
```

R: Correlation, Variance and Covariance (Matrices)

cor {stats} R Documentation

Correlation, Variance and Covariance (Matrices)

Description

`var`, `cov` and `cor` compute the variance of `x` and the covariance or correlation of `x` and `y` if these are vectors. If `x` and `y` are matrices then the covariances (or correlations) between the columns of `x` and the columns of `y` are computed.

`cov2cor` scales a covariance matrix into the corresponding correlation matrix *efficiently*.

Usage

```
var(x, y = NULL, na.rm = FALSE, use)

cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cov2cor(V)
```

Arguments

`x`	a numeric vector, matrix or data frame.
`y`	`NULL` (default) or a vector, matrix or data frame with compatible dimensions to `x`. The default is equivalent to `y = x` (but more efficient).
`na.rm`	logical. Should missing values be removed?
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"`.
`method`	a character string indicating which correlation coefficient (or covariance) is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`, can be abbreviated.
`V`	symmetric numeric matrix, usually positive definite such as a covariance matrix.

Details

For `cov` and `cor` one must *either* give a matrix or data frame for `x` *or* give both `x` and `y`.

The inputs must be numeric (as determined by `is.numeric`: logical values are also allowed for historical compatibility): the `"kendall"` and `"spearman"` methods make sense for ordered inputs but `xtfrm` can be used to find a suitable prior transformation to numbers.

`var` is just another interface to `cov`, where `na.rm` is used to determine the default for `use` when that is unspecified. If `na.rm` is `TRUE` then the complete observations (rows) are used (`use = "na.or.complete"`) to compute the variance. Otherwise, by default `use = "everything"`.

If `use` is `"everything"`, `NA`s will propagate conceptually, i.e., a resulting value will be `NA` whenever one of its contributing observations is `NA`.

cor {stats}

Correlation, Variance and Covariance (Matrices)

Description

`var`, `cov` and `cor` compute the variance of `x` and the covariance or correlation of `x` and `y` if these are vectors. If `x` and `y` are matrices then the covariances (or correlations) between the columns of `x` and the columns of `y` are computed.

`cov2cor` scales a covariance matrix into the corresponding correlation matrix *efficiently*.

Usage

```
var(x, y = NULL, na.rm = FALSE, use)

cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cov2cor(V)
```

Arguments

`x`	a numeric vector, matrix or data frame.
`y`	`NULL` (default) or a vector, matrix or data frame with compatible dimensions to `x`. The default is equivalent to `y = x` (but more efficient).
`na.rm`	logical. Should missing values be removed?
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"`.
`method`	a character string indicating which correlation coefficient (or covariance) is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`, can be abbreviated.
`V`	symmetric numeric matrix, usually positive definite such as a covariance matrix.

Details

For `cov` and `cor` one must *either* give a matrix or data frame for `x` *or* give both `x` and `y`.

The inputs must be numeric (as determined by `is.numeric`: logical values are also allowed for historical compatibility): the `"kendall"` and `"spearman"` methods make sense for ordered inputs but `xtfrm` can be used to find a suitable prior transformation to numbers.

Value

For `r <- cor(*, use = "all.obs")`, it is now guaranteed that `all(r <= 1)`.

Examples

