Meet R

Dr. Zahid Asghar
Professor of Economics
zasghar@qau.edu.pk
zahedasghar
zahidasghar.com

Data Science

- **You go into data analysis with the tools you know, not the tools you need**
- The next 2-3 weeks are all about giving you the tools you need
  - Admittedly, a bit before you know what you need them *for*
- We will extend them as we learn specific models

R

- **Free** and **open source**
- A very large community
  - Written by statisticians for statistics
  - Most packages are written for `R` first
- Can handle virtually any data format
- Makes replication easy
- Can integrate into documents (with `R markdown`)
- R is a *language* so it can do *everything*
  - A good stepping stone to learning other languages like *Python*

Excel (or Stata) Can’t Do This

Code Output

```
ggplot(data = gapminder,
       aes(x = gdpPercap,
           y = lifeExp,
           color = continent))+
  geom_point(alpha=0.3)+
  geom_smooth(method = "lm")+
    scale_x_log10(breaks=c(1000,10000, 100000),
                  label=scales::dollar)+
    labs(x = "GDP/Capita",
         y = "Life Expectancy (Years)")+
  facet_wrap(~continent)+
  guides(color = F)+
  theme_light()
```

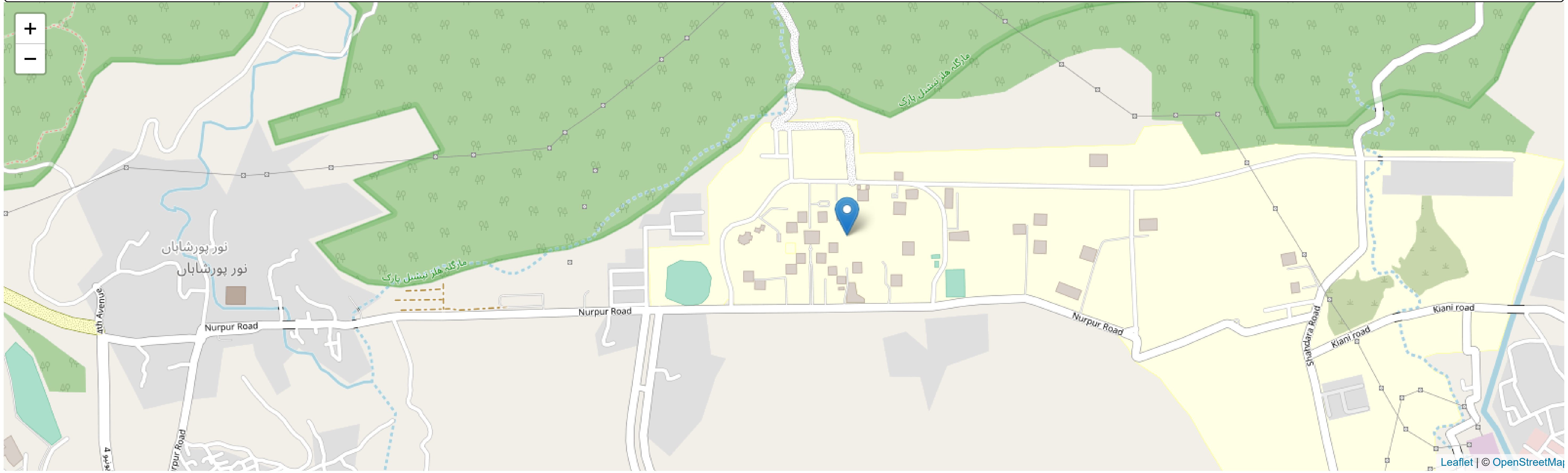
Or This

Input Output

The average GDP per capita is ` r dollar(mean(gapminder$gdpPercap)) ` with a standard deviation of ` r dollar(sd(gapminder$gdpPercap)) `.

Or This

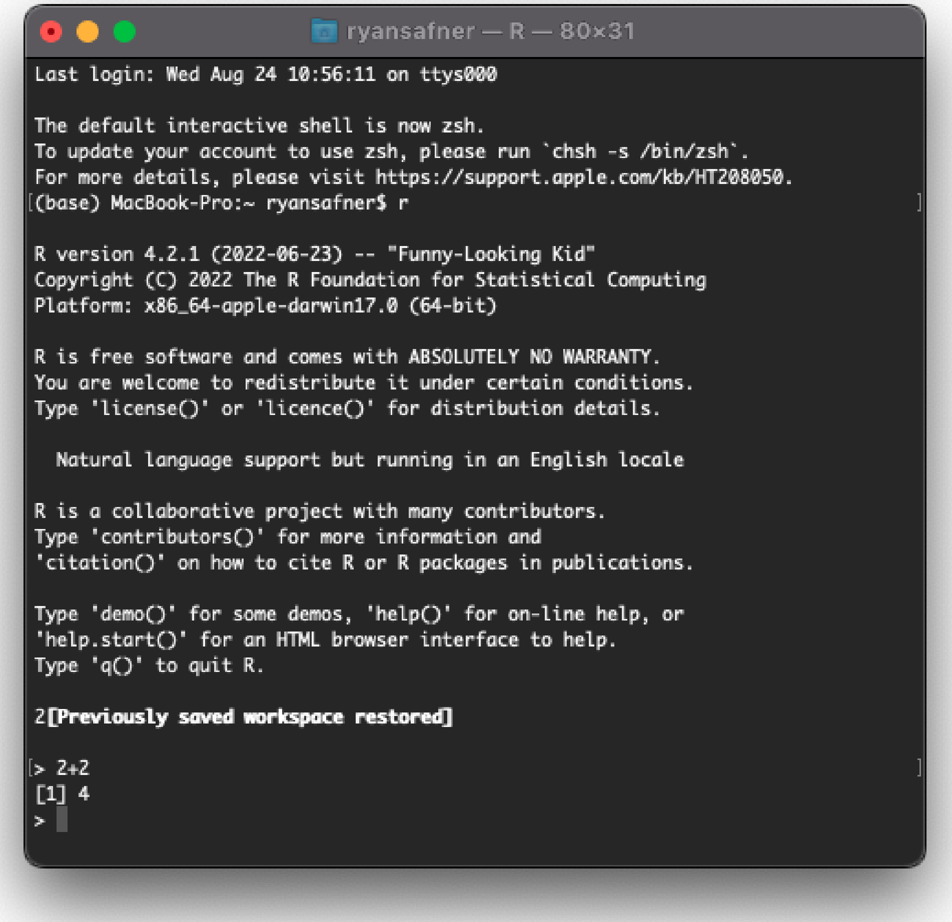
```r
library(leaflet)
leaflet() |>
  addTiles() |>
  addMarkers(lng = 73.1381, lat = 33.7476,
             popup = "Quaid-i-Azam University, Islamabad")
```