```
var(1:10)  # 9.166667

var(1:5, 1:5) # 2.5

## Two simple vectors
cor(1:10, 2:11) # == 1

## Correlation Matrix of Multivariate sample:
(Cl <- cor(longley))
## Graphical Correlation Matrix:
symnum(Cl) # highly correlated
```

Exercise

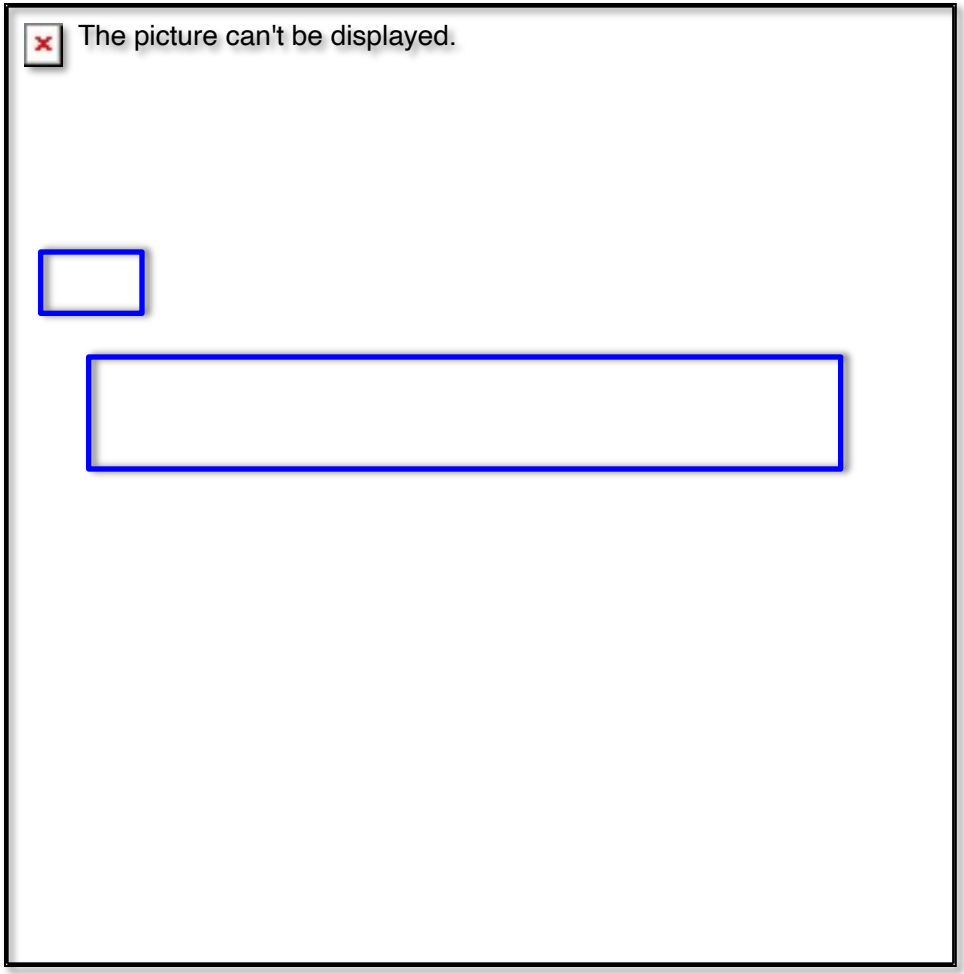
1. Look up documentation for `scale` and `plot`.
2. Using the `height` variable from our `empire` data.frame, make a new variable called `height_z`, using `scale.`
3. Do the same thing for `mass.`
4. Combine `height_z` vector and `mass_z` vector into a new data.frame called `empire_z`.
5. Make a scatter plot of standardized height (*hint: y-axis*) by standardized mass, using the `plot` function.
6. Add a title to your plot.
7. Add labels to the x and y axes.

What happens if you add `type = "l"` ?

What is the default for type?


The picture can't be displayed.

Back to the Documentation!



PACKAGES

Packages

R automatically loads some basic functions

- Generally useful!
- But sometimes, maybe you want something better.

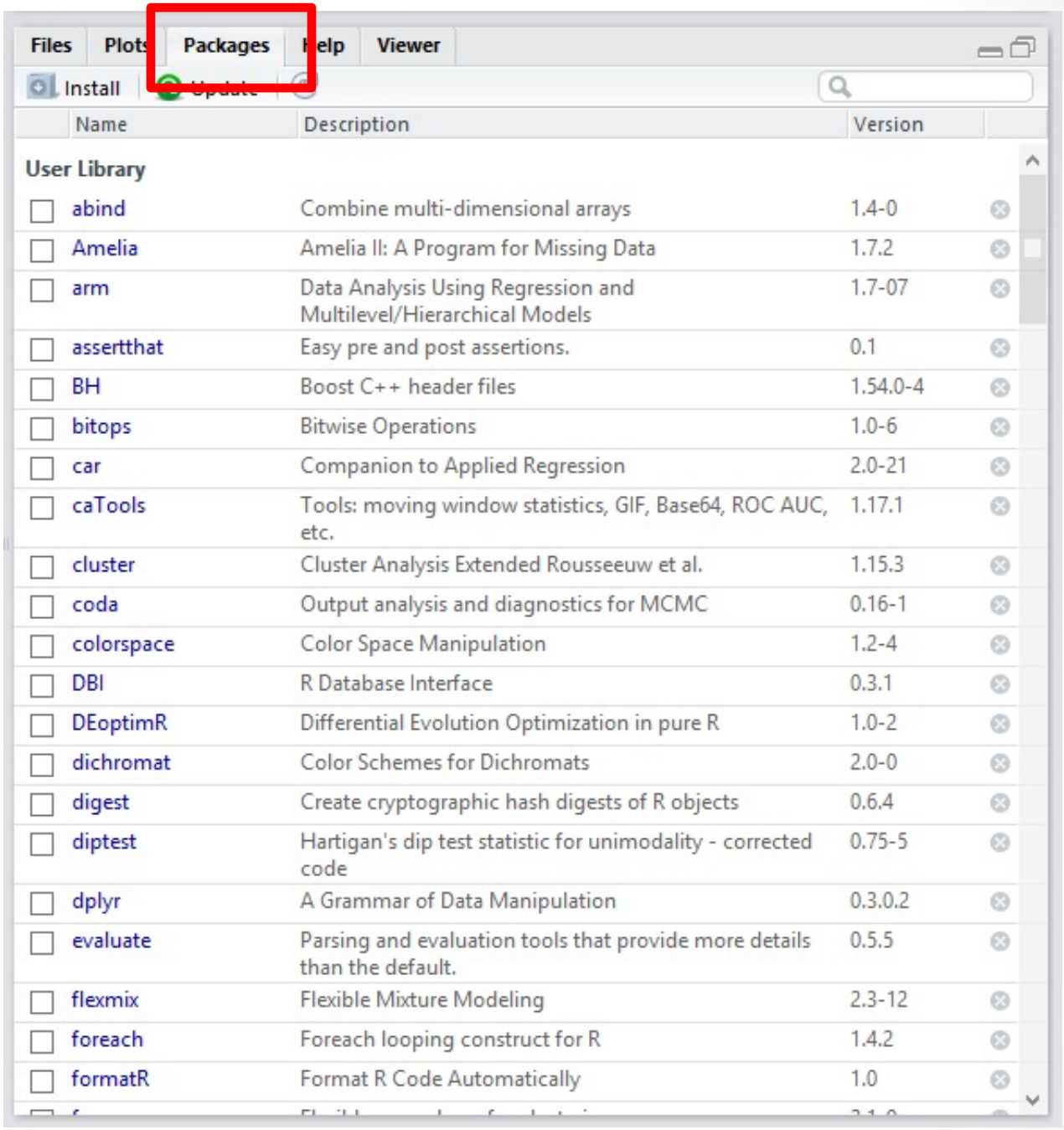
Packages

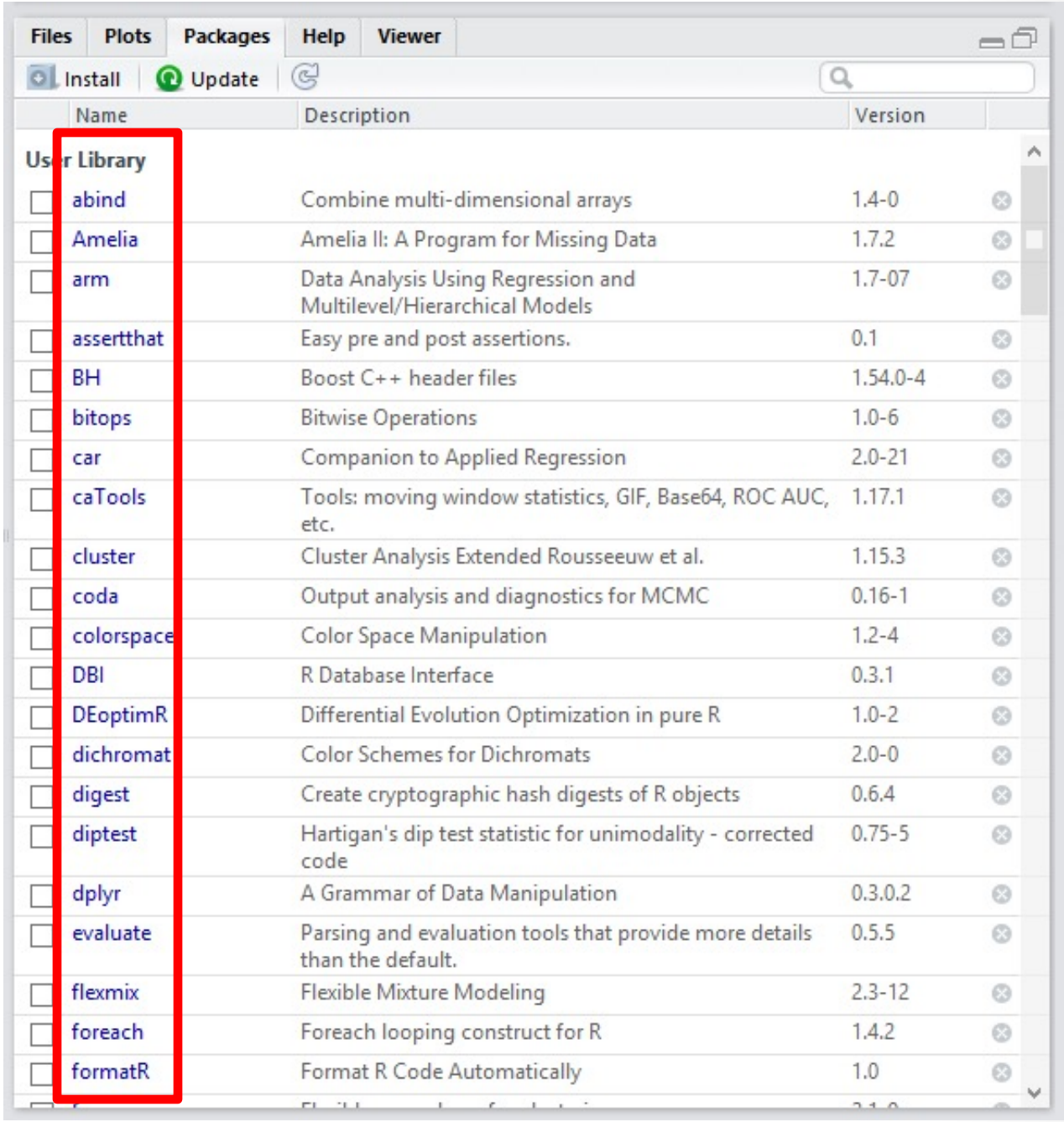
What is a package?

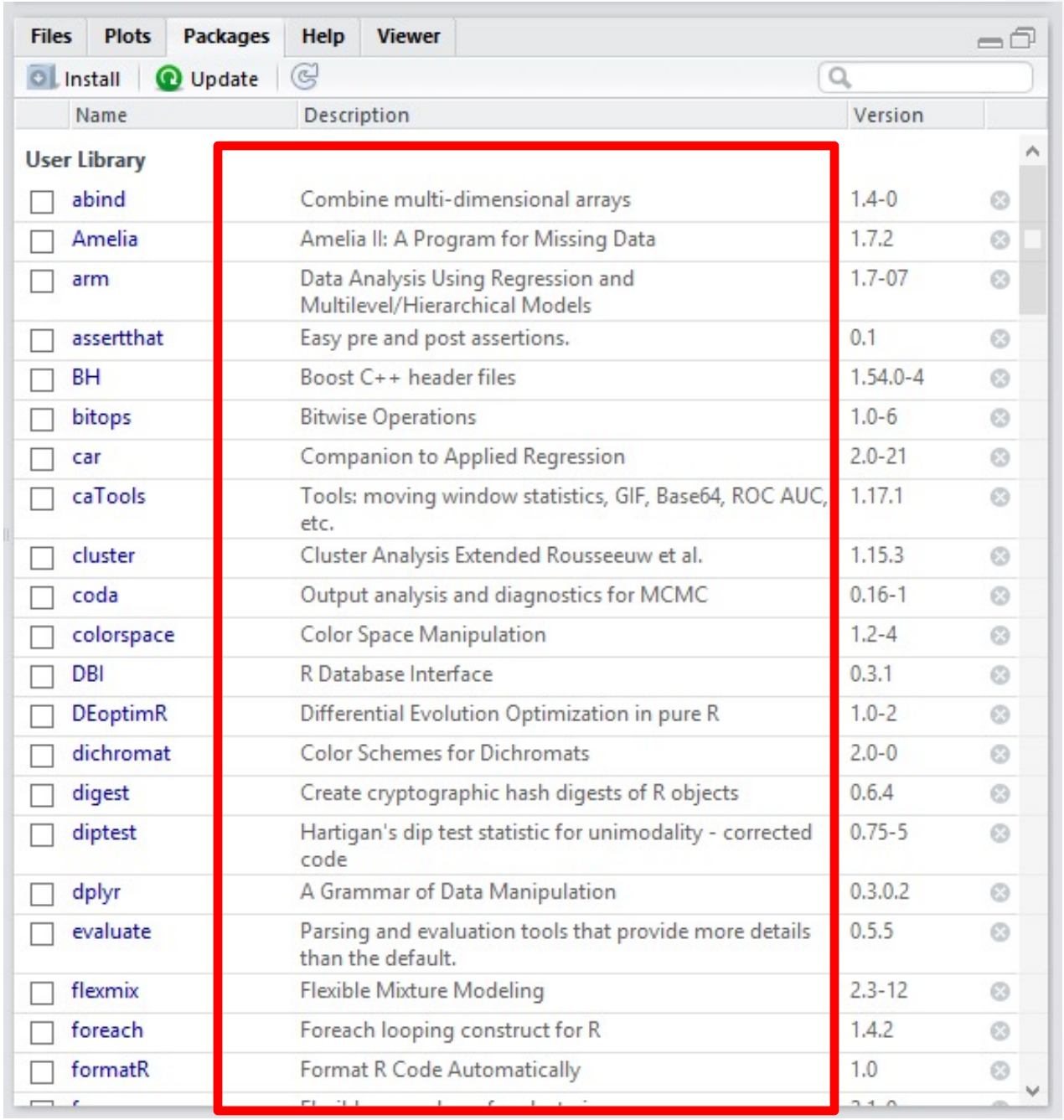
- A collection of functions and datasets.
- Open source (free!)

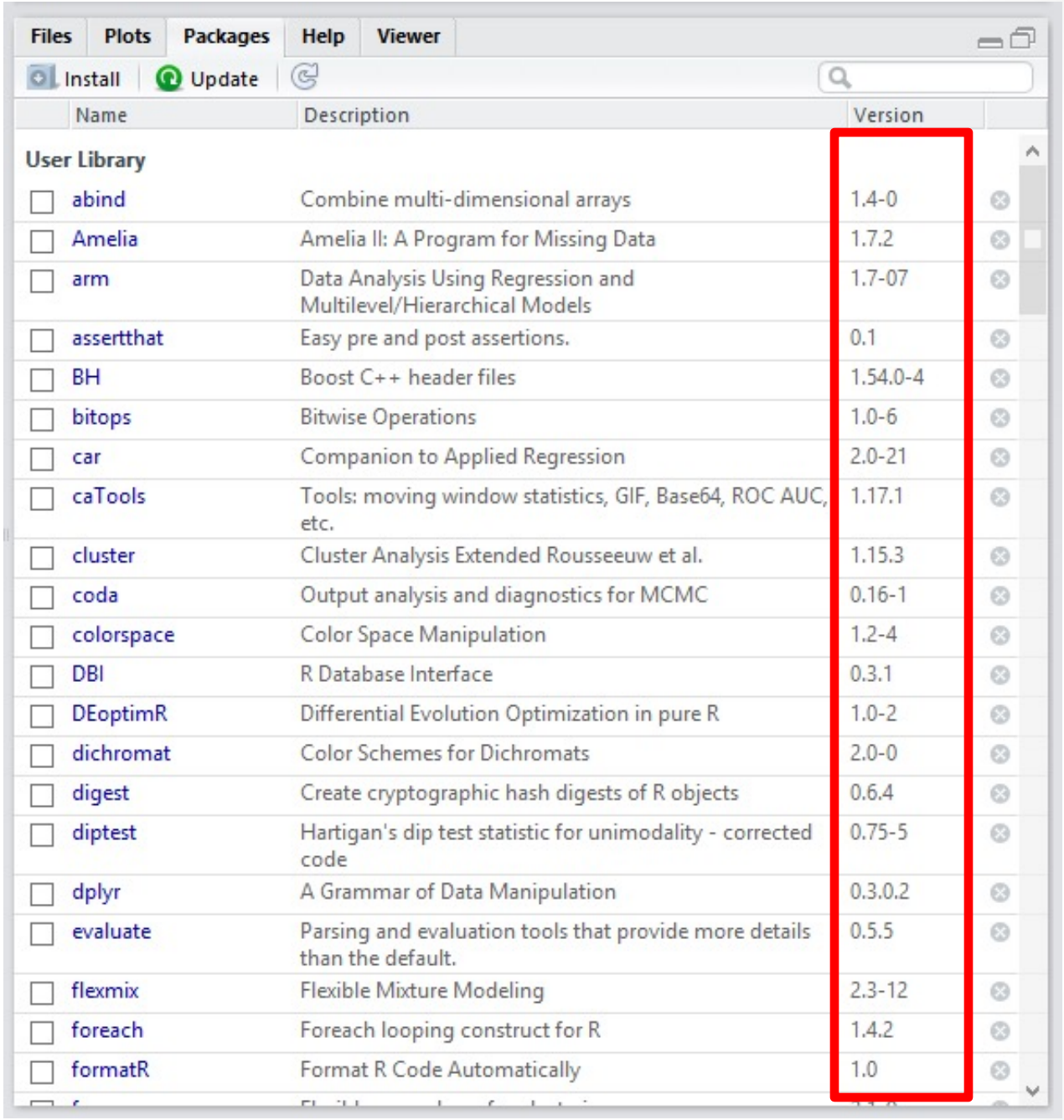
Packages are the reason R is so powerful.

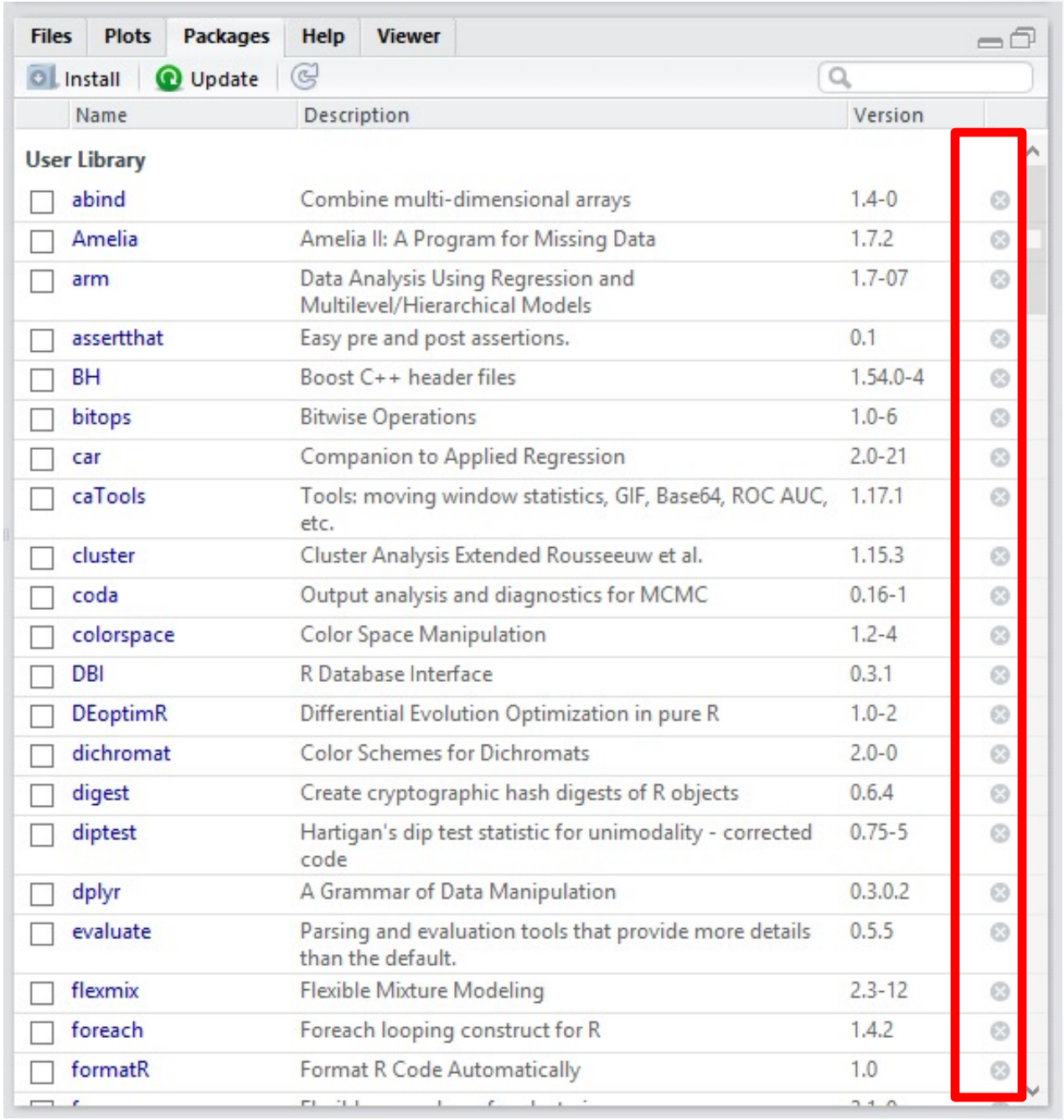
- And why it will never be out-of-date.

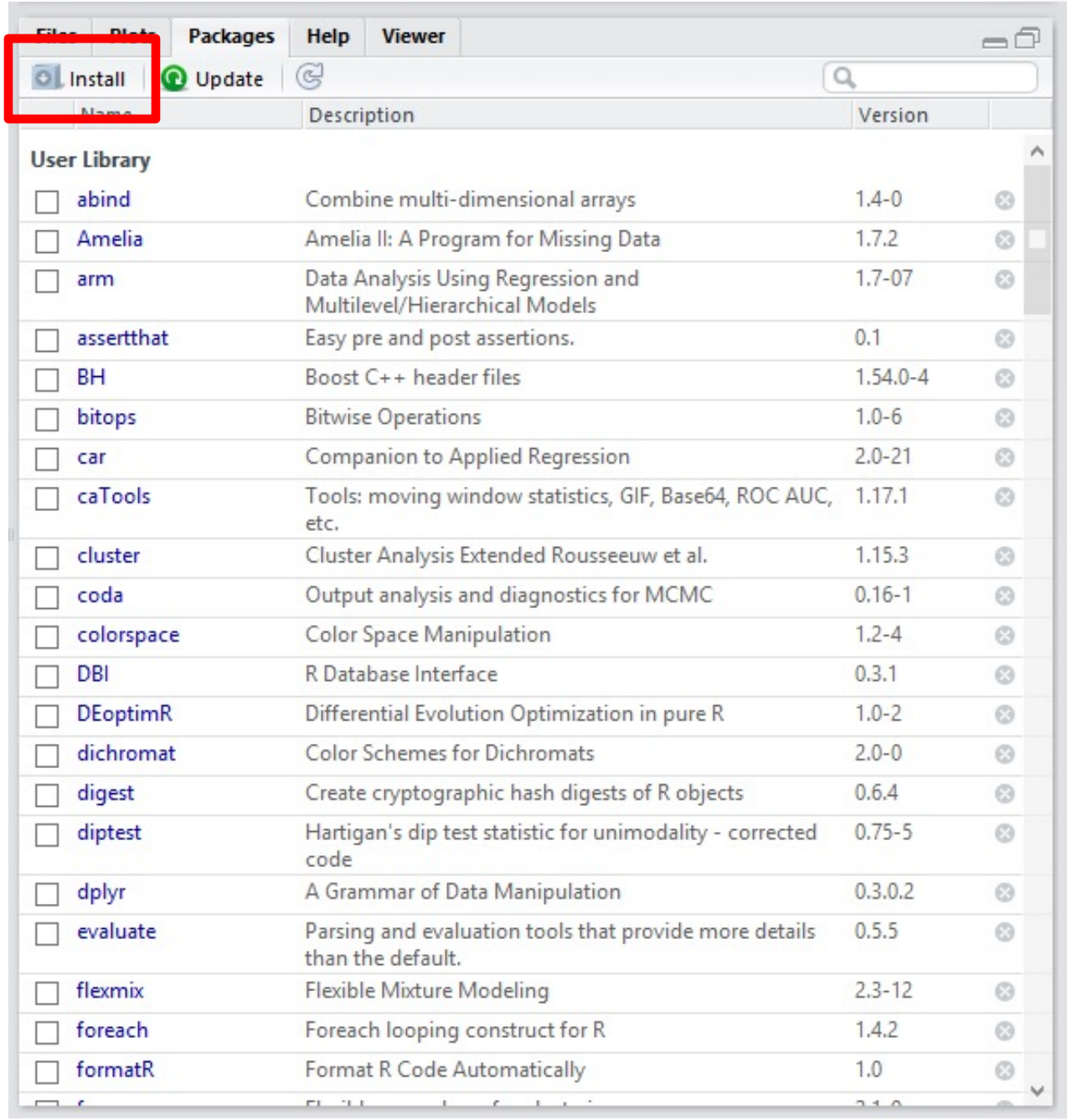

Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
abind
Combine multi-dimensional arrays
1.4-0
Amelia
Amelia II: A Program for Missing Data
1.7.2
arm
Data Analysis Using Regression and Multilevel/Hierarchical Models
1.7-07
assertthat
Easy pre and post assertions.
0.1
BH
Boost C++ header files
1.54.0-4
bitops
Bitwise Operations
1.0-6
car
Companion to Applied Regression
2.0-21
caTools
Tools: moving window statistics, GIF, Base64, ROC AUC, etc.
1.17.1
cluster
Cluster Analysis Extended Rousseeuw et al.
1.15.3
coda
Output analysis and diagnostics for MCMC
0.16-1
colorspace
Color Space Manipulation
1.2-4
DBI
R Database Interface
0.3.1
DEoptimR
Differential Evolution Optimization in pure R
1.0-2
dichromat
Color Schemes for Dichromats
2.0-0
digest
Create cryptographic hash digests of R objects
0.6.4
diptest
Hartigan's dip test statistic for unimodality - corrected code
0.75-5
dplyr
A Grammar of Data Manipulation
0.3.0.2
evaluate
Parsing and evaluation tools that provide more details than the default.
0.5.5
flexmix
Flexible Mixture Modeling
2.3-12
foreach
Foreach looping construct for R
1.4.2
formatR
Format R Code Automatically
1.0

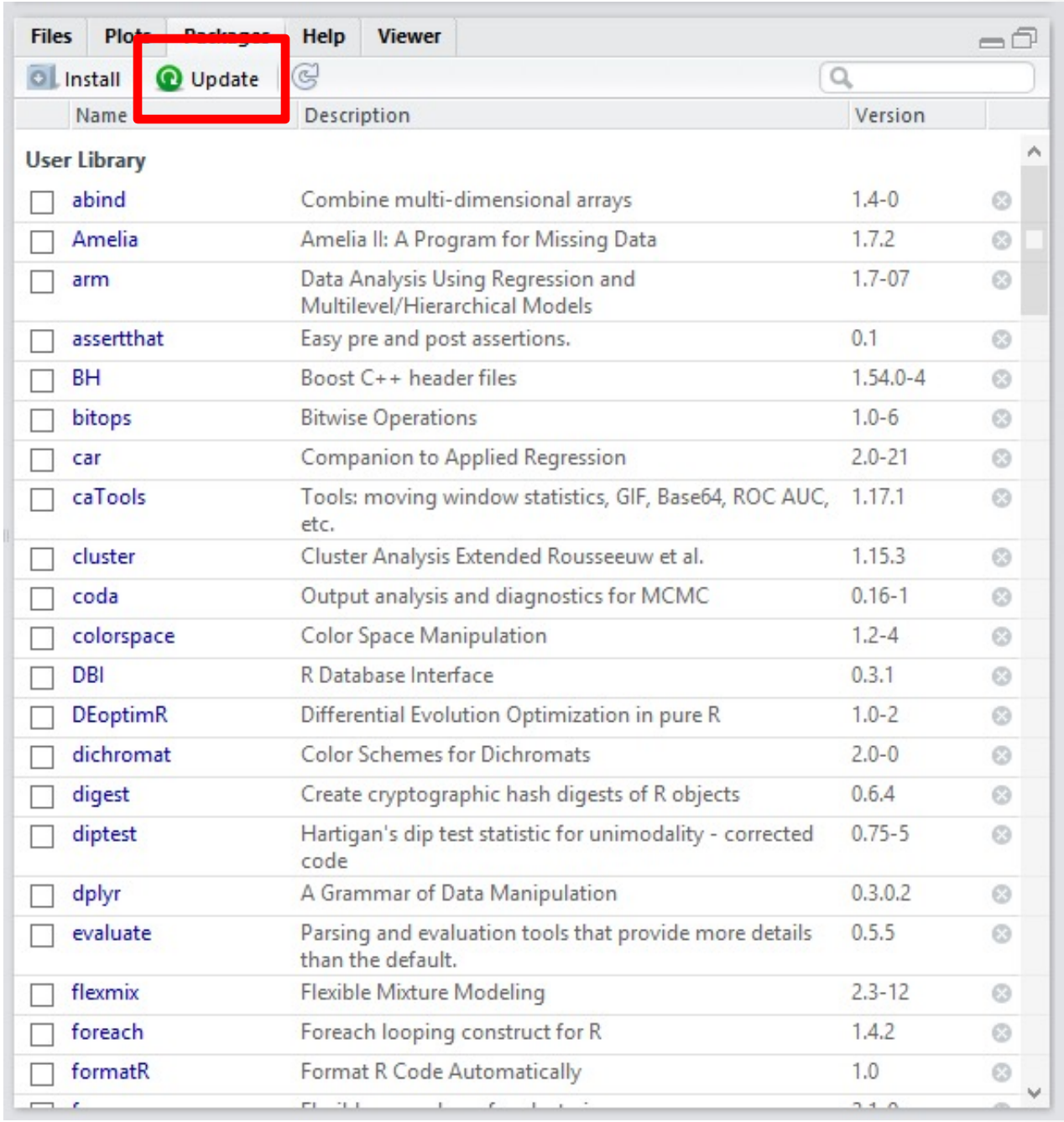

Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
abind
Combine multi-dimensional arrays
1.4-0
Amelia
Amelia II: A Program for Missing Data
1.7.2
arm
Data Analysis Using Regression and Multilevel/Hierarchical Models
1.7-07
assertthat
Easy pre and post assertions.
0.1
BH
Boost C++ header files
1.54.0-4
bitops
Bitwise Operations
1.0-6
car
Companion to Applied Regression
2.0-21
caTools
Tools: moving window statistics, GIF, Base64, ROC AUC, etc.
1.17.1
cluster
Cluster Analysis Extended Rousseeuw et al.
1.15.3
coda
Output analysis and diagnostics for MCMC
0.16-1
colorspace
Color Space Manipulation
1.2-4
DBI
R Database Interface
0.3.1
DEoptimR
Differential Evolution Optimization in pure R
1.0-2
dichromat
Color Schemes for Dichromats
2.0-0
digest
Create cryptographic hash digests of R objects
0.6.4
diptest
Hartigan's dip test statistic for unimodality - corrected code
0.75-5
dplyr
A Grammar of Data Manipulation
0.3.0.2
evaluate
Parsing and evaluation tools that provide more details than the default.
0.5.5
flexmix
Flexible Mixture Modeling
2.3-12
foreach
Foreach looping construct for R
1.4.2
formatR
Format R Code Automatically
1.0


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
abind
Combine multi-dimensional arrays
1.4-0
Amelia
Amelia II: A Program for Missing Data
1.7.2
arm
Data Analysis Using Regression and Multilevel/Hierarchical Models
1.7-07
assertthat
Easy pre and post assertions.
0.1
BH
Boost C++ header files
1.54.0-4
bitops
Bitwise Operations
1.0-6
car
Companion to Applied Regression
2.0-21
caTools
Tools: moving window statistics, GIF, Base64, ROC AUC, etc.
1.17.1
cluster
Cluster Analysis Extended Rousseeuw et al.
1.15.3
coda
Output analysis and diagnostics for MCMC
0.16-1
colorspace
Color Space Manipulation
1.2-4
DBI
R Database Interface
0.3.1
DEoptimR
Differential Evolution Optimization in pure R
1.0-2
dichromat
Color Schemes for Dichromats
2.0-0
digest
Create cryptographic hash digests of R objects
0.6.4
diptest
Hartigan's dip test statistic for unimodality - corrected code
0.75-5
dplyr
A Grammar of Data Manipulation
0.3.0.2
evaluate
Parsing and evaluation tools that provide more details than the default.
0.5.5
flexmix
Flexible Mixture Modeling
2.3-12
foreach
Foreach looping construct for R
1.4.2
formatR
Format R Code Automatically
1.0


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
abind
Combine multi-dimensional arrays
1.4-0
Amelia
Amelia II: A Program for Missing Data
1.7.2
arm
Data Analysis Using Regression and Multilevel/Hierarchical Models
1.7-07
assertthat
Easy pre and post assertions.
0.1
BH
Boost C++ header files
1.54.0-4
bitops
Bitwise Operations
1.0-6
car
Companion to Applied Regression
2.0-21
caTools
Tools: moving window statistics, GIF, Base64, ROC AUC, etc.
1.17.1
cluster
Cluster Analysis Extended Rousseeuw et al.
1.15.3
coda
Output analysis and diagnostics for MCMC
0.16-1
colorspace
Color Space Manipulation
1.2-4
DBI
R Database Interface
0.3.1
DEoptimR
Differential Evolution Optimization in pure R
1.0-2
dichromat
Color Schemes for Dichromats
2.0-0
digest
Create cryptographic hash digests of R objects
0.6.4
diptest
Hartigan's dip test statistic for unimodality - corrected code
0.75-5
dplyr
A Grammar of Data Manipulation
0.3.0.2
evaluate
Parsing and evaluation tools that provide more details than the default.
0.5.5
flexmix
Flexible Mixture Modeling
2.3-12
foreach
Foreach looping construct for R
1.4.2
formatR
Format R Code Automatically
1.0


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
abind
Combine multi-dimensional arrays
1.4-0
Amelia
Amelia II: A Program for Missing Data
1.7.2
arm
Data Analysis Using Regression and Multilevel/Hierarchical Models
1.7-07
assertthat
Easy pre and post assertions.
0.1
BH
Boost C++ header files
1.54.0-4
bitops
Bitwise Operations
1.0-6
car
Companion to Applied Regression
2.0-21
caTools
Tools: moving window statistics, GIF, Base64, ROC AUC, etc.
1.17.1
cluster
Cluster Analysis Extended Rousseeuw et al.
1.15.3
coda
Output analysis and diagnostics for MCMC
0.16-1
colorspace
Color Space Manipulation
1.2-4
DBI
R Database Interface
0.3.1
DEoptimR
Differential Evolution Optimization in pure R
1.0-2
dichromat
Color Schemes for Dichromats
2.0-0
digest
Create cryptographic hash digests of R objects
0.6.4
diptest
Hartigan's dip test statistic for unimodality - corrected code
0.75-5
dplyr
A Grammar of Data Manipulation
0.3.0.2
evaluate
Parsing and evaluation tools that provide more details than the default.
0.5.5
flexmix
Flexible Mixture Modeling
2.3-12
foreach
Foreach looping construct for R
1.4.2
formatR
Format R Code Automatically
1.0


Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
abind
Combine multi-dimensional arrays
1.4-0
Amelia
Amelia II: A Program for Missing Data
1.7.2
arm
Data Analysis Using Regression and Multilevel/Hierarchical Models
1.7-07
assertthat
Easy pre and post assertions.
0.1
BH
Boost C++ header files
1.54.0-4
bitops
Bitwise Operations
1.0-6
car
Companion to Applied Regression
2.0-21
caTools
Tools: moving window statistics, GIF, Base64, ROC AUC, etc.
1.17.1
cluster
Cluster Analysis Extended Rousseeuw et al.
1.15.3
coda
Output analysis and diagnostics for MCMC
0.16-1
colorspace
Color Space Manipulation
1.2-4
DBI
R Database Interface
0.3.1
DEoptimR
Differential Evolution Optimization in pure R
1.0-2
dichromat
Color Schemes for Dichromats
2.0-0
digest
Create cryptographic hash digests of R objects
0.6.4
diptest
Hartigan's dip test statistic for unimodality - corrected code
0.75-5
dplyr
A Grammar of Data Manipulation
0.3.0.2
evaluate
Parsing and evaluation tools that provide more details than the default.
0.5.5
flexmix
Flexible Mixture Modeling
2.3-12
foreach
Foreach looping construct for R
1.4.2
formatR
Format R Code Automatically
1.0


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
User Library
abind
Combine multi-dimensional arrays
1.4-0
Amelia
Amelia II: A Program for Missing Data
1.7.2
arm
Data Analysis Using Regression and Multilevel/Hierarchical Models
1.7-07
assertthat
Easy pre and post assertions.
0.1
BH
Boost C++ header files
1.54.0-4
bitops
Bitwise Operations
1.0-6
car
Companion to Applied Regression
2.0-21
caTools
Tools: moving window statistics, GIF, Base64, ROC AUC, etc.
1.17.1
cluster
Cluster Analysis Extended Rousseeuw et al.
1.15.3
coda
Output analysis and diagnostics for MCMC
0.16-1
colorspace
Color Space Manipulation
1.2-4
DBI
R Database Interface
0.3.1
DEoptimR
Differential Evolution Optimization in pure R
1.0-2
dichromat
Color Schemes for Dichromats
2.0-0
digest
Create cryptographic hash digests of R objects
0.6.4
diptest
Hartigan's dip test statistic for unimodality - corrected code
0.75-5
dplyr
A Grammar of Data Manipulation
0.3.0.2
evaluate
Parsing and evaluation tools that provide more details than the default.
0.5.5
flexmix
Flexible Mixture Modeling
2.3-12
foreach
Foreach looping construct for R
1.4.2
formatR
Format R Code Automatically
1.0

How do I get packages?

Packages can be downloaded from the CRAN (**C**omprehensive **R** **A**rchive **N**etwork).

You do this from inside RStudio!

- Need to be connected to the internet

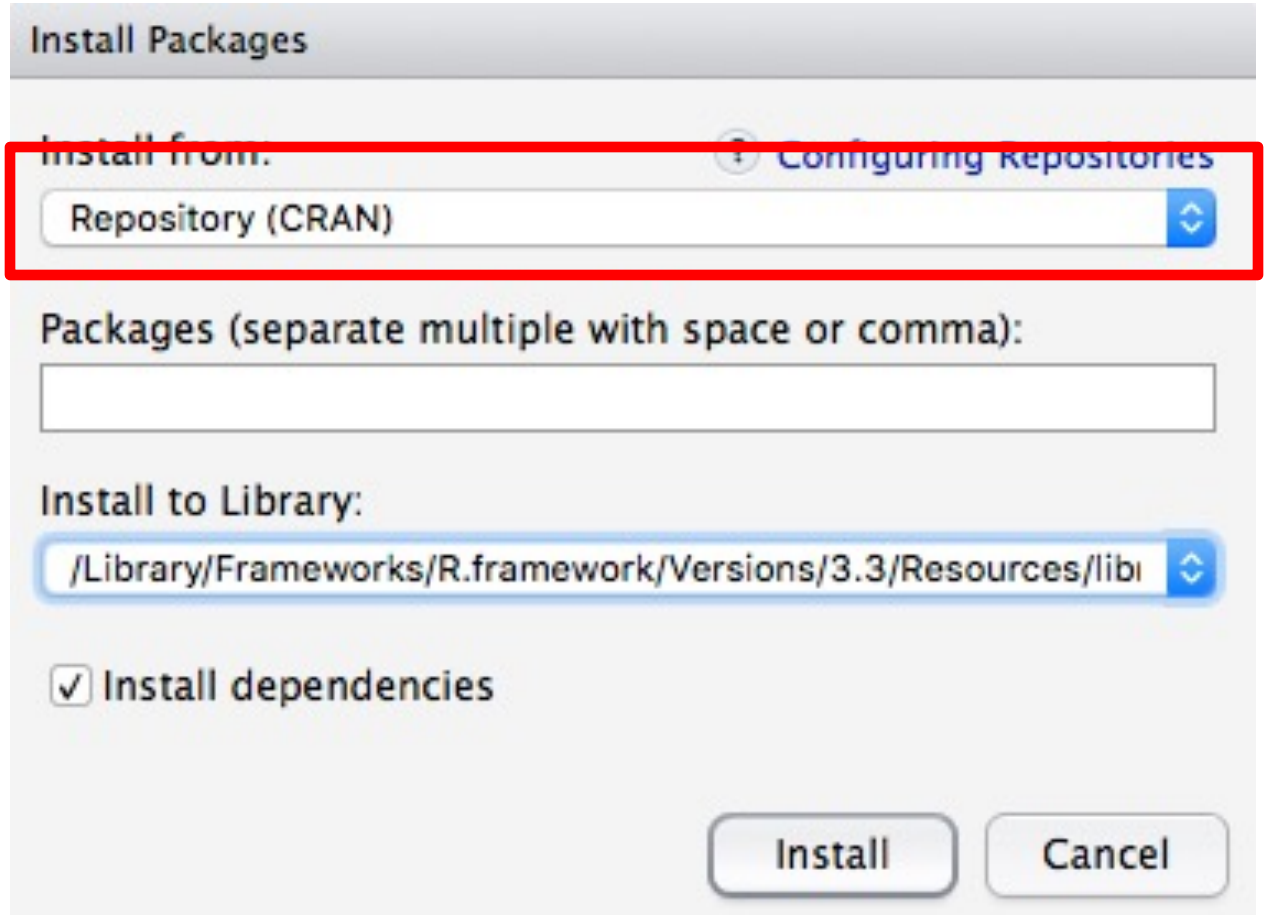
2 ways to install packages

1) Install button in the Packages window

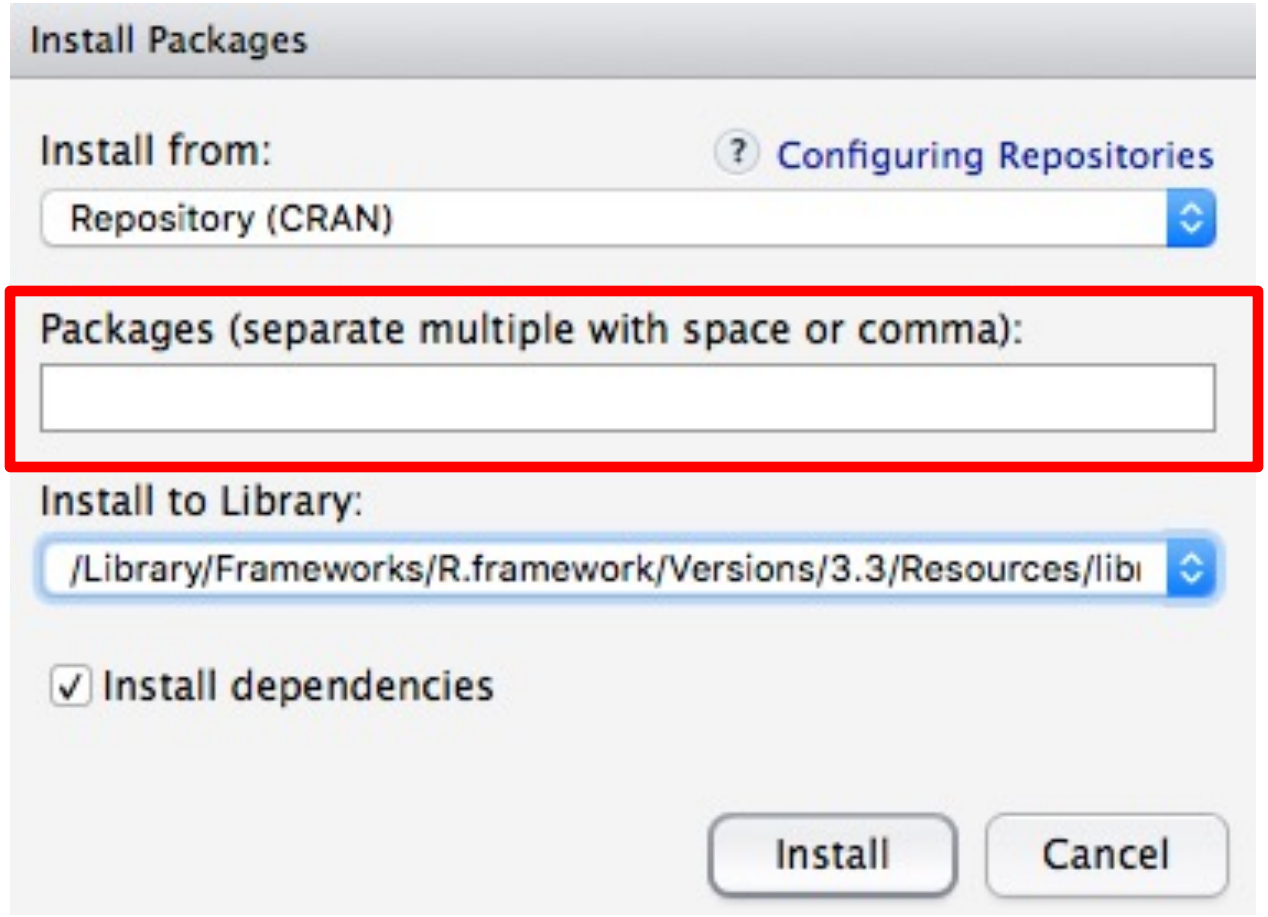
2) R Code

Either way, you'll need to know the name of the package.

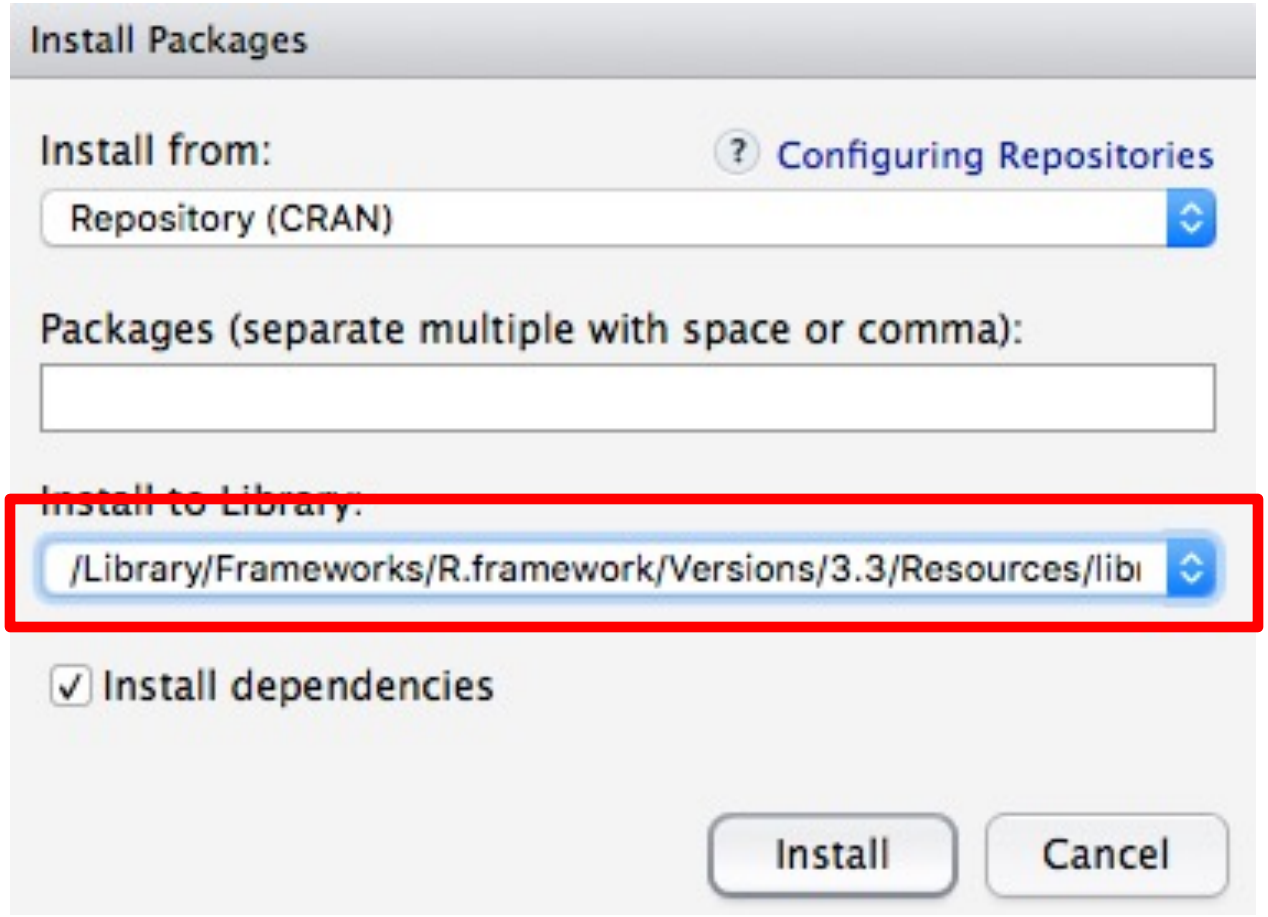
Install Button



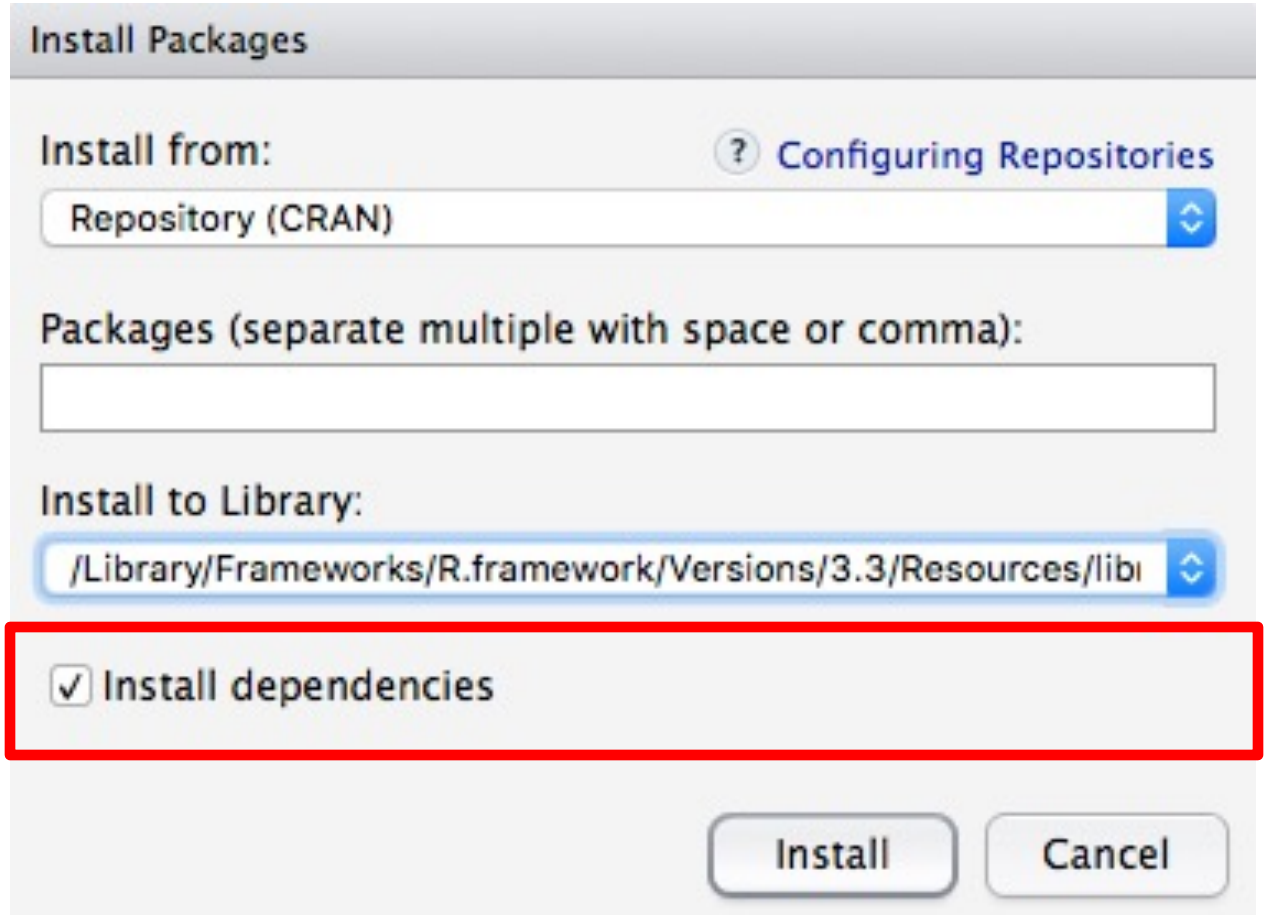
Install Button



Install Button



Install Button



R Code to Install Packages

Note the quotation marks!

```
install.packages("psych")
```

Packages

INSTALLING

- Downloading the package from the internet and saving it to your computer.
- The package is available for use.

LOADING

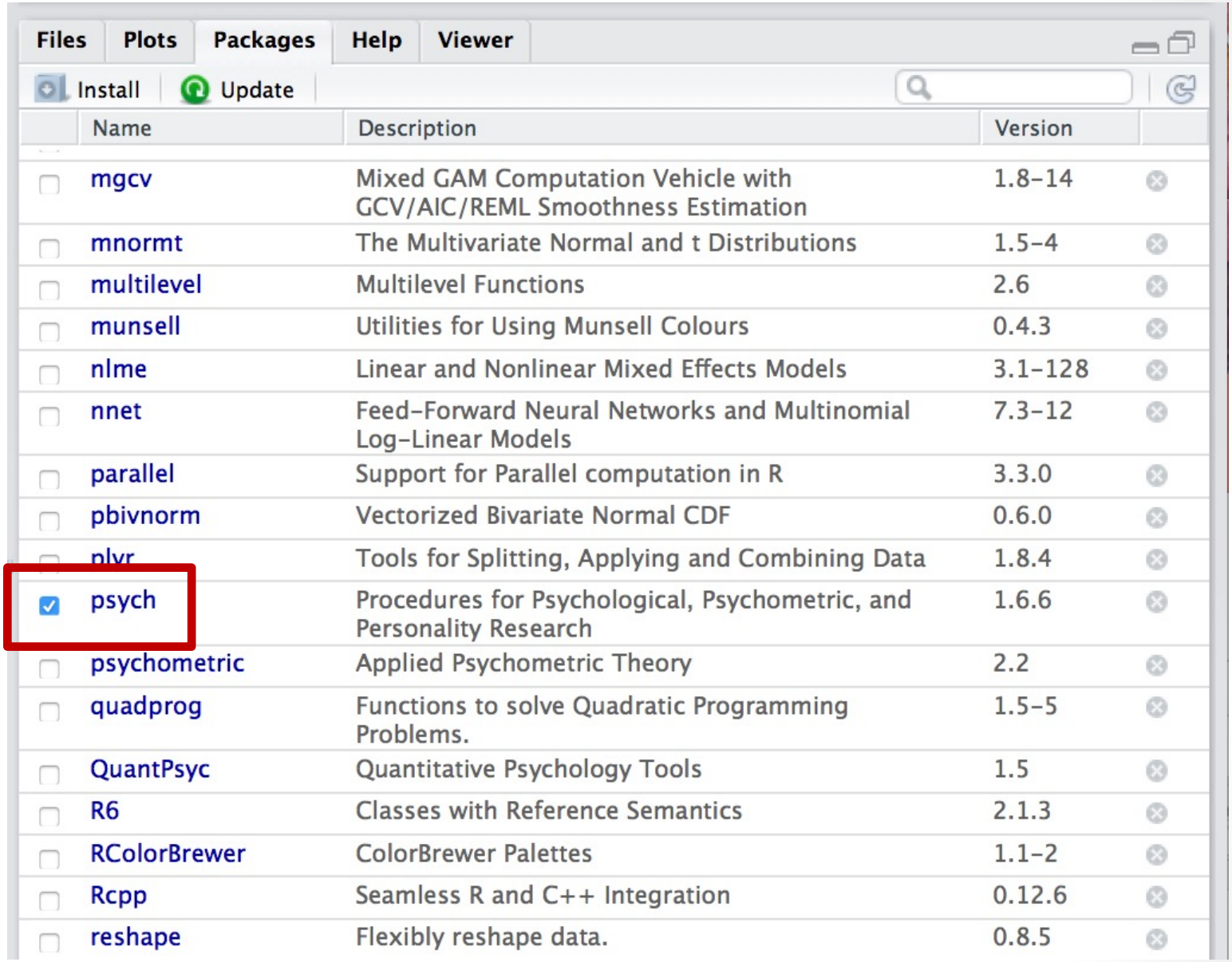
- Calling the package from your computer and reading its contents.
- R is ready to use the functions in the package.

2 ways to load packages

1) Checkbox in the Packages window

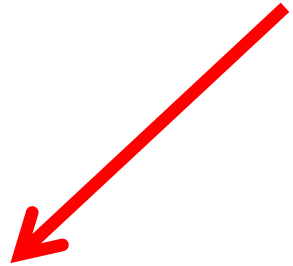
2) R Code

Either way, you'll need to know the name of the package.


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
mgcv
Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation
1.8-14
mnormt
The Multivariate Normal and t Distributions
1.5-4
multilevel
Multilevel Functions
2.6
munsell
Utilities for Using Munsell Colours
0.4.3
nlme
Linear and Nonlinear Mixed Effects Models
3.1-128
nnet
Feed-Forward Neural Networks and Multinomial Log-Linear Models
7.3-12
parallel
Support for Parallel computation in R
3.3.0
pbivnorm
Vectorized Bivariate Normal CDF
0.6.0
plyr
Tools for Splitting, Applying and Combining Data
1.8.4
psych
Procedures for Psychological, Psychometric, and Personality Research
1.6.6
psychometric
Applied Psychometric Theory
2.2
quadprog
Functions to solve Quadratic Programming Problems.
1.5-5
QuantPsyc
Quantitative Psychology Tools
1.5
R6
Classes with Reference Semantics
2.1.3
RColorBrewer
ColorBrewer Palettes
1.1-2
Rcpp
Seamless R and C++ Integration
0.12.6
reshape
Flexibly reshape data.
0.8.5

R Code to Load Packages

Note: NO quotation marks!