Meet R and R Studio

R and R Studio

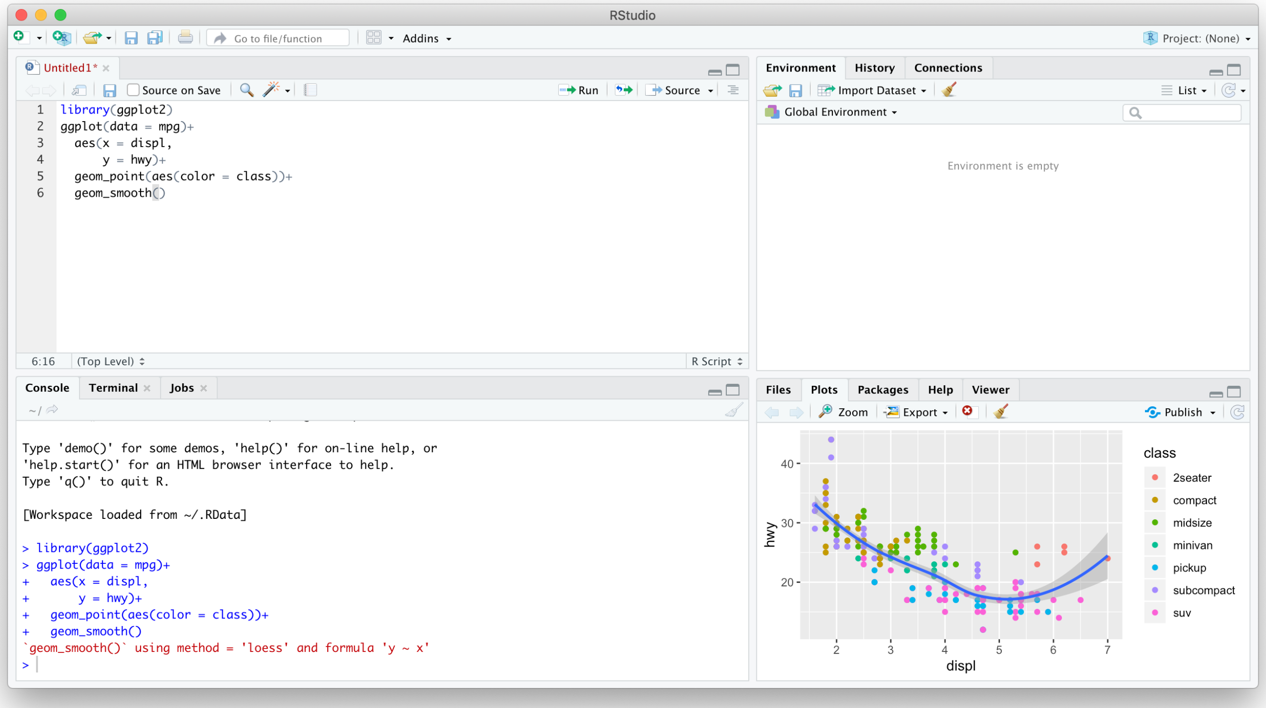
- R is the programming language that executes commands
- Could run this from your computer's shell
  - On Windows: **Command prompt**
  - On Mac/Linux: **Terminal**



```
Last login: Wed Aug 24 10:56:11 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) MacBook-Pro:~ ryansafner$ r

R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 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.

2[Previously saved workspace restored]

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

R and R Studio

- **R Studio**[1] is an **integrated development environment** (IDE) that makes your coding life a lot easier
  - Write code in scripts
  - Execute individual commands & scripts
  - Auto-complete, highlight syntax
  - View data, objects, and plots
  - Get help and documentation on commands and functions
  - Integrate code into documents with `Quarto`

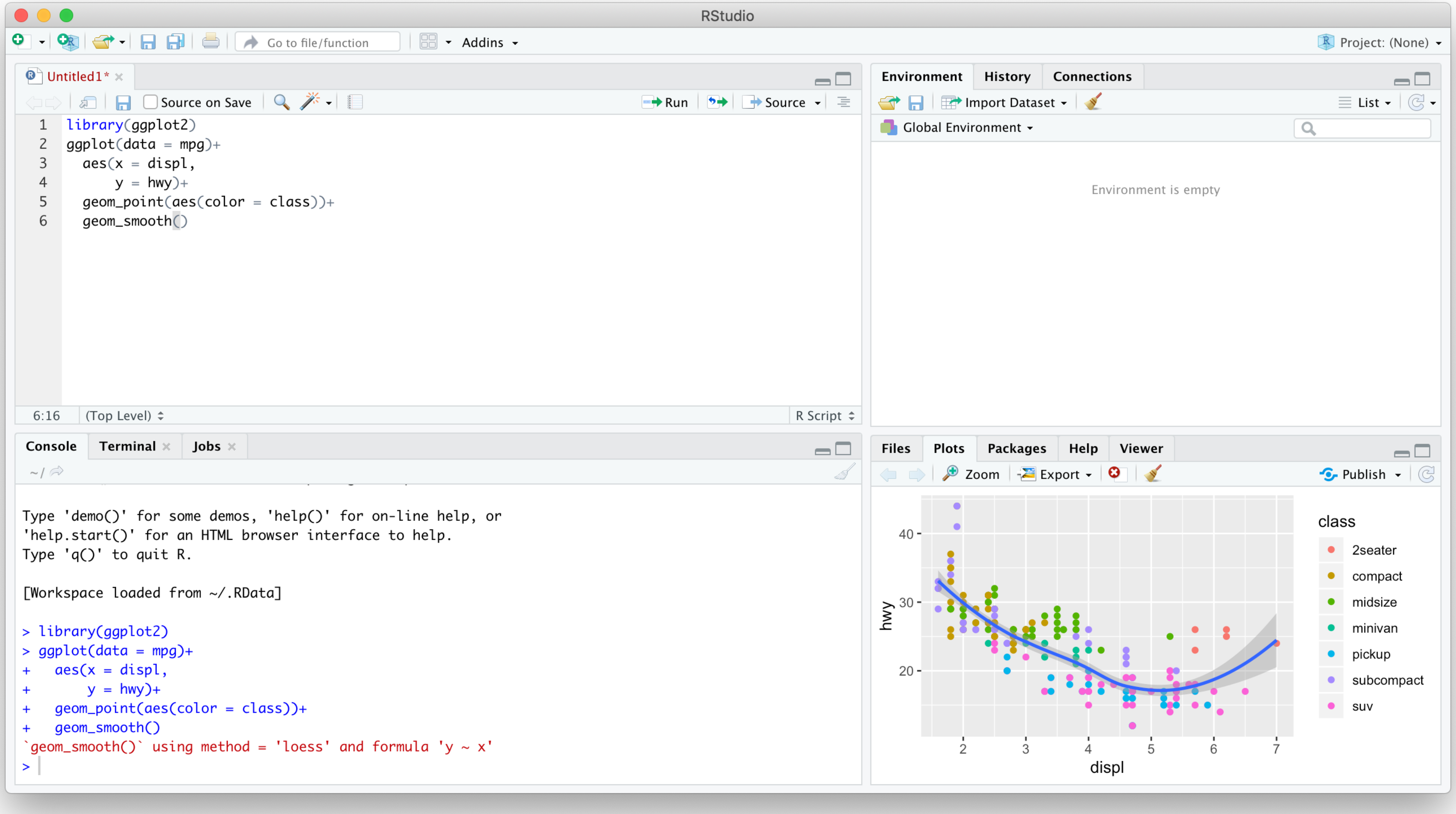


```r
library(ggplot2)
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()
```

```
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.

[Workspace loaded from ~/.RData]

> library(ggplot2)
> ggplot(data = mpg)+
+   aes(x = displ,
+       y = hwy)+
+   geom_point(aes(color = class))+
+   geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> 
```

[1] The company R Studio recently announced they will be rebranding later this fall as **Posit**

R Studio — Four Panes



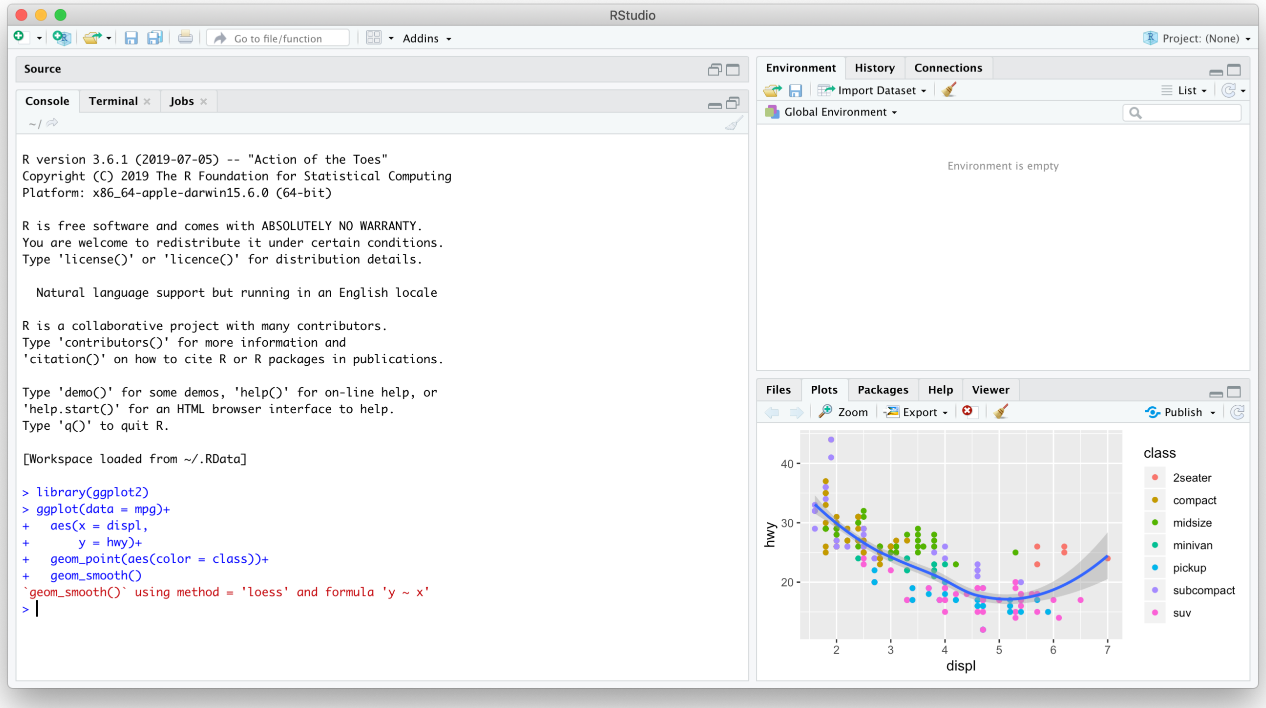
```r
library(ggplot2)
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()
```

Environment is empty

```
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.

[Workspace loaded from ~/.RData]

> library(ggplot2)
> ggplot(data = mpg)+
+   aes(x = displ,
+       y = hwy)+
+   geom_point(aes(color = class))+
+   geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```

Ways to Use R Studio: Using the Console

- Type individual commands into the console pane (bottom left)
- Great for testing individual commands to see what happens
- Not saved! Not reproducible! Not recommended!

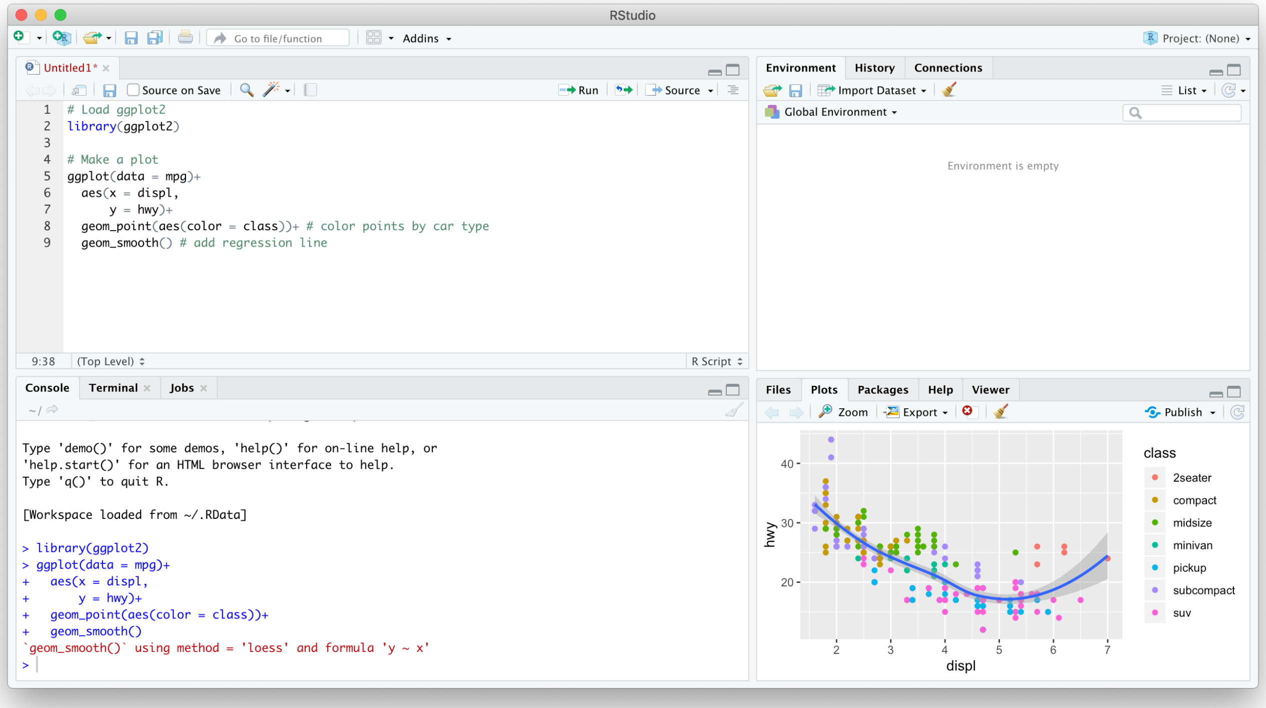


```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

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.

[Workspace loaded from ~/.RData]

> library(ggplot2)
> ggplot(data = mpg)+
+   aes(x = displ,
+       y = hwy)+
+   geom_point(aes(color = class))+
+   geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> 
```