```
library(psych)
```

Dependencies

```
> library(lme4)
Loading required package: Matrix
Loading required package: Rcpp
```

Uses functions from another packages.

Installed automatically.

Loaded automatically.

RStudio

starwarscode.R

```
install.packages("dplyr")
install.packages("ggplot2")

library(dplyr)
library(ggplot2)

empire <- starwars

empire <- empire[c(1:5, 10, 13, 14, 19, 21), c(1:3,8:10)]
empire$species <- factor(empire$species)
empire <- data.frame(empire)

ggplot(data = empire, aes(x = mass, y = height)) +
  geom_point(aes(size = mass, color = species), alpha = .5) +
  labs(title = "Star Wars Characters", subtitle = "by size") +
  scale_size(guide = "none")
```

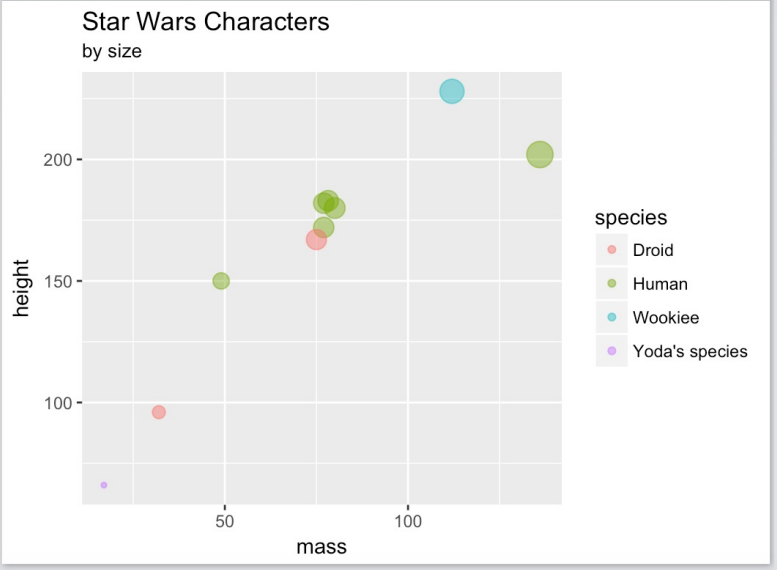
Console ~/Dropbox/

```
> empire <- starwars
>
> empire <- empire[c(1:5, 10, 13, 14, 19, 21), c(1:3,8:10)]
> empire$species <- factor(empire$species)
> empire <- data.frame(empire)
>
> ggplot(data = empire, aes(x = mass, y = height)) +
+   geom_point(aes(size = mass, color = species), alpha = .5) +
+   labs(title = "Star Wars Characters", subtitle = "by size") +
+   scale_size(guide = "none")
>
```

Environment | History

Data

empire 10 obs. of 6 variables



Exercise

1) Check your **Packages** tab to see if the following three packages are listed there:

- psych
- lsr
- tidyverse

- If NO, proceed to Step 2!

2) INSTALL the above three packages.

3) LOAD the above three packages.

Remember:

- **Need quotes** to install. **No quotes** to load.
- Spelling and capitalization matter.

```
install.packages("psych")
install.packages("lsr")
install.packages("tidyverse")
```

```
library(psych)
library(lsr)
library(tidyverse)
```

Help! (again)

Ways to find documentation:

`?psych` – opens documentation specific to that package or function

`??psych` – searches for this in all documentation

**Only looks in documentation for packages you have installed and loaded.

To find a package that does what you need: GOOGLE! ☺

Help! (again)

To find functions available in a package:

In the packages tab, click on the name of the package to see what functions are available!

Exercise

1. Make a new data.frame that consists of 2 column vectors:
   - `person` is a vector with the names of 5 people:
     - Kendra, Emorie, Violet, Maddy, and you
   - `pets` is a vector with the types of pets that those people have:
     - Kendra has GumNut, Emorie has an axolotl, Violet has a gorilla, and Maddy has a dog, and you have a _____
   - Hint: Make sure you combine the 2 vectors into a `data.frame()`
     - Name the data.frame whatever you want.
     - Make sure the names of the columns are “Name” and “Animal”, respectively (see Day 1 for reference)
     - Be sure to include the argument `stringsAsFactors = FALSE`

2. Tell R that the `Animal` vector (in your data.frame) is a `factor`.
   - Hint: how can you acce$$ just the `Animal` column of your data.frame? Check the class of `Animal` to verify al went as planned.

Exercise

3. Make a new vector called `humanAge`, and add it to your new data.frame. The column header should be “Age”.
   - Kendra is 75 – Emorie is 27 – Violet is 25 – Maddy is 89. You are ##

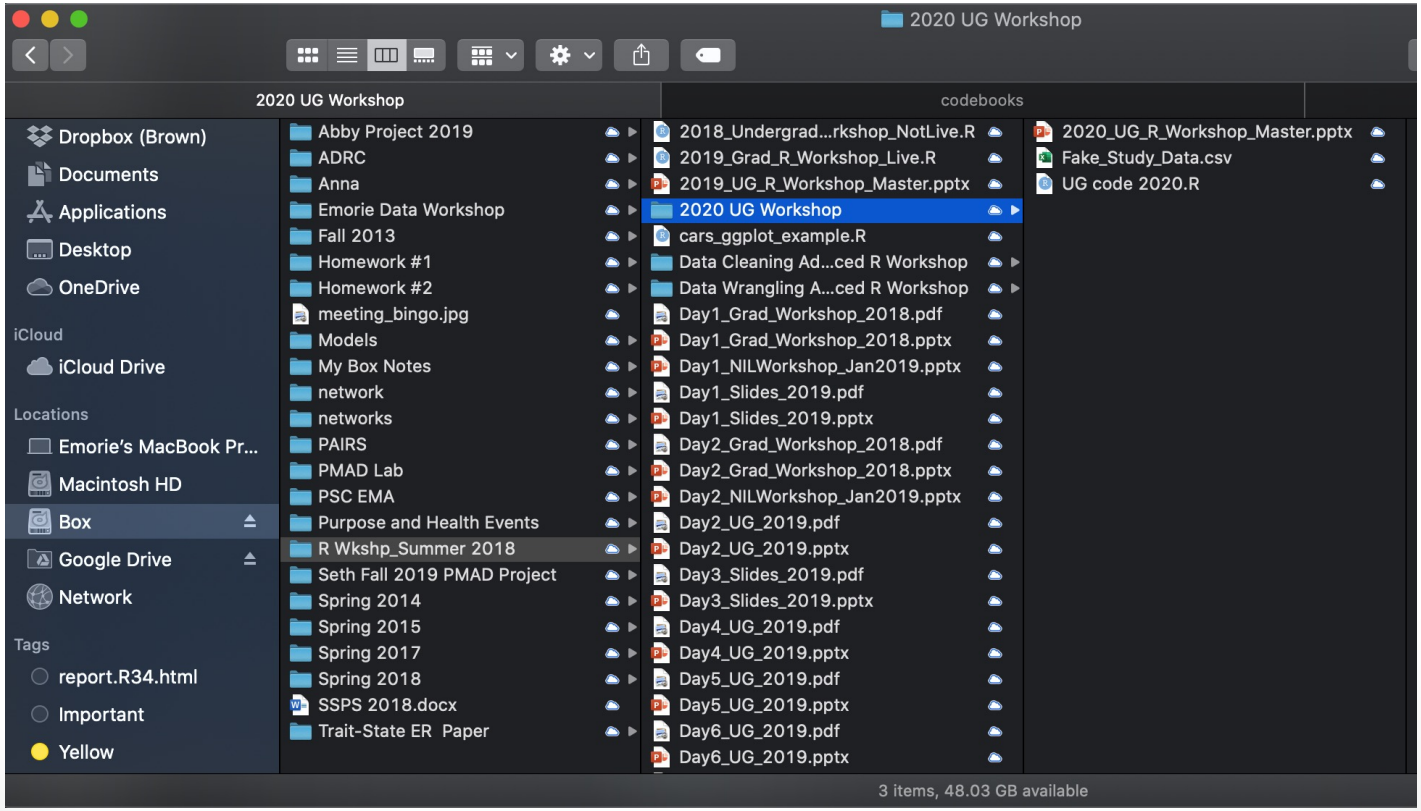
4. Convert `Age` (located within the data.frame) to dog years – save as `dogAge` vector.
   - 1 human year = 7 dog years
   - Hint: how can you acce$$ just the `Age` column of your data.frame?

5. Another way to combine a vector with a data.frame is to use the `cbind()` function. It stands for ”column bind”. Use it to combine your data.frame and the `dogAge` vector into a single data.frame.
   - Hint: you should end with 4 columns and 4 rows.

DIRECTORIES

Directories

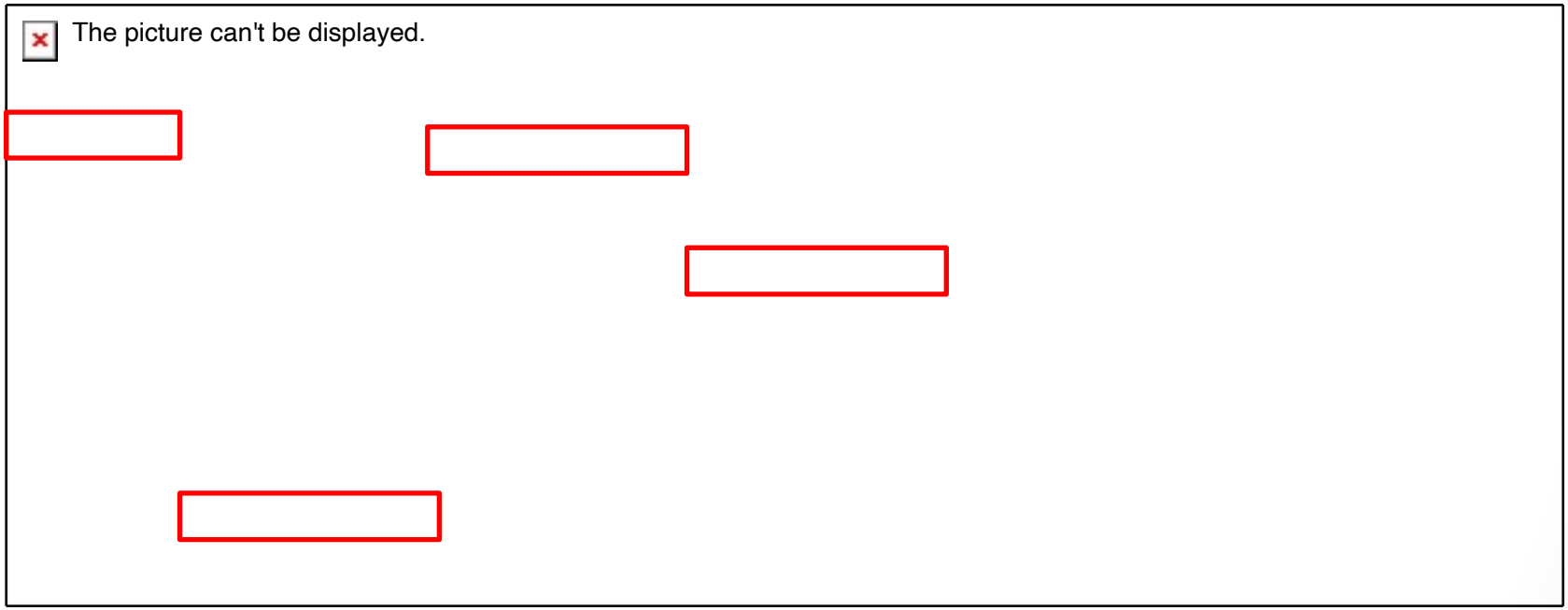
- Your computer is made up of a series of folders.



Directories

This file path is the directory that contains the file “PPT.pptx”

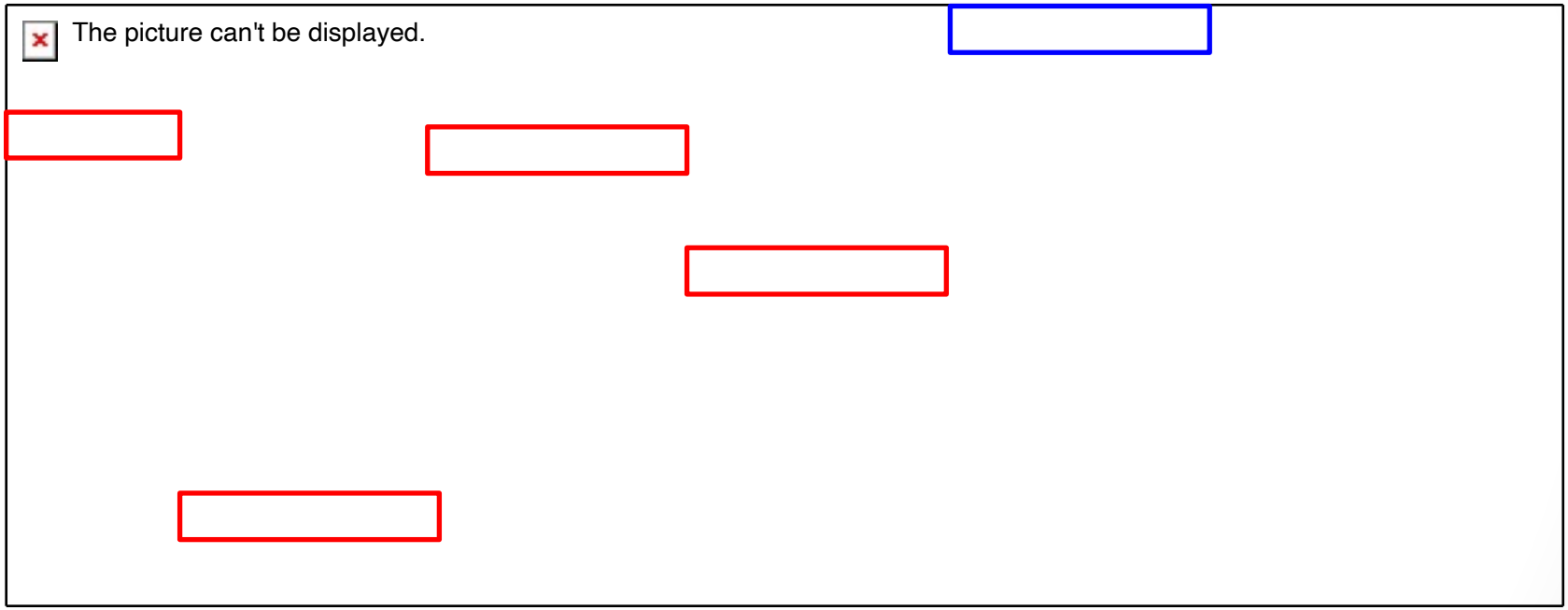
/Users/zashawks/Desktop/Stats/Q1/R workshop/PPT.pptx



Directories

This is the actual file “PPT.pptx”

/Users/zashawks/Desktop/Stats/Q1/R workshop/PPT.pptx



Setting your working directory

R will only look in one folder to read your files.

Working Directory

- Where R is going to ***look for*** files
- Where R is going to ***save*** files

Setting your working directory

```
getwd()
```

```
setwd("/yourpath/goes/here")
```

Hint: press tab within the quotes and see what happens

(unimportant bonus info):

In R, ~ points to a very general directory by default. On Macs, this is usually /Users/YourName. On PCs, this is C:/Users/YourName. If the directory you want is located within this general directory, then you can use the ~ to shorten the file path. Ex:

```
setwd("~/goes/here")
```

Exercise

1. Create a folder on your desktop for today's R workshop.

2. `Set` your working directory to that folder

3. Try the `getwd()` function

TYPES OF FILES

Types of files

.R

Your data

.Rdata (or .RData or .rdata or .rda; all are equivalent)

.R files

.R files are text files.

- They contain the code that you've written – the commands that you want R to run.

Equivalent to syntax files in SPSS.

Also called **scripts**.

.R files – why use them?

Keep track of what commands you use.

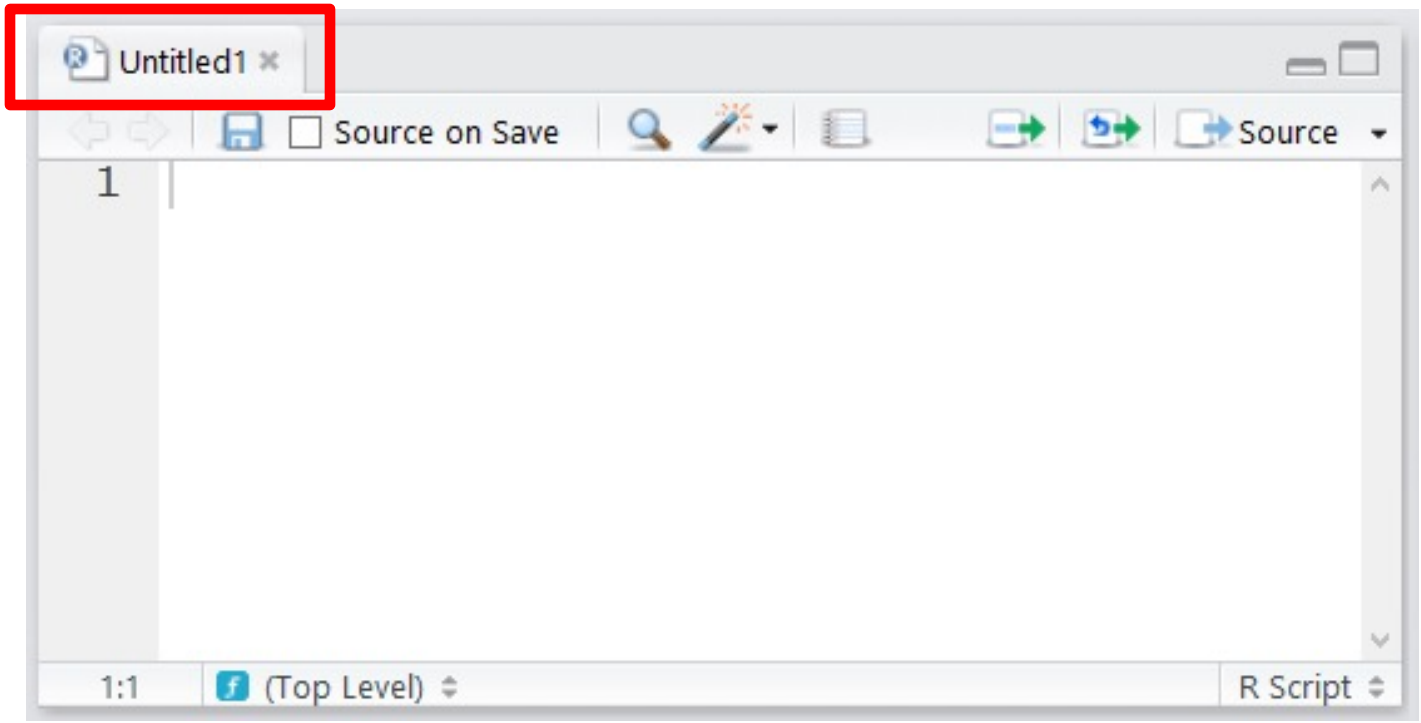
Save only the commands that are useful.

Make notes to yourself!

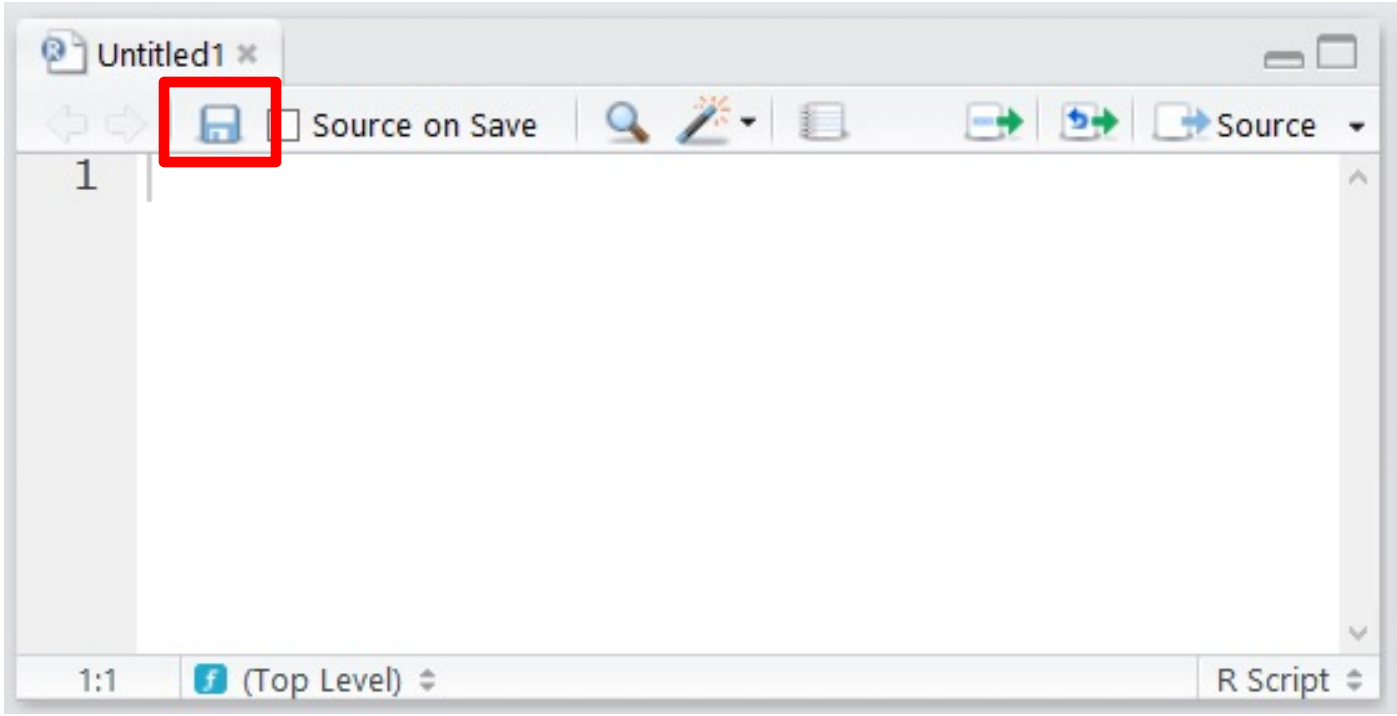
- `#Updated code for R workshop!`
- `#reliability estimates for depression scale`
- `#scatter plot for BMI predicting diabetes diagnosis`

Share your analyses with collaborators and readers.

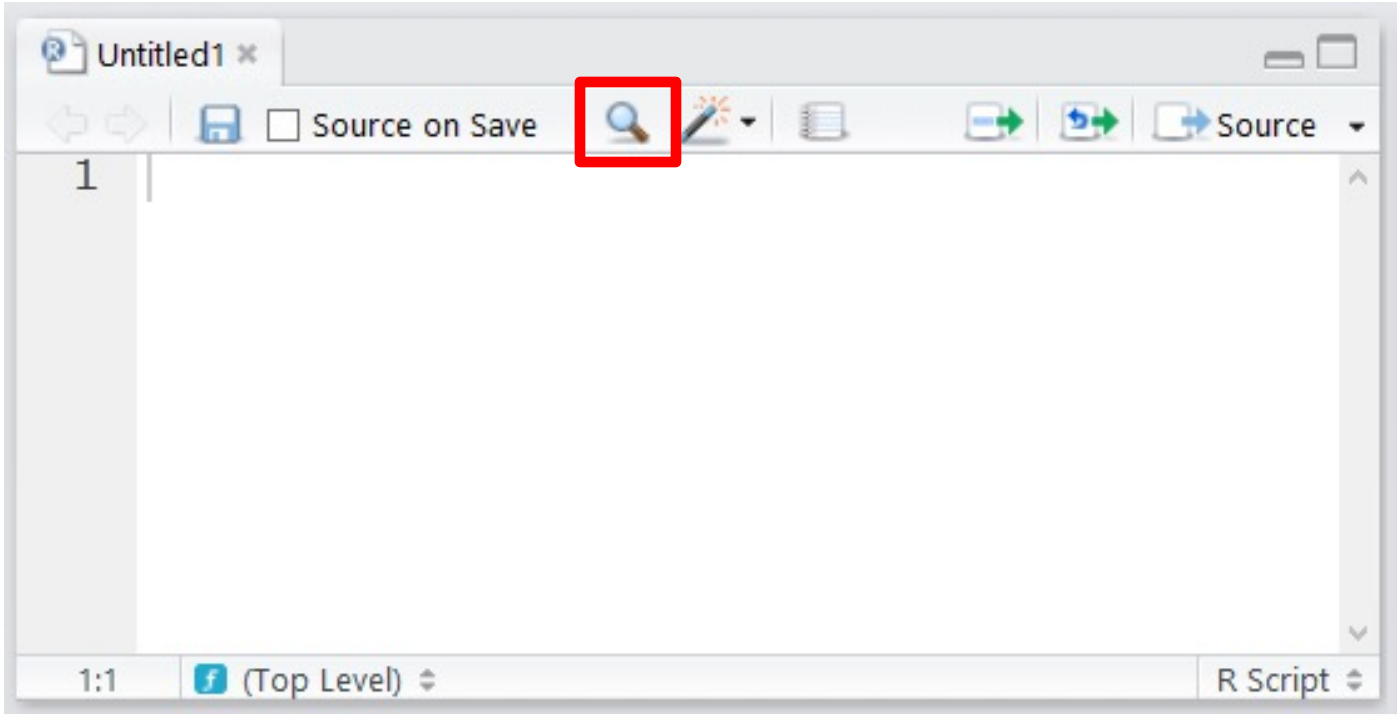
.R files



.R files



.R files



Open your file

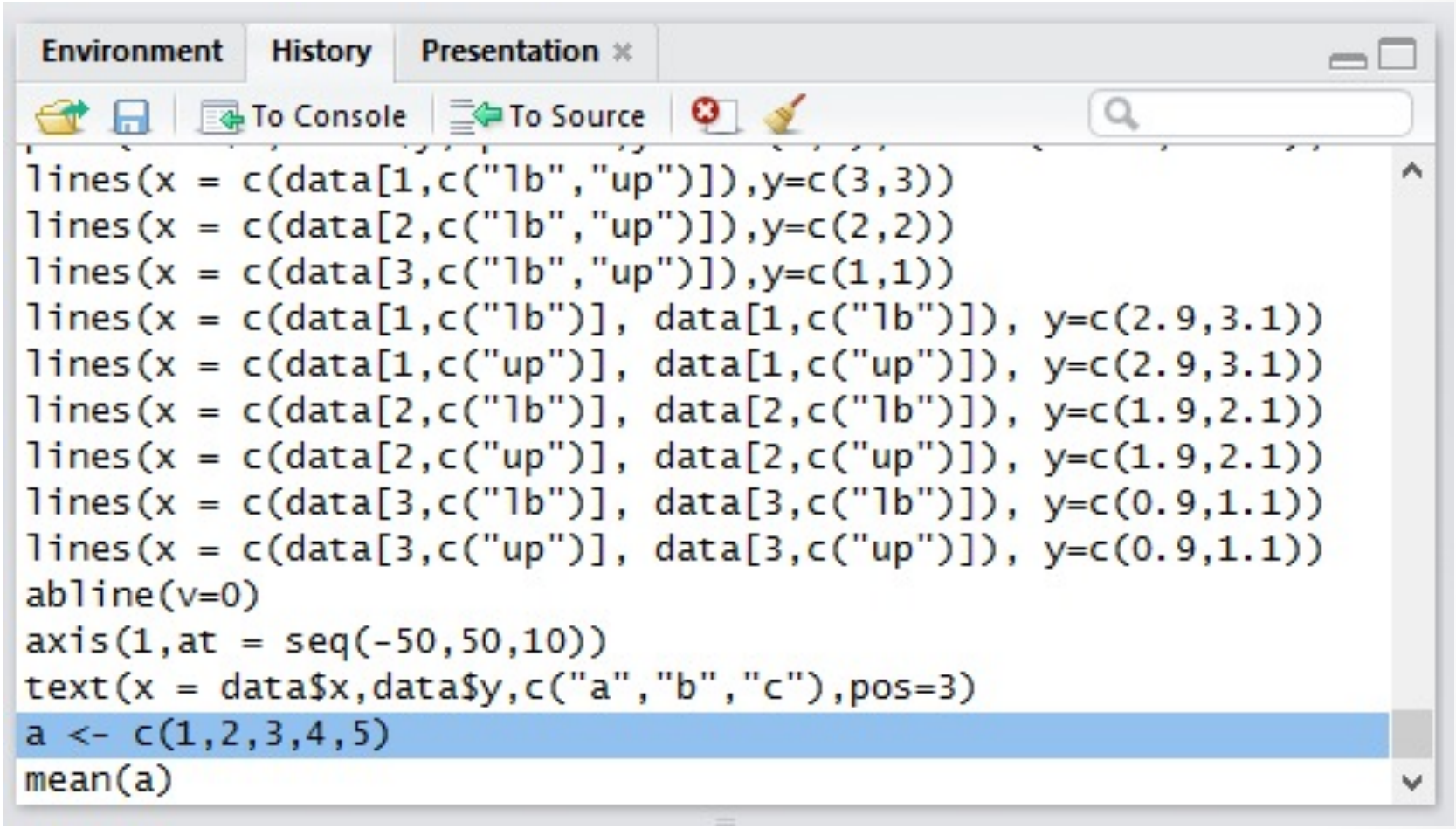
Open a file like you would any other document.

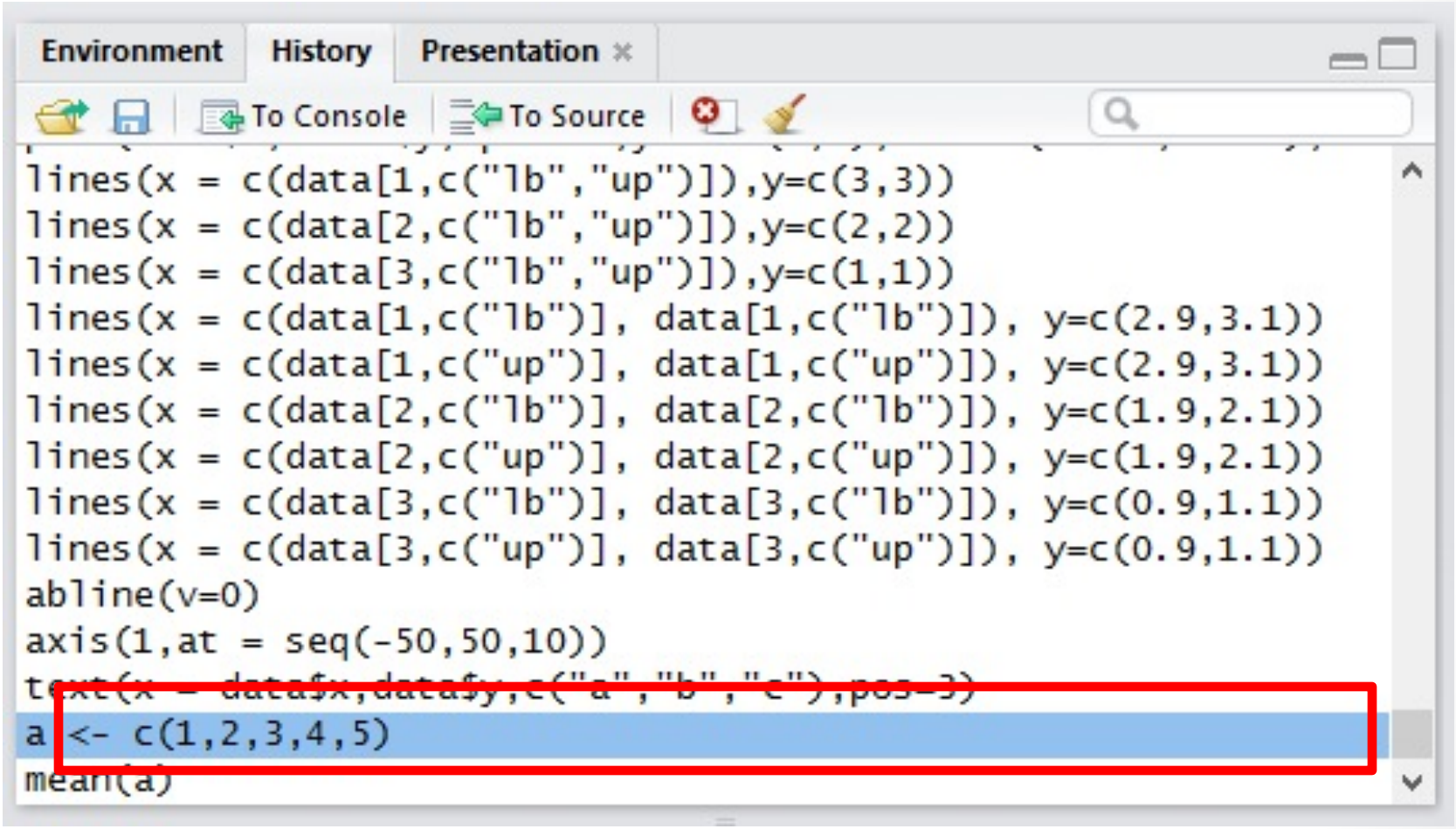
How to add things to your .R file

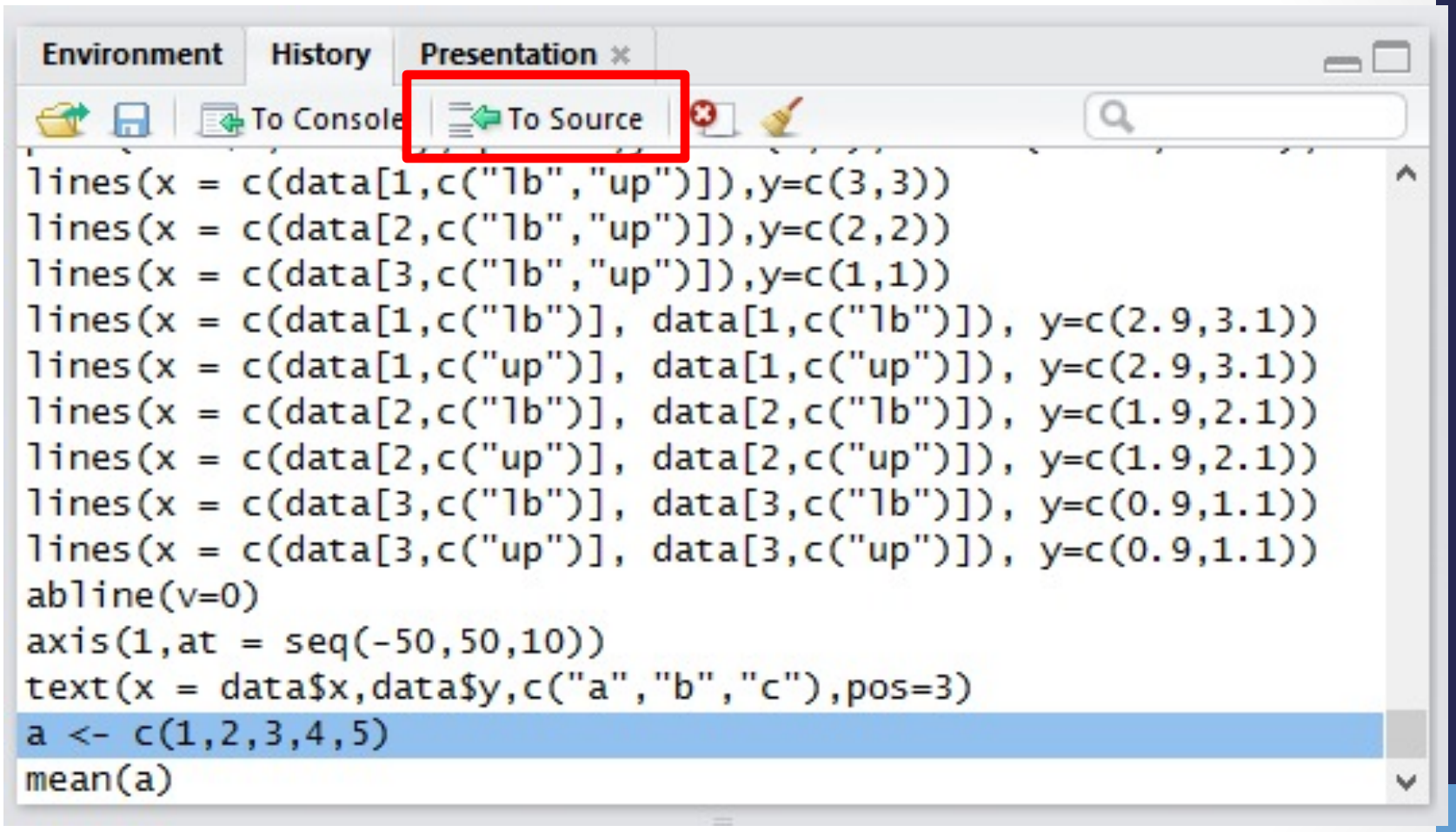
- Type into it!

OR

- Copy over from the History tab.


Environment
History
Presentation
To Console
To Source
lines(x = c(data[1,c("lb","up")]),y=c(3,3))
lines(x = c(data[2,c("lb","up")]),y=c(2,2))
lines(x = c(data[3,c("lb","up")]),y=c(1,1))
lines(x = c(data[1,c("lb")], data[1,c("lb")]), y=c(2.9,3.1))
lines(x = c(data[1,c("up")], data[1,c("up")]), y=c(2.9,3.1))
lines(x = c(data[2,c("lb")], data[2,c("lb")]), y=c(1.9,2.1))
lines(x = c(data[2,c("up")], data[2,c("up")]), y=c(1.9,2.1))
lines(x = c(data[3,c("lb")], data[3,c("lb")]), y=c(0.9,1.1))
lines(x = c(data[3,c("up")], data[3,c("up")]), y=c(0.9,1.1))
abline(v=0)
axis(1,at = seq(-50,50,10))
text(x = data$x,data$y,c("a","b","c"),pos=3)
a <- c(1,2,3,4,5)
mean(a)


Environment
History
Presentation
To Console
To Source
lines(x = c(data[1,c("lb","up")]),y=c(3,3))
lines(x = c(data[2,c("lb","up")]),y=c(2,2))
lines(x = c(data[3,c("lb","up")]),y=c(1,1))
lines(x = c(data[1,c("lb")], data[1,c("lb")]), y=c(2.9,3.1))
lines(x = c(data[1,c("up")], data[1,c("up")]), y=c(2.9,3.1))
lines(x = c(data[2,c("lb")], data[2,c("lb")]), y=c(1.9,2.1))
lines(x = c(data[2,c("up")], data[2,c("up")]), y=c(1.9,2.1))
lines(x = c(data[3,c("lb")], data[3,c("lb")]), y=c(0.9,1.1))
lines(x = c(data[3,c("up")], data[3,c("up")]), y=c(0.9,1.1))
abline(v=0)
axis(1,at = seq(-50,50,10))
a <- c(1,2,3,4,5)
mean(a)


Environment
History
Presentation
To Console
To Source
lines(x = c(data[1,c("lb","up")]),y=c(3,3))
lines(x = c(data[2,c("lb","up")]),y=c(2,2))
lines(x = c(data[3,c("lb","up")]),y=c(1,1))
lines(x = c(data[1,c("lb")], data[1,c("lb")]), y=c(2.9,3.1))
lines(x = c(data[1,c("up")], data[1,c("up")]), y=c(2.9,3.1))
lines(x = c(data[2,c("lb")], data[2,c("lb")]), y=c(1.9,2.1))
lines(x = c(data[2,c("up")], data[2,c("up")]), y=c(1.9,2.1))
lines(x = c(data[3,c("lb")], data[3,c("lb")]), y=c(0.9,1.1))
lines(x = c(data[3,c("up")], data[3,c("up")]), y=c(0.9,1.1))
abline(v=0)
axis(1,at = seq(-50,50,10))
text(x = data$x,data$y,c("a","b","c"),pos=3)
a <- c(1,2,3,4,5)
mean(a)

Your Data

Original data files

- Most of the time, these are going to be either .csv or .txt, depending on how you collect data.

These are not altered by R!

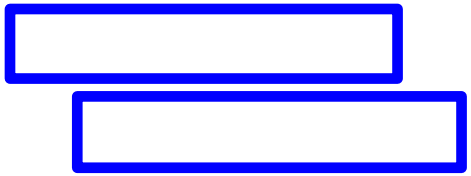
If your data is not a **.csv** or **.txt** file, don't worry! R can do a lot of stuff!

We will work with .csv, just to keep things simple.

Loading .csv files

Loading .csv files

The picture can't be displayed.



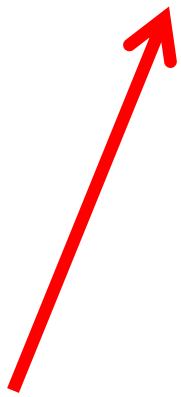
Loading .csv files

he picture can't be displayed.


The picture can't be displayed.

Loading .csv files