Zahid Asghar

Ways to Use R Studio: Writing a `.R` Script

- Source pane is a text-editor
- Make `.R` files: all input commands in a single script
- Comment with `#`
- Can run any or all of script at once
- Can save, reproduce, and send to others!



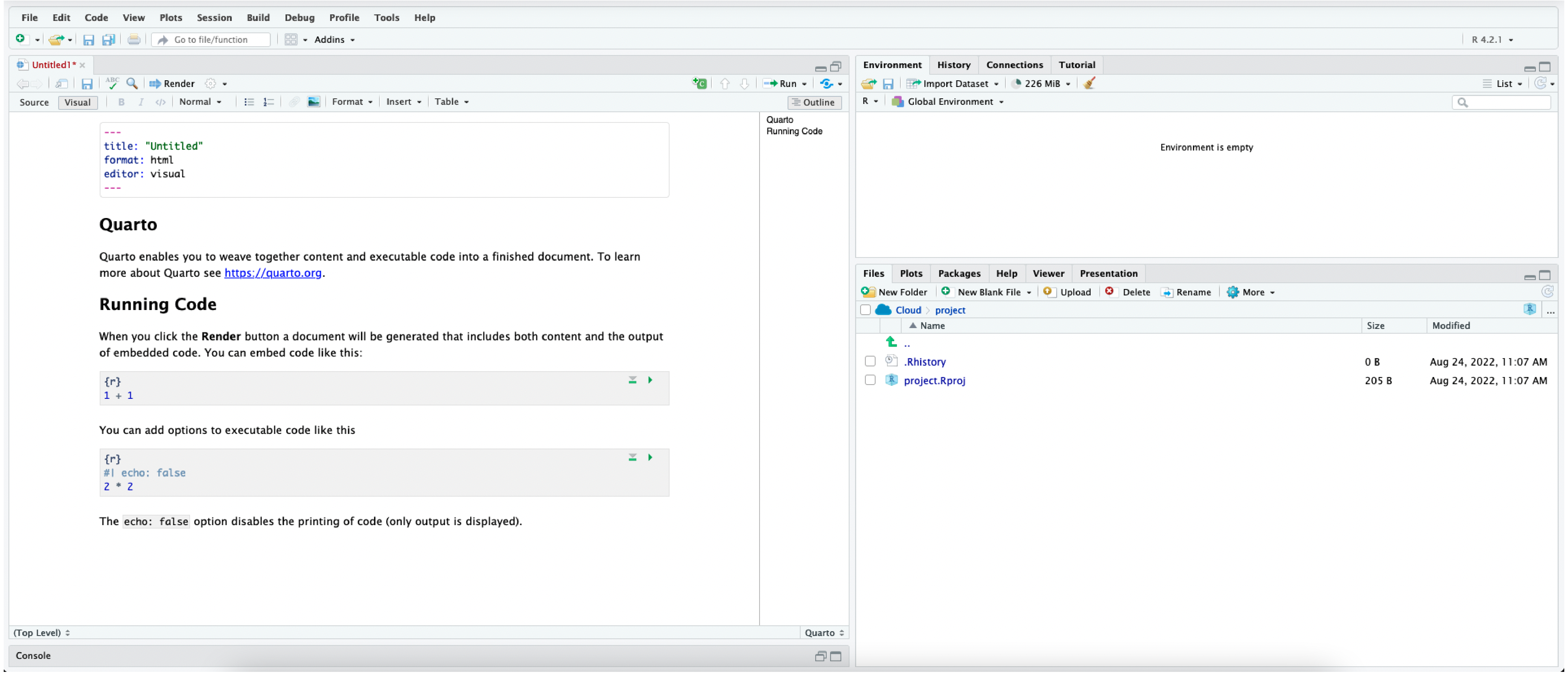
```r
# Load ggplot2
library(ggplot2)

# Make a plot
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+ # color points by car type
  geom_smooth() # add regression line
```

```
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.

[Workspace loaded from ~/.RData]

> library(ggplot2)
> ggplot(data = mpg)+
+   aes(x = displ,
+       y = hwy)+
+   geom_point(aes(color = class))+
+   geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> 
```

Environment is empty

Zahid Asghar

Ways to Use R Studio: Quarto Documents



```
---
title: "Untitled"
format: html
editor: visual
---
```

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
{r}
1 + 1
```

You can add options to executable code like this