```
anxiety <- read.csv("~/Box Sync/R
Wkshp_Undergrad_Summer 2018/Fake_Study_Data.csv")
```



This will appear in your console

We strongly recommend copying and pasting this code into your script file!

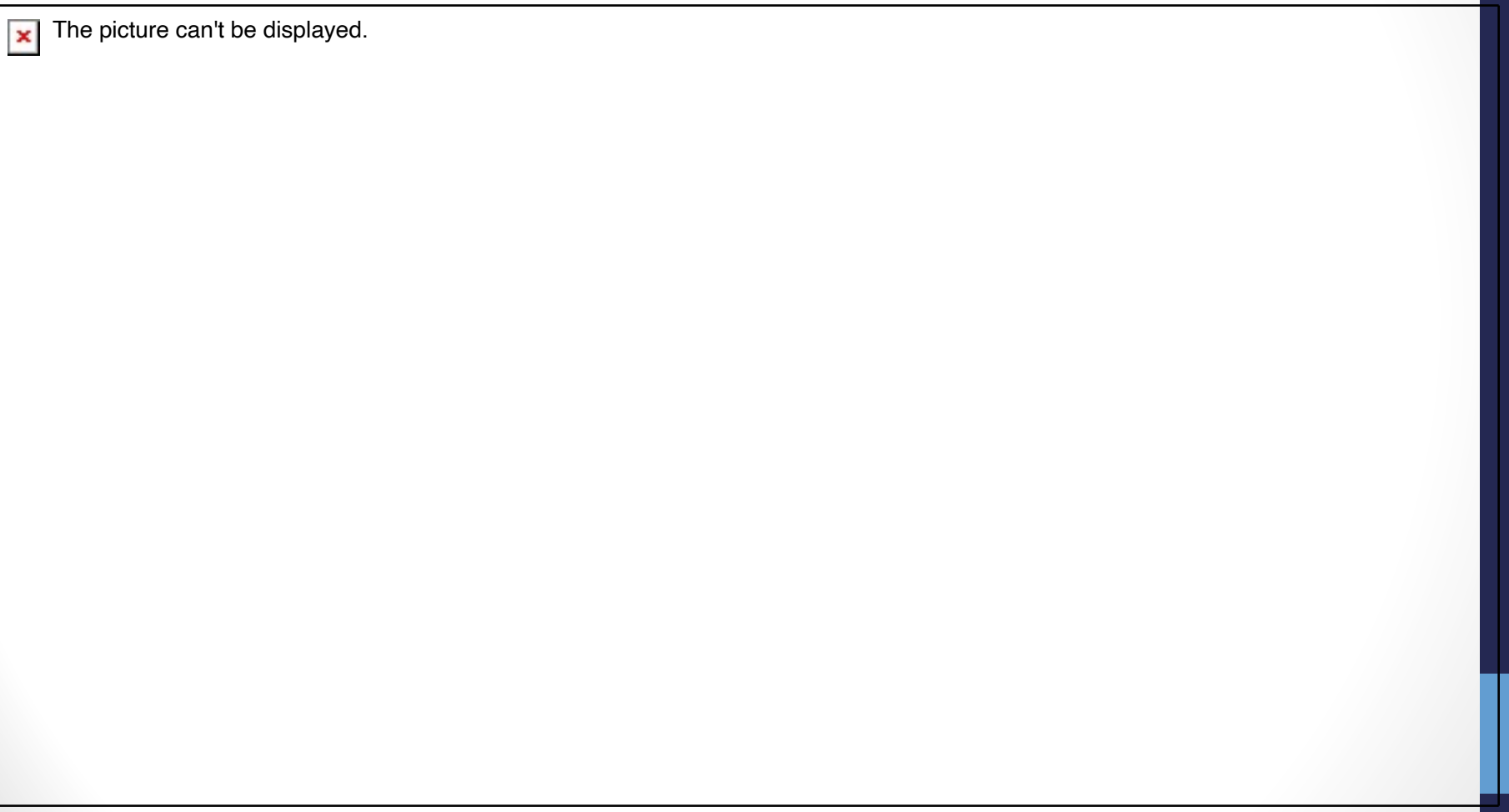
Typical workflow in R

1. Open your data (usually from a .csv file)

2. Run analyses in the console (or in your script file)

3. Save the stuff you want to be able to run again later
   - Save to a .R file
   - Make sure that this includes the code to pull your .csv from your desktop/Dropbox/Box/etc.
     - Note: R doesn't change this original file!

Typical Format of .R File



Exercise

1. Clear your Global Environment using the broom icon.
2. Open a new .R script file.

FOR THE FOLLOWING STEPS, TYPE ANY CODE DIRECTLY INTO YOUR NEW SCRIPT FILE:

3. Load the following packages:
   - psych
   - dplyr
   - ggplot2

4. `Get` your working directory: make sure it's the folder on your Desktop that you made on Tuesday. **Hint:** you might need to re`set` your working directory.

5. Drag and drop the Fake_Study_Data.csv file into that same folder.

Exercise, continued:

6. Import data file and rename as **anxiety**

7. Make sure that the import code ends up in your SCRIPT file & then Save the script file.

Exercise, continued:

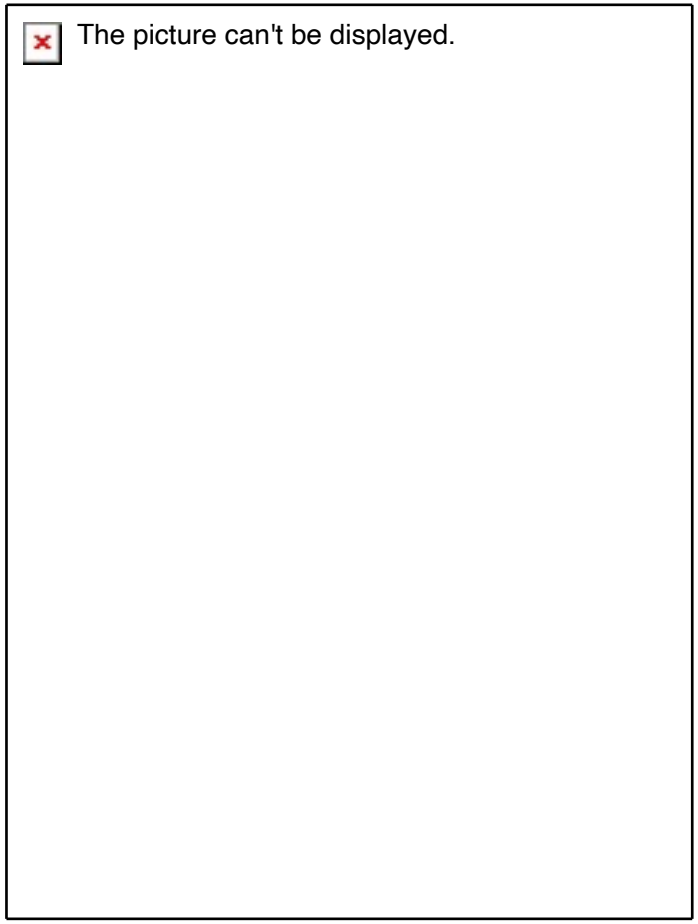
Now…

Clear your global environment again

Select entire script file and RUN!

Hint: You can also click “Source” in the upper right corner of your script file in order to run the entire script (rather than highlighting everything)

REMOVE POST-ITS!!



(and save!)

Anxiety Data

Clinical trial for a new anxiety treatment

- Participants 18 to 50 years old
- Randomly assigned “treatment” or “placebo” group
- Socioeconomic status (SES)
  - Scale: 0 (low SES) to 10 (high SES)
- Anxiety scores: **Self**-Reported and **Relative**-Reported
  - Scale: 0 (no anxiety) to 100 (severe anxiety)
- Stress
  - Scale: 0 (no stress) to 30 (high stress)
- Drug Use
  - Scale: 0 (no drug use) to 50 (high drug use)

Quick review

What sign would you use if you wanted to access the Stress variable from your anxiety data.frame?