```
{r}
#| echo: false
2 * 2
```

The `echo: false` option disables the printing of code (only output is displayed).

Environment is empty

Name	Size	Modified
..		
.Rhistory	0 B	Aug 24, 2022, 11:07 AM
project.Rproj	205 B	Aug 24, 2022, 11:07 AM

Zahid Asghar

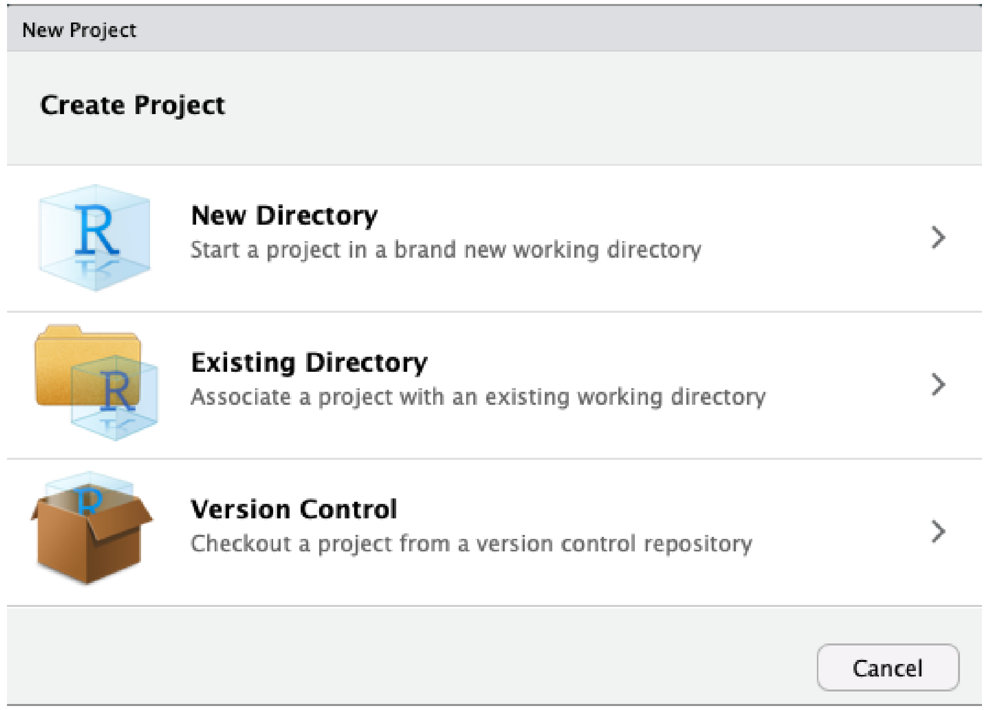
Getting to Know Your Computer

- R assumes a default (often inconvenient) **“working directory”** on your computer
  - The first place it looks to `open` or `save` files
- Find out where R this is with `getwd()`
- Change it with `setwd(path/to/folder)`[1]

[1] Note the path is OS-specific. For Windows it might be `C:/Documents/`. For Mac it is often your username folder

Avoid this Hassle with R Projects

- A `R Project` is a way of systematically organizing your `R` history, working directory, and related files in a single, self-contained directory
- Can easily be sent to others who can reproduce your work easily
- Connects well with version control software like GitHub
- Can open multiple projects in multiple windows



New Project

Create Project

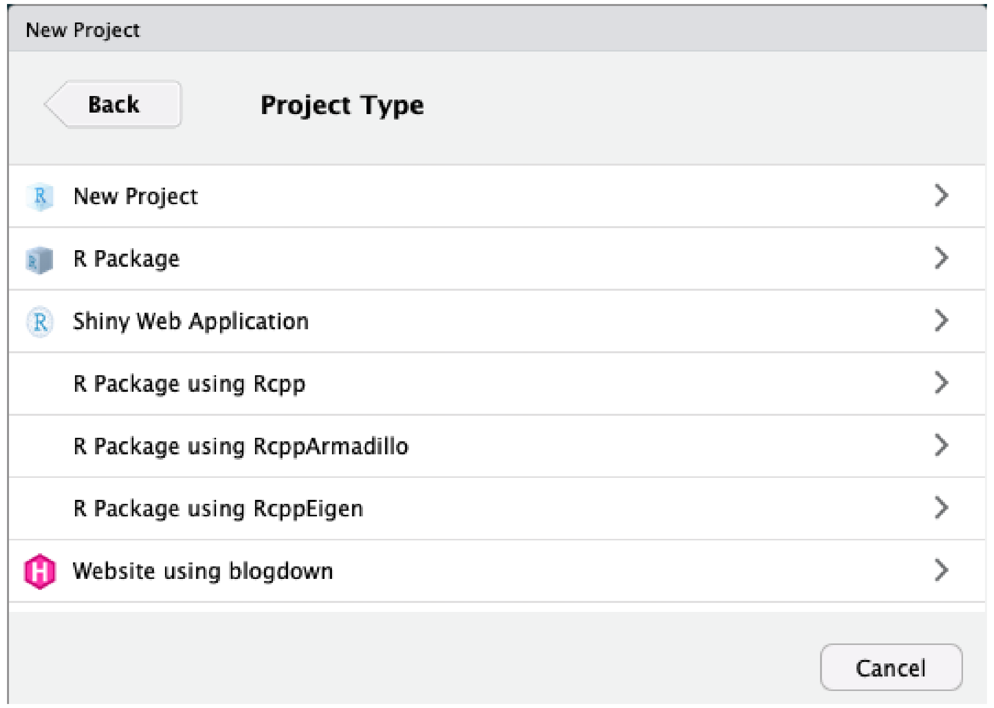
New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

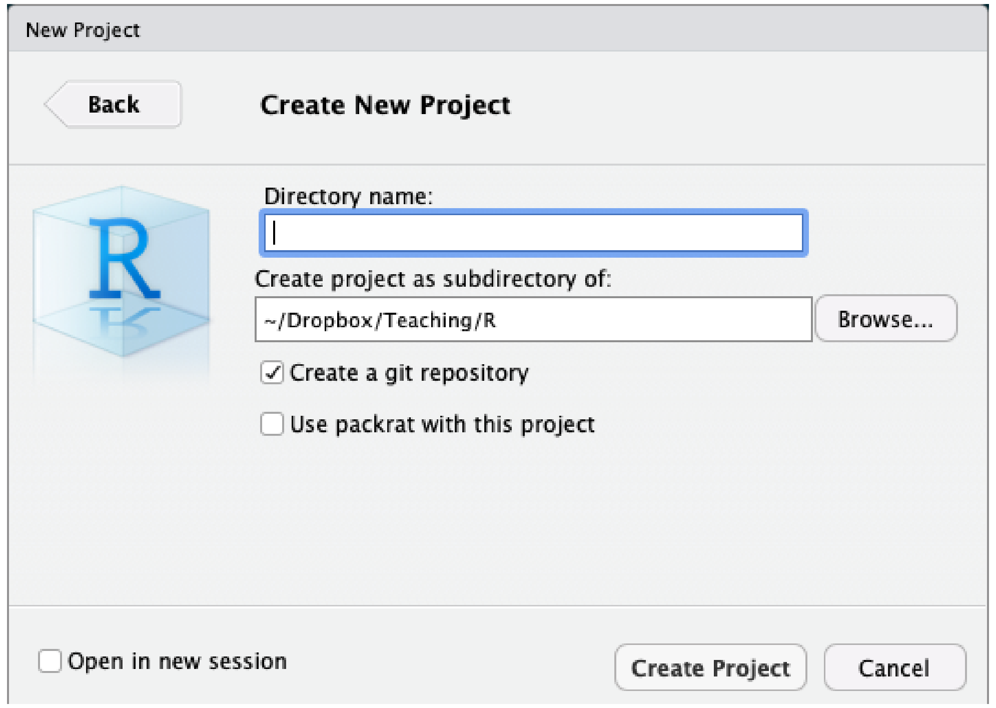
Avoid this Hassle with R Projects

- In almost all cases, you simply want a `New Project`
- For more advanced uses, your project can be an `R Package` or a `Shiny Web Application`
- If you have other packages that create templates installed (as I do, in the previous image), they will also show up as options



Avoid this Hassle with R Projects

- Enter a name for the project in the top field
  - Also creates a folder on your computer with the name you enter into the field
- Choose the location of the folder on your computer
- Depending on if you have other packages or utilities installed (such as `git`, see below!), there may be additional options, do not check them unless you know what you are doing
- Bottom left checkbox allows you to open a new instance (window) of `R` just for this project (and keep existing windows open)



Directory name:

Create project as subdirectory of: ~/Dropbox/Teaching/R

☑ Create a git repository

☐ Use packrat with this project

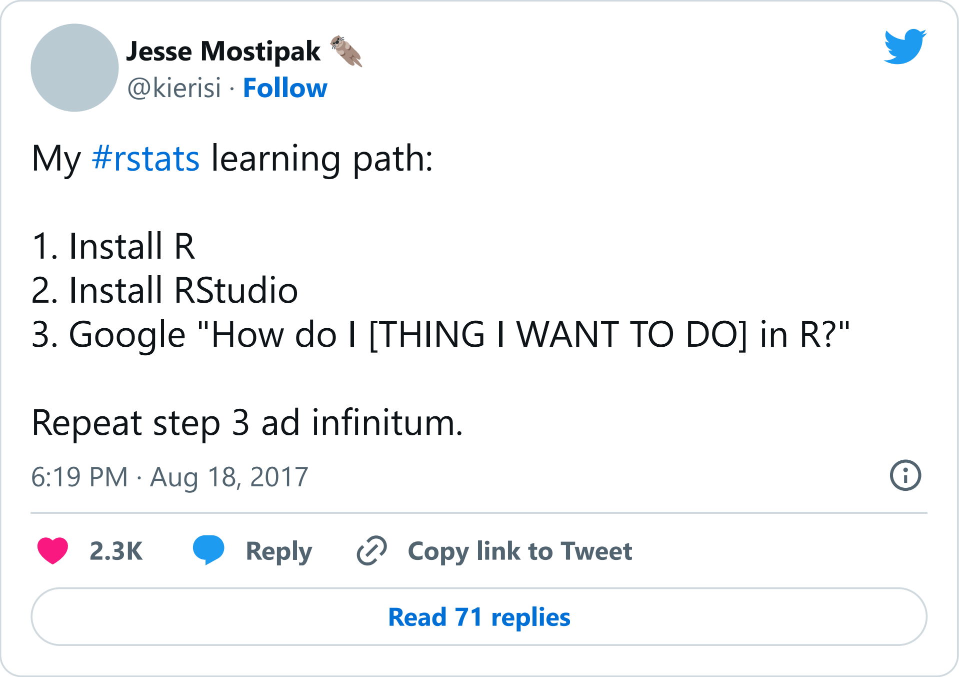
☐ Open in new session

An Intro to Coding

Learning...

- **You don’t “*learn R*”, you learn *how to do things in R***
- In order to do learn this, you need to learn *how to search for what you want to do*

Learning...



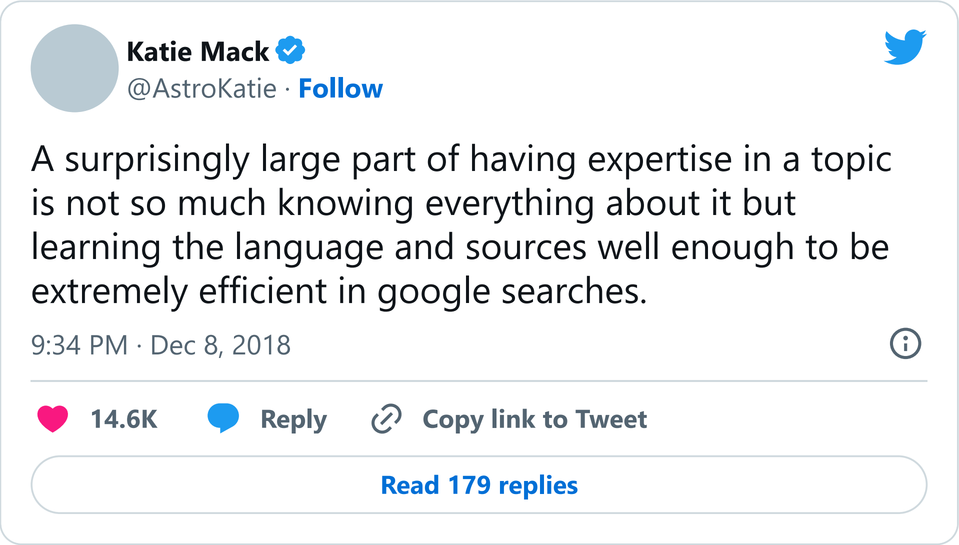
Jesse Mostipak 🦦
@kierisi · Follow

My #rstats learning path:

1. Install R
2. Install RStudio
3. Google "How do I [THING I WANT TO DO] in R?"

Repeat step 3 ad infinitum.

6:19 PM · Aug 18, 2017

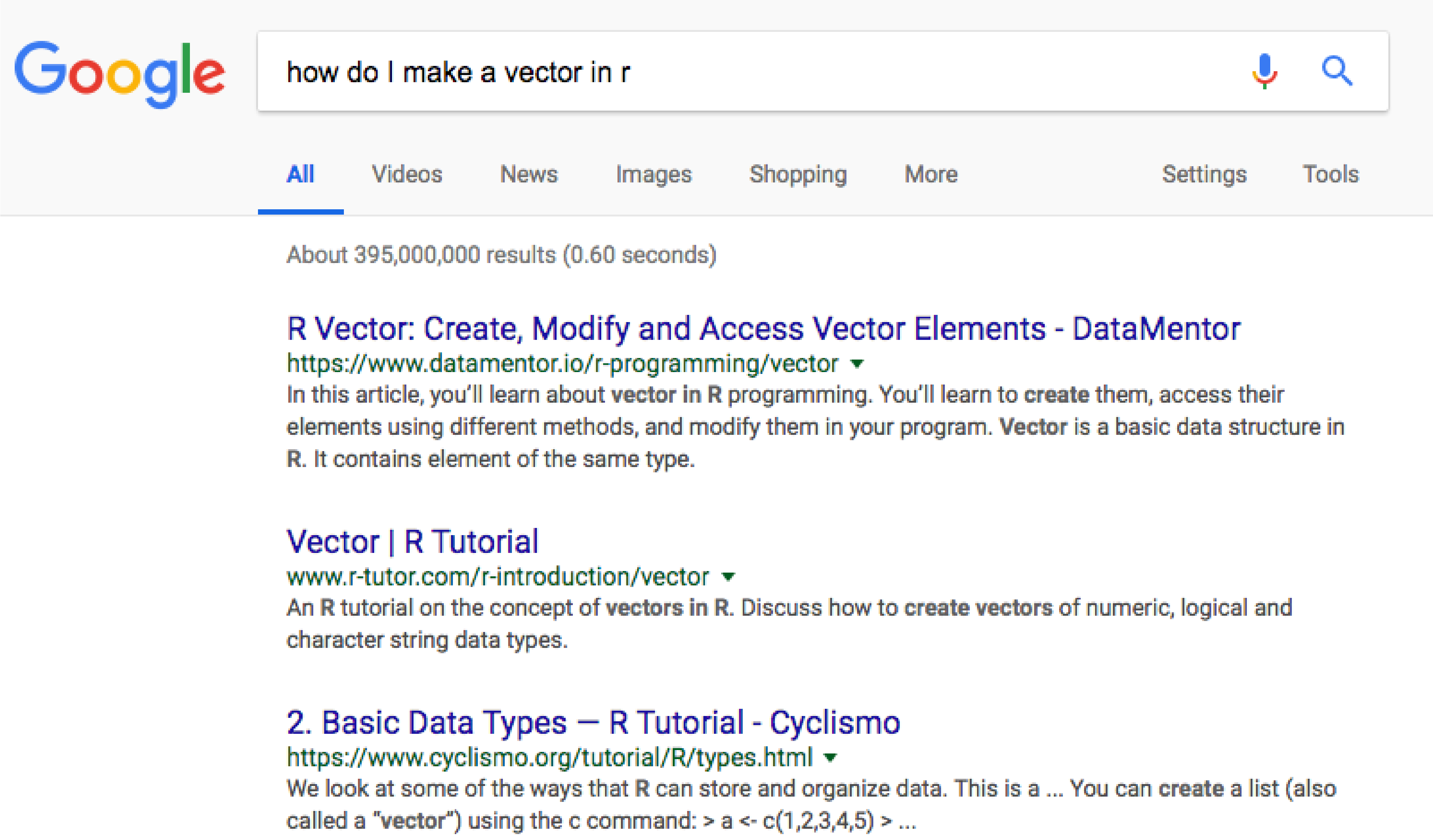


Katie Mack
@AstroKatie · Follow

A surprisingly large part of having expertise in a topic is not so much knowing everything about it but learning the language and sources well enough to be extremely efficient in google searches.

9:34 PM · Dec 8, 2018

Say Hello To My Little Friend



how do I make a vector in r

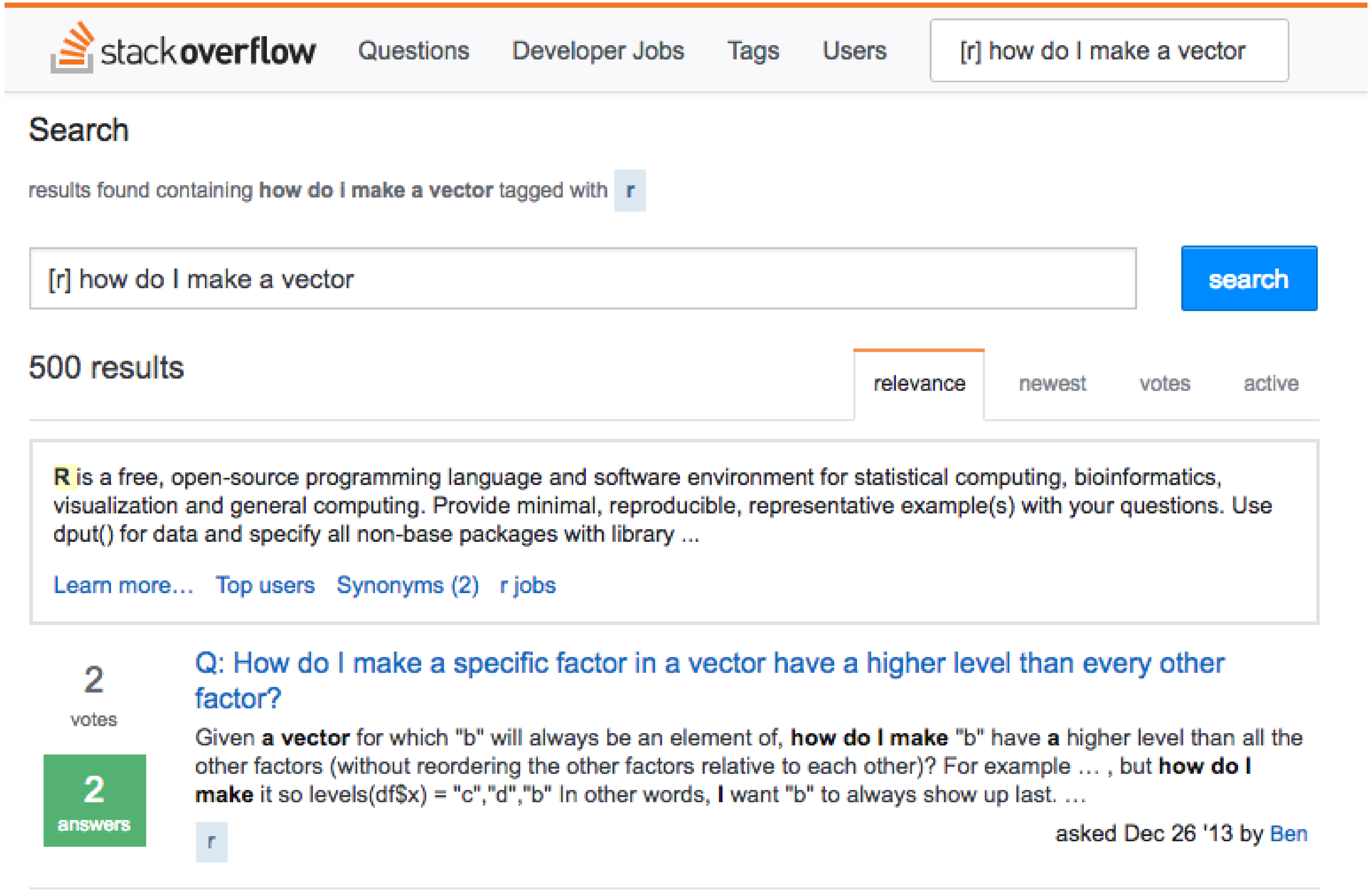
About 395,000,000 results (0.60 seconds)

R Vector: Create, Modify and Access Vector Elements - DataMentor
https://www.datamentor.io/r-programming/vector ▾