```
anxiety$Stress
```

Indexing…continued!

Indexing a **vector** (1 dimension):

```
anxiety$Age[1]
```

- The number `1` gives us the **1st** item in the Age vector.

Indexing a **data.frame** (2 dimensions):

```
anxiety[1, 2:3]
```

- This gives us the 1st row and the 2nd thru 3rd column of this data.frame.

Logical Operator Review

- Returns a value of TRUE or FALSE (Boolean)

==	equality
!=	inequality
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

A = 3 – "A is an object that stores the number 3."

A == 3 – "Is A equal to 3?"

Logical Operators

- Multiple conditions

&	and
|	or
!	not

Is this true? AND Is this true?

```
(10 < 100  &  24 == 23 + 1)
```

TRUE

```
(5 > 4  &  5 > 10)
```

FALSE

Is this true? OR Is this true?

```
(5 > 4  |  5 > 10)
```

TRUE

```
(A == B  |  C != D)
```

TRUE

Subset your data

Sometimes, you will want to perform functions on only some of your data points

You can subset your data to identify subjects in a certain subgroup (e.g., females, persons over 40)

```
filter(.data = mtcars, cyl > 5)
```

Note: Some functions have a “data” argument, where you name your data.frame. Here, you don’t need to use the dollar $ign.

Exercise

1) Make a new data.frame that only includes the treatment group, not the placebos.

   Hint: Give a new name to your new data.frame.

2) Make a new data.frame that only includes people in the placebo group that are younger than 35.

   Hint: Give a new name to your new data.frame.

3) In the original `anxiety` data.frame, find the number of people in the placebo group and the number of people in the treatment group. Use `table()`.

What if I don't remember the arguments?

Check the R documentation

```
?filter
```

When in doubt, you can always search the "Help" tab or the Internet

Back to Arguments

Reminder: Most functions take more than one argument.

Reminder: Separate arguments with commas.

Reminder: **Most** arguments in functions have names – use them!

```
round (x = 2.30467, digits = 3)
[1] 2.305
```

Exercise

- Look up the `subset()` function.
- What is the difference between arguments called `subset =` and `select =` ?
- What function do you have in your repertoire that's similar to `subset()` ?

Functions within Functions

Often you can wrap functions within functions

This is called “nesting”

```
round(x = sqrt(86), digits = 1)
[1] 9.3
```

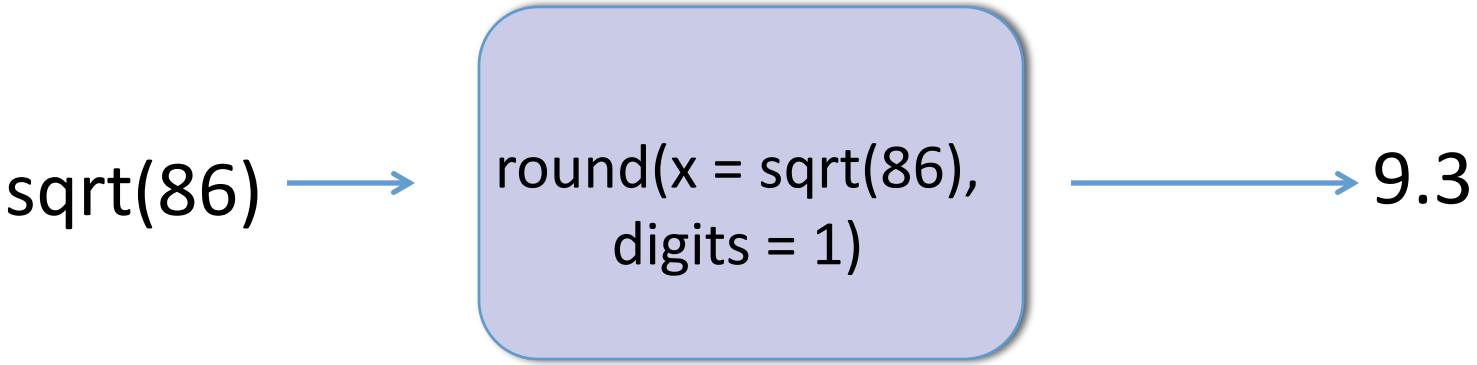
Visualizing Functions

Object
(Argument)

NESTED
functions
(Transformation)

Transformed Object
(Output)

Visualizing Functions



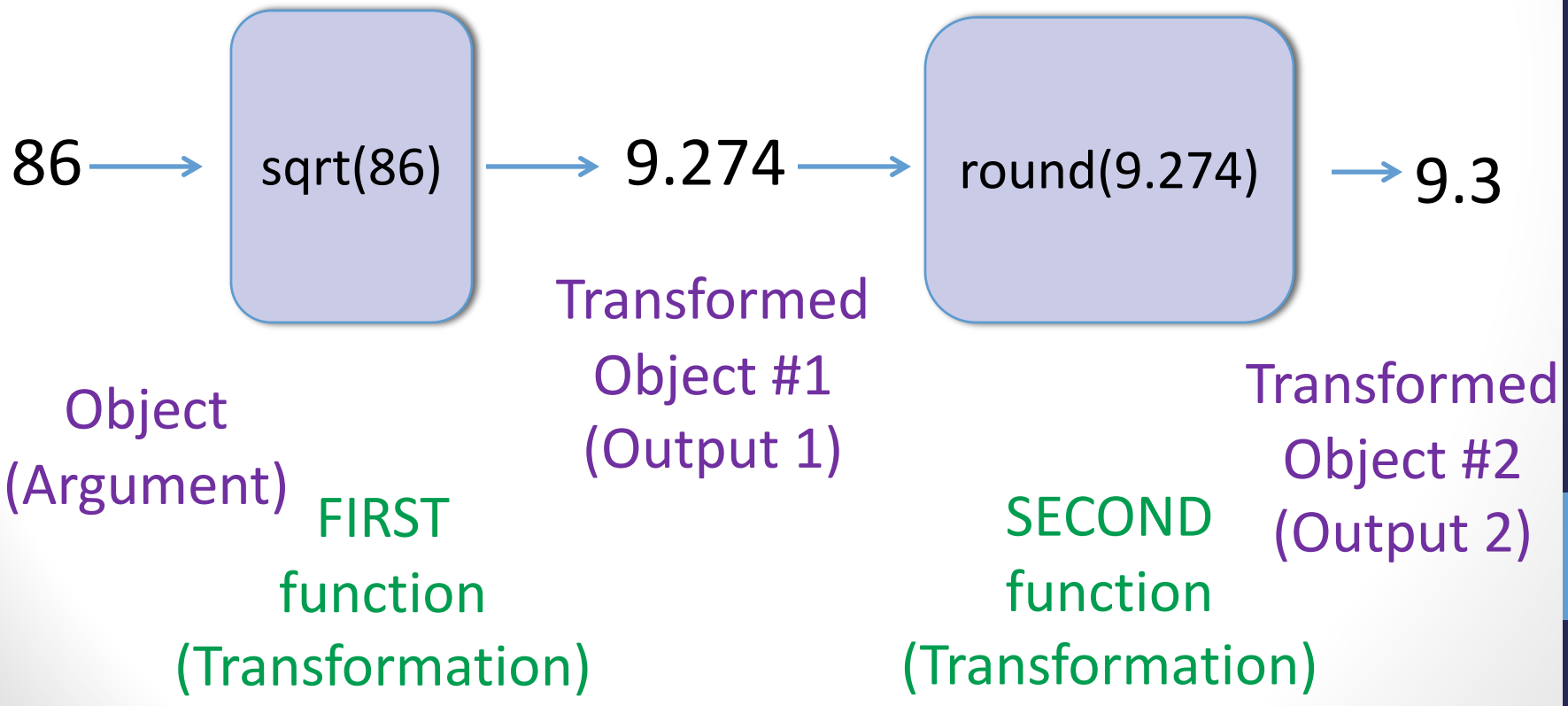
Object (Argument)

NESTED functions (Transformation)

Transformed Object (Output)

Visualizing Nested Functions

```
round(x = sqrt(86), digits = 1)
```



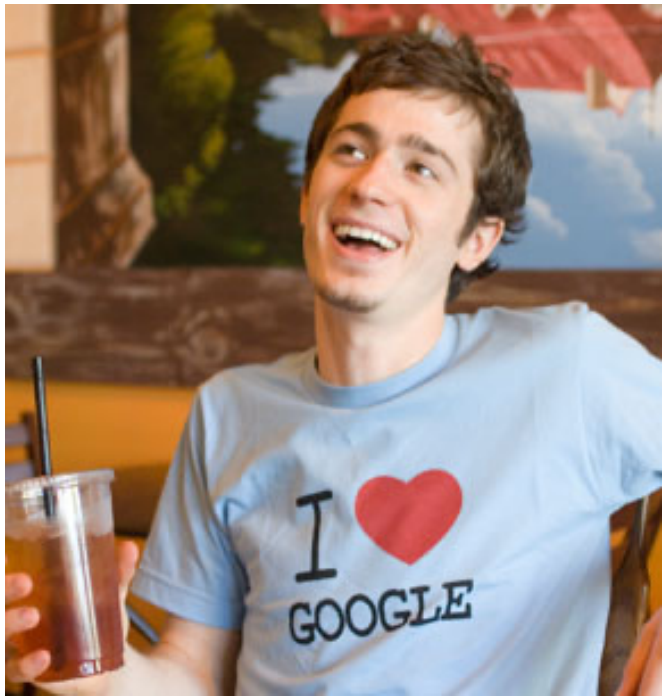
Exercise

Use nested functions to create a subsetted data.frame which contains only participants who have `Anxiety_SelfReport` scores above the `mean`.

Hint: Name this new data.frame!

What if I don't remember how to use a function?

Use the internet!



Any problem you have, the internet has already solved it!

Google is your best friend!

Recap #1

What's the correct way to subset all rows of columns 3, 5, 7 of my anxiety data.frame – *without* changing the data.frame?

A) `anxiety <- anxiety[ ,c(3, 5, 7)]`
B) `anxiety2 <- anxiety[ ,c(3, 5, 7)]`
C) `anxiety2 <- anxiety[3, 5, 7, ]`
D) `anxiety2 <- anxiety[c(3, 5, 7), ]`

Recap #1

What's the correct way to subset all rows of columns 3, 5, 7 of my anxiety data.frame – *without* changing the data.frame?

A) `anxiety <- anxiety[ ,c(3, 5, 7)]`

B) `anxiety2 <- anxiety[ ,c(3, 5, 7)]`

C) `anxiety2 <- anxiety[3, 5, 7, ]`

D) `anxiety2 <- anxiety[c(3, 5, 7), ]`

Recap #2

Okay, now I want to add a Hi-Low SES variable (imagine it exists as a vector in your environment) to my anxiety dataset! How can I do this? (**SELECT ALL THAT APPLY!**)

A) `anxiety <- cbind(anxiety, Hi_LowSES)`

B) `anxiety <- rbind(anxiety, Hi_LowSES)`

C) `anxiety <- anxiety["Hi_LowSES"]`

D) `anxiety["Hi_LowSES"] <- Hi_LowSES`

E) `anxiety$Hi_LowSES <- Hi_LowSES`

Recap #2

Okay, now I want to add a Hi-Low SES variable to my anxiety dataset so that I can use it for my analyses! How can I do this? (**MORE THAN ONE MAY BE CORRECT!**)

A) `anxiety <- cbind(anxiety, Hi_LowSES)`

B) `anxiety <- rbind(anxiety, Hi_LowSES)`

C) `anxiety <- anxiety["Hi_LowSES"]`

D) `anxiety["Hi_LowSES"] <- Hi_LowSES`

E) `anxiety$Hi_LowSES <- Hi_LowSES`

Recap #3

How often do you need to install packages? Load packages?

How would you read this code?

- `(Age >= 35 | Anxiety == 50)`
- `(A == B & B == C)`

`abs(median(x))`

- Which function is performed first? `abs()` or `median()`?
- BONUS: what's it called when one function is wrapped another function?

ADD AND FILTER DATA

Remember Subsetting?

Subsetting an entire group:

```
filter(.data = anxiety,
          Treatment_Group == “placebo”)

filter(.data = anxiety,
          Treatment_Group != “treatment”)
```

What are each of these function calls doing?

Adding and Filtering Data

Why would you **add** data?

- New variable
- Collected new subjects
- Etc.

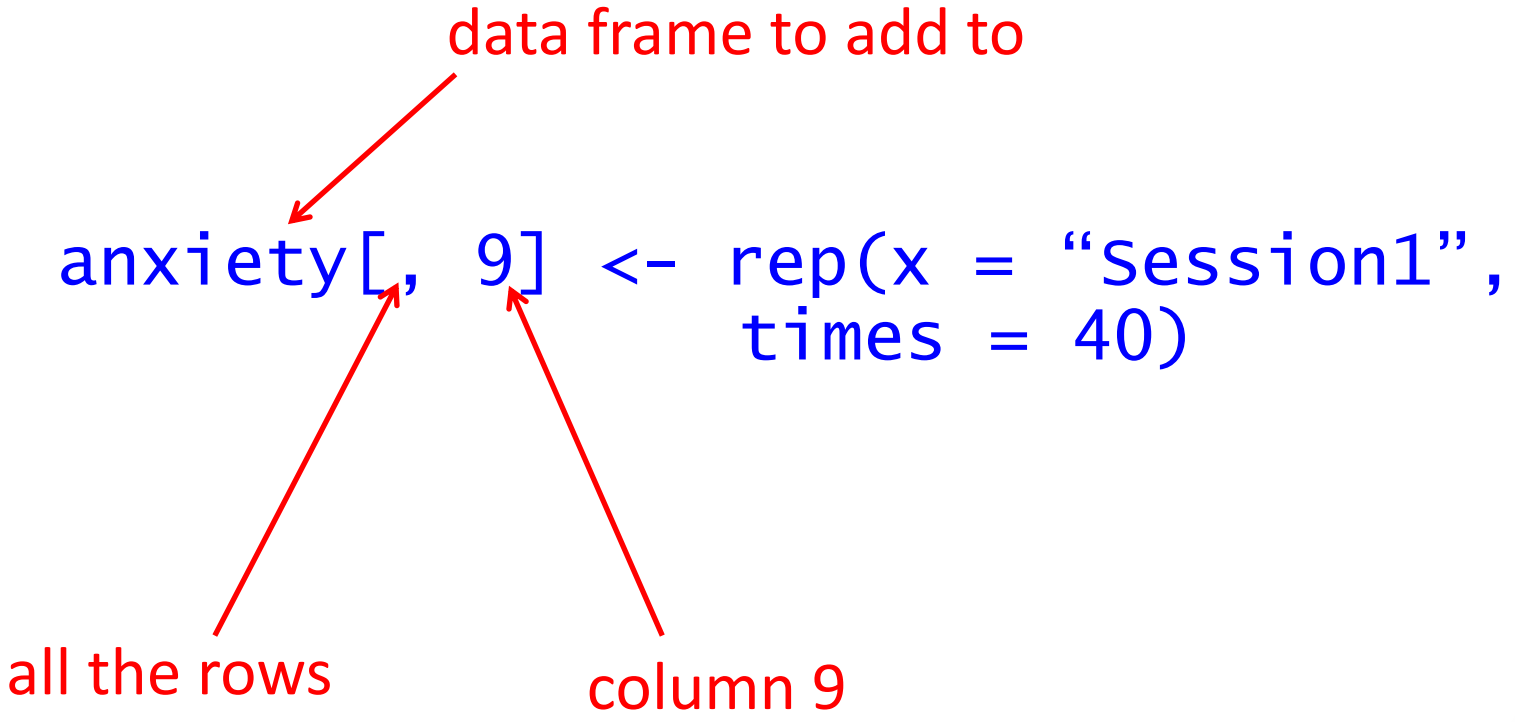
Why would you **filter (e.g. exclude)** data?

- Outliers
- Data file has lots of variables that are not of interest to you
- Etc.

Note: Adding/filtering NEVER changes your original .csv file!

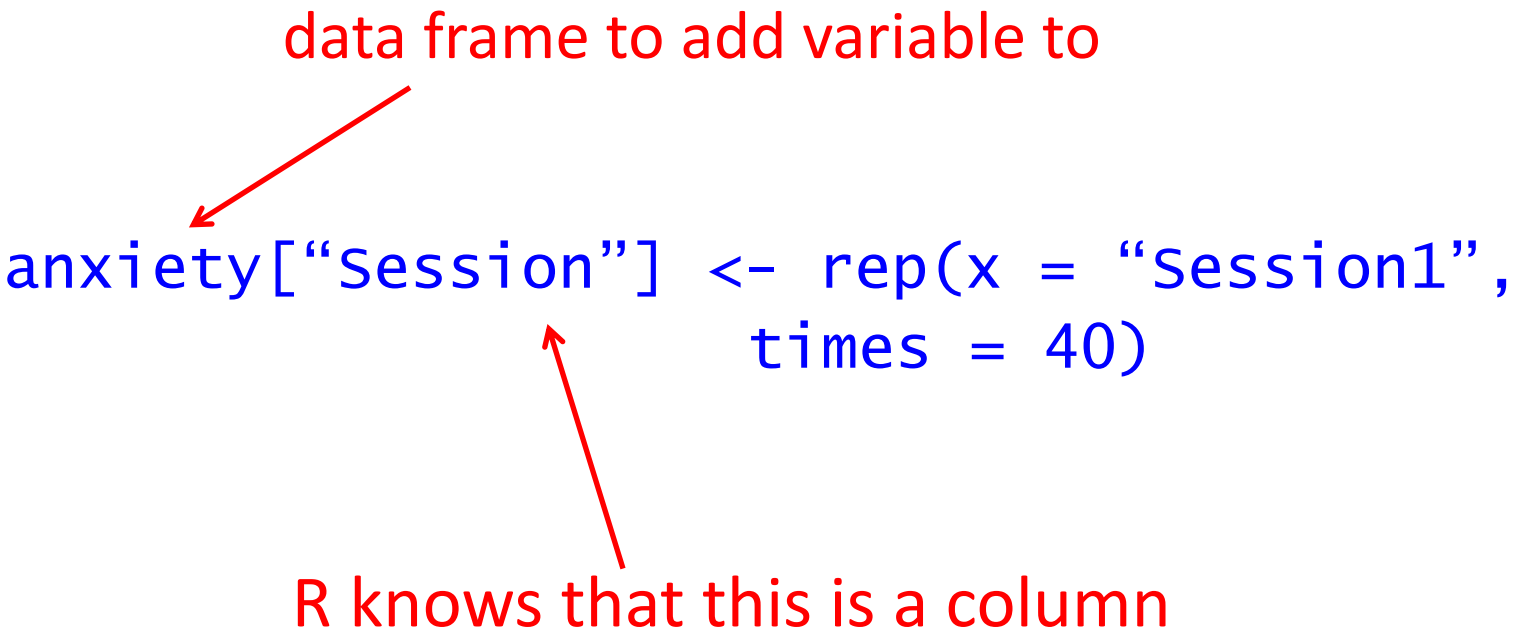
Remember Indexing?

Selecting variables to **ADD** by indexing:



What did R name this column?

Remember Indexing?

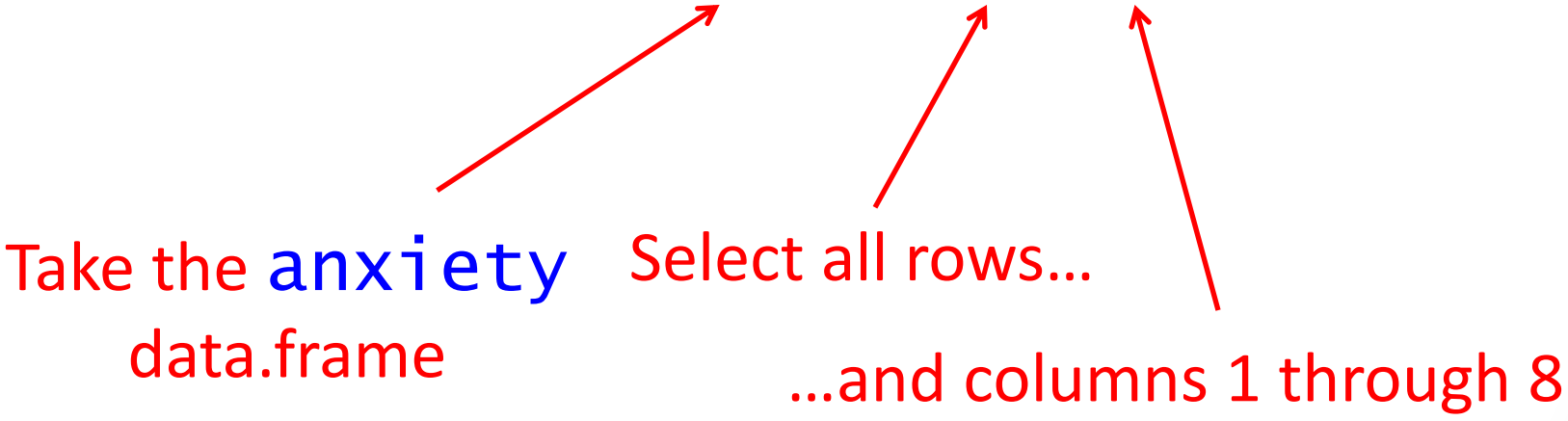


What did R name this column?

Remember Indexing?

Selecting variables to **REMOVE** by indexing:

```
anxiety2 <- anxiety[, 1:8]
```

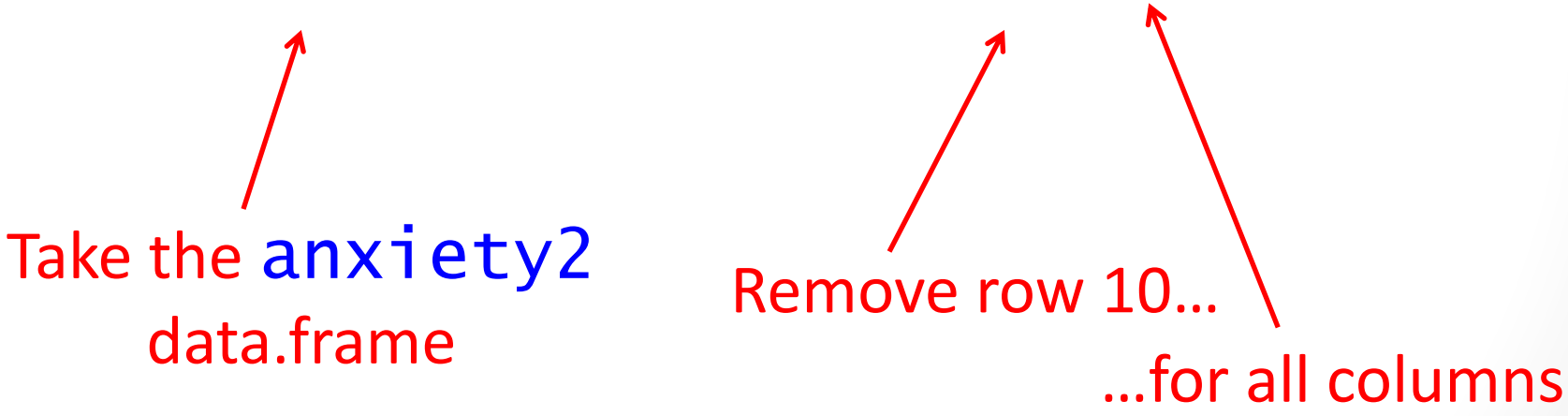


Did `anxiety` change at all? Why?

Remember Indexing?

Let's say Subject 10 is an outlier that we'd like to remove…

```
anxiety2 <- anxiety2[-10, ]
```



Did `anxiety2` change at all? Why?

Remember Subsetting?

Remove one subject:

```
filter(.data = anxiety,
          Subject_ID != “ID0001”)
```

Remember Subsetting?

Subsetting an entire group:

```
filter(.data = anxiety,
          Treatment_Group == “placebo”)

filter(.data = anxiety,
          Treatment_Group != “treatment”)
```

What are each of these function calls doing?

Cleaning Global Environment

Sometimes your Global Environment can fill up with stuff that you don't need. You can clean this!

One option: Delete EVERYTHING using the broom (we've done this several times)

Another option: Switch to GRID view, check boxes of individual objects you DON'T want, then press the broom button

Exercise

Clean the global environment so that the only thing remaining in the global environment is `anxiety`. Nothing else should be there.

Filter the last 2 columns of the `anxiety` data.frame. (Remove `V9` and `Session`)

Hint: `V9` is column 9; `Session` is column 10.

SUMMARIZING DATA

Summarizing Data

Before you start to run any real analyses, most of the time you want to get a feel for what's happening in your data

Summarizing Data

For really quick viewing of what you have: `describe()`

For count/frequency information: `table()`

Try:

```
describe(anxiety)
table(anxiety$Treatment_Group)
```

Note: We recommend `dplyr` and `tidyr` (both part of the "tidyverse").

PLOTTING

Plotting

R can plot without the help of any packages, but they're very ugly.

We will use the `ggplot2` package because it is much more flexible.

You loaded `ggplot2` already, so you should be ready to go!

Plotting

ggplot2 has the following structure:

```
ggplot(things that impact entire plot) +
    geom_something (things that
    impact just the something)
```

Plotting

A `geom_` normally means shape. What shapes do you want to use to represent your plot?

- `geom_histogram` (1 variable)
- `geom_density` (1 variable)
- `geom_boxplot` (1 variable)
- `geom_point` (2 variables, scatter plot)
- `geom_col` (2 variables, barplot)

Plotting

`ggplot()` and each `geom_()` can take on different aesthetics as an `aes()` argument.

What do you want your plot to look like? How can you make it pretty?

- Which variables are the `x-axis` and `y-axis?`
- `color` (should you color the plot by some variable?)
- `fill` (very similar to color, should you fill the plot in somehow – used for bar graphs, box plots)
- `shape` (represent groups using different shapes)
- `size` (represent groups using different sizes)

Plotting

- Usually `aes()` contains some information from the data.
- If the information isn't based on the data, it doesn't need to be inside an `aes()`

```
ggplot(data = anxiety, aes(x = Age, y = SES)) +
    geom_point()

ggplot(data = anxiety, aes(x = Age, y = SES)) +
    geom_point(color = "cornflowerblue"))

ggplot(data = anxiety, aes(x = Age, y = SES)) +
    geom_point(aes(color = Treatment_Group))
```

Reminder: Anxiety Data

Clinical trial for a new anxiety treatment

- Participants 18 to 50 years old
- Randomly assigned “treatment” or “placebo” group.
- Socioeconomic status (SES)
  - Scale: 0 (low SES) to 10 (high SES)
- Anxiety scores: **Self**-Reported and **Relative**-Reported
  - Scale: 0 (no anxiety) to 100 (severe anxiety)
- Stress
  - Scale: 0 (no stress) to 30 (high stress)
- Drug_Use
  - Scale: 0 (no drug use) to 50 (high drug use)

Exercise

Make a scatter plot of `Anxiety_SelfReport` (x-axis) against `Drug_Use` (y-axis).

Make the points of the scatter plot different `shapes` based on `Treatment_Group` (for example, placebo might be circles and treatment might be squares).

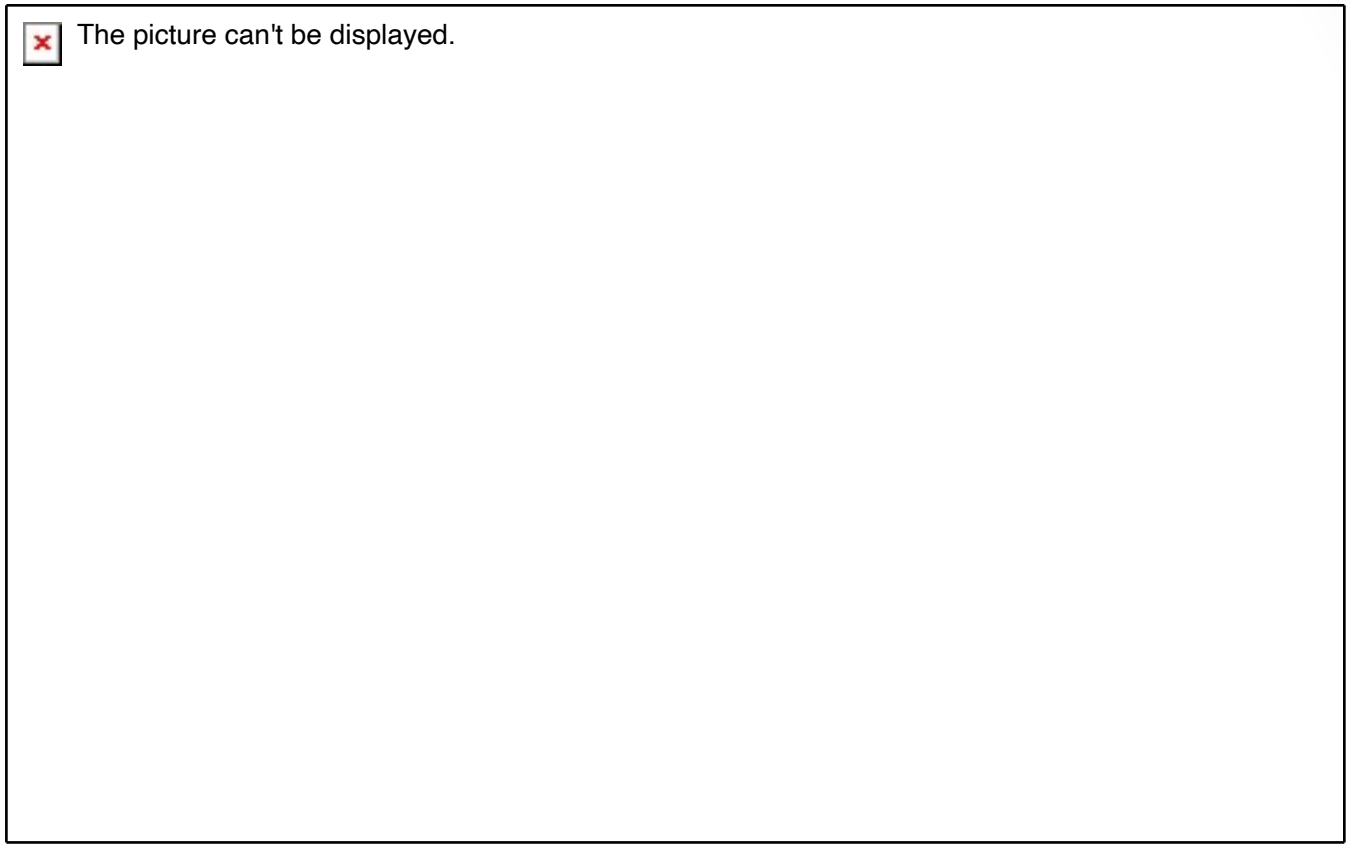
Make the `color` of the points different based on `Treatment_Group`.

Set the `size` of all points equal to 3

HINT: one of these…

- `geom_boxplot`
- `geom_point`
- `geom_col`

REMOVE POST-ITS!!



(and save!)

Finally……an actual analysis!!

There are thousands of different statistics you can compute in R.

There are thousands of different ways to compute the same statistic in R.

We will go few some of the basic statistics, as they relate to this data.frame.

The goal is to go through what you might actually do with real data!

Question 1

Are there any differences in demographics between the treatment and non-treatment groups?

1a: Is there a difference in Age between the treatment and placebo groups?

- Statistic: *t*-test
- Plot: Boxplot

Notes about formulas

You can often read the ~ (tilde) as “by” or as “predicted by”

Example:

`Age ~ Treatment_Group` **means…**

“Age by Treatment_Group”

“Is Age predicted by Treatment_Group?”

Question 1 - Exercise

Question 1b: Is there a difference in SES between the treatment groups?

Statistic: `t.test()`

Plot: `geom_boxplot()`

Make sure to `fill` by `Treatment_Group`

BONUS:

Set the `size` of the boxplot line thickness to 2

Set the `color` of the boxplot line to `purple`

Question 2

Does self-reported anxiety correlate with things that it should be correlated with?

2a: Does self-report anxiety correlate with relative-report anxiety?

- Statistic: Correlation
- Plot: Scatter Plot

Question 2 - Exercise

Question 2b: Does `Anxiety_SelfReport` correlate with `Stress`?

Statistic: `cor.test()`

Plot: `geom_point()`

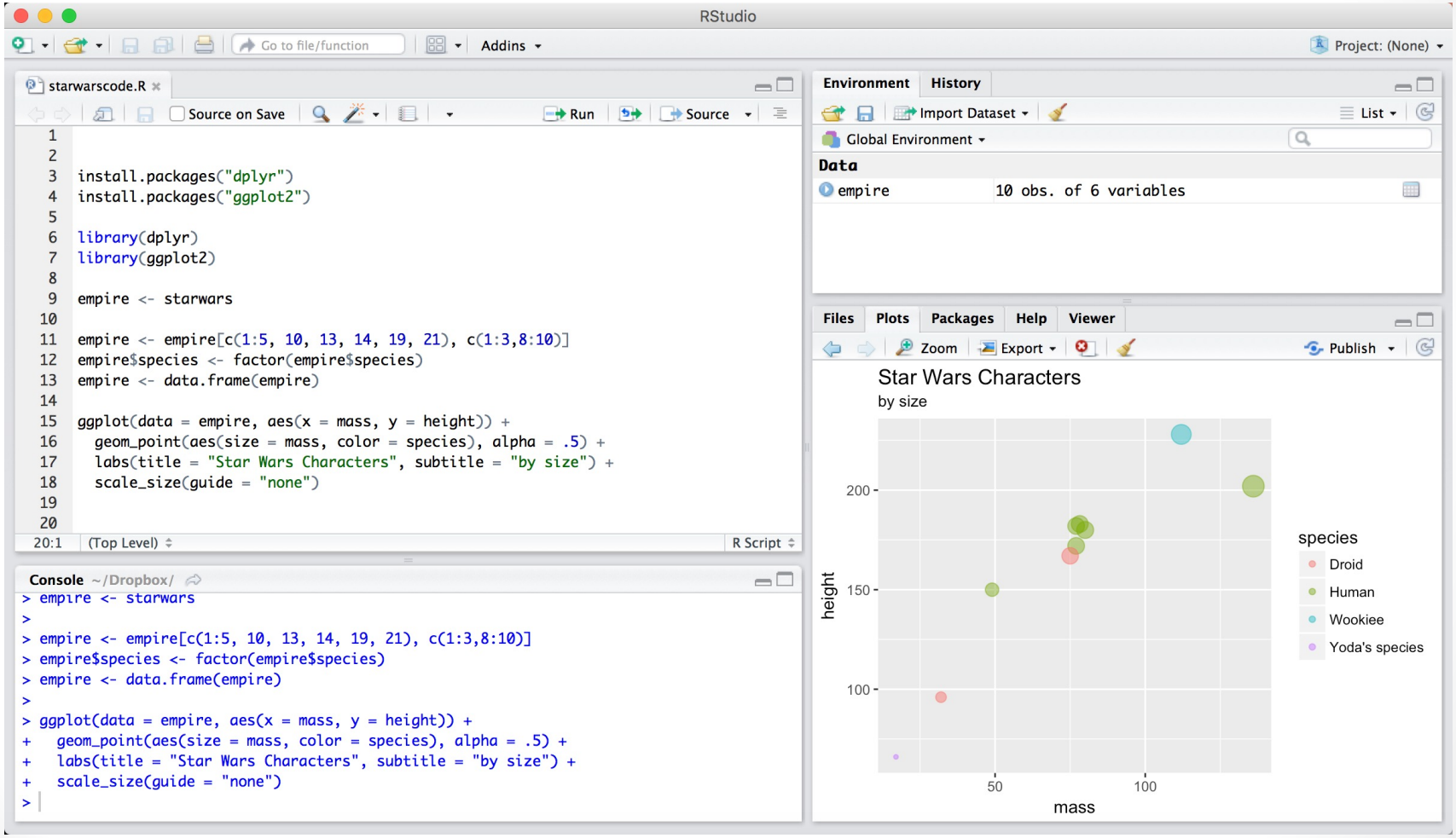
BONUS:

Set the `shape` of all the points to `21`

Set the `fill` the points with `green`

Set the `alpha` (transparency of the dots) to `0.4`

Remember this???



Other things you might need…

Data Transformation:

- Renaming columns
- Save the output

These are all things that you may one day find interesting!

Renaming Columns

```
colnames(data.frame)[columnNumber]
    <- “NewName”

colnames(anxiety)[5] <- “Session1”
colnames(anxiety)[6] <- “Session2”
```

About Saving Data…

It's very easy to export data.frames once you have manipulated them in R. You can use this to your advantage!

```
write.csv(x = R_object,
          file = "newfile.csv",
          row.names = FALSE)
```

R RESOURCES

- **Learning Statistics with R** – free, well-written book available in PDF format; integrates learning R with learning basic stats
  - https://health.adelaide.edu.au/psychology/ccs/teaching/lsr/
- **DataCamp** – online tutorials to teach R
  - https://www.datacamp.com/introduction-to-statistics
  - (also try Coursera and Code School)
- **Favorite websites for Reference:**
  - Quick-R
  - Cookbook for R
  - Institute for Digital Research and Education (UCLA)
  - Stack Overflow
- **swirl** package inside RStudio: learn R, in R!
- **Email us!**
  - Zoë Hawks – hawksz@wustl.edu
  - Kendra Smith – kendrasmith@wustl.edu
  - Emorie Beck – edbeck@wustl.edu
- **Google, Google, Google!!!!!**