In this article, you'll learn about **vector in R** programming. You'll learn to **create** them, access their elements using different methods, and modify them in your program. **Vector** is a basic data structure in **R**. It contains element of the same type.

Vector | R Tutorial
www.r-tutor.com/r-introduction/vector ▾
An **R** tutorial on the concept of **vectors in R**. Discuss how to **create vectors** of numeric, logical and character string data types.

2. Basic Data Types — R Tutorial - Cyclismo
https://www.cyclismo.org/tutorial/R/types.html ▾
We look at some of the ways that **R** can store and organize data. This is a ... You can **create** a list (also called a "**vector**") using the c command: > a <- c(1,2,3,4,5) > ...

Say Hello to My Better Friend



Search

results found containing **how do I make a vector** tagged with **r**

[r] how do I make a vector

500 results

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, visualization and general computing. Provide minimal, reproducible, representative example(s) with your questions. Use dput() for data and specify all non-base packages with library ...

Learn more… Top users Synonyms (2) r jobs

2 votes

2 answers

Q: How do I make a specific factor in a vector have a higher level than every other factor?

Given **a vector** for which "b" will always be an element of, **how do I make** "b" have **a** higher level than all the other factors (without reordering the other factors relative to each other)? For example … , but **how do I make** it so levels(df$x) = "c","d","b" In other words, **I** want "b" to always show up last. …

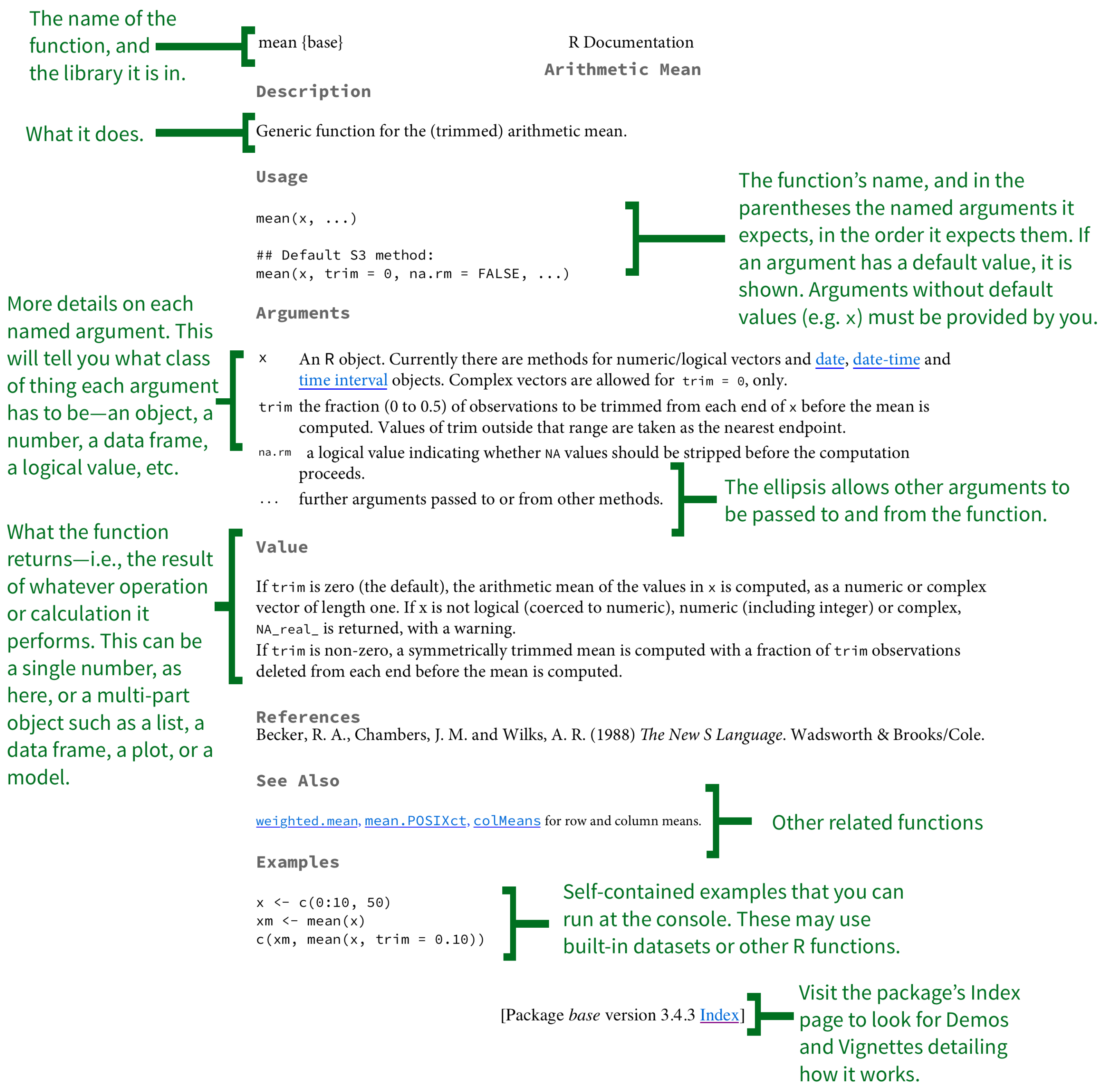
r

asked Dec 26 '13 by Ben

R Is Helpful Too!

- Type `help(function_name)` or `?(function_name)` to get documentation on a function

```r
help(mean)

?mean() # does the same thing
```



The name of the function, and the library it is in.

mean {base} R Documentation

Arithmetic Mean

Description

What it does.

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

The function's name, and in the parentheses the named arguments it expects, in the order it expects them. If an argument has a default value, it is shown. Arguments without default values (e.g. x) must be provided by you.

Arguments

More details on each named argument. This will tell you what class of thing each argument has to be—an object, a number, a data frame, a logical value, etc.

`x` An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for `trim = 0`, only.

`trim` the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.

`na.rm` a logical value indicating whether `NA` values should be stripped before the computation proceeds.

`...` further arguments passed to or from other methods.

The ellipsis allows other arguments to be passed to and from the function.

Value

What the function returns—i.e., the result of whatever operation or calculation it performs. This can be a single number, as here, or a multi-part object such as a list, a data frame, a plot, or a model.

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.
If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`weighted.mean`, `mean.POSIXct`, `colMeans` for row and column means.

Other related functions

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.

[Package *base* version 3.4.3 Index]

Visit the package's Index page to look for Demos and Vignettes detailing how it works.

From Kieran Healy, Data Visualization.

Tips for Writing Code

- Comment, comment, comment!
- The hashtag # starts a comment, R will ignore everything on the rest of that line
- Save often!
  - Write scripts that save the commands that did what you wanted (and comment them!)
  - Better yet, use a version control system like Git (I may cover this later)

Style and Naming

- Once we start writing longer blocks of code, it helps to have a consistent (and human-readable!) style
- I follow this style guide (you are not required to)[1]
- Naming objects and files will become important
  - DO NOT USE SPACES! You’ve seen seen webpages intended to be called `my webpage in html` turned into `http://my%20webpage%20in%20html.html`

```
i_use_underscores
some.people.use.snake.case
othersUseCamelCase
```

[1] Consider your folders on your computer as well

Simple Commands

- You’ll have to get used to the fact that you are coding in commands to execute
- Start with the easiest: simple math operators and calculations:

```
> 2+2
```

```
[1] 4
```

- Note that R will ask for **input** with `>` and give you **output** starting with `[1]`

Simple Commands

- We can start using more fancy commands

```r
2^3
```

```
[1] 8
```

```r
sqrt(25)
```

```
[1] 5
```

```r
log(6)
```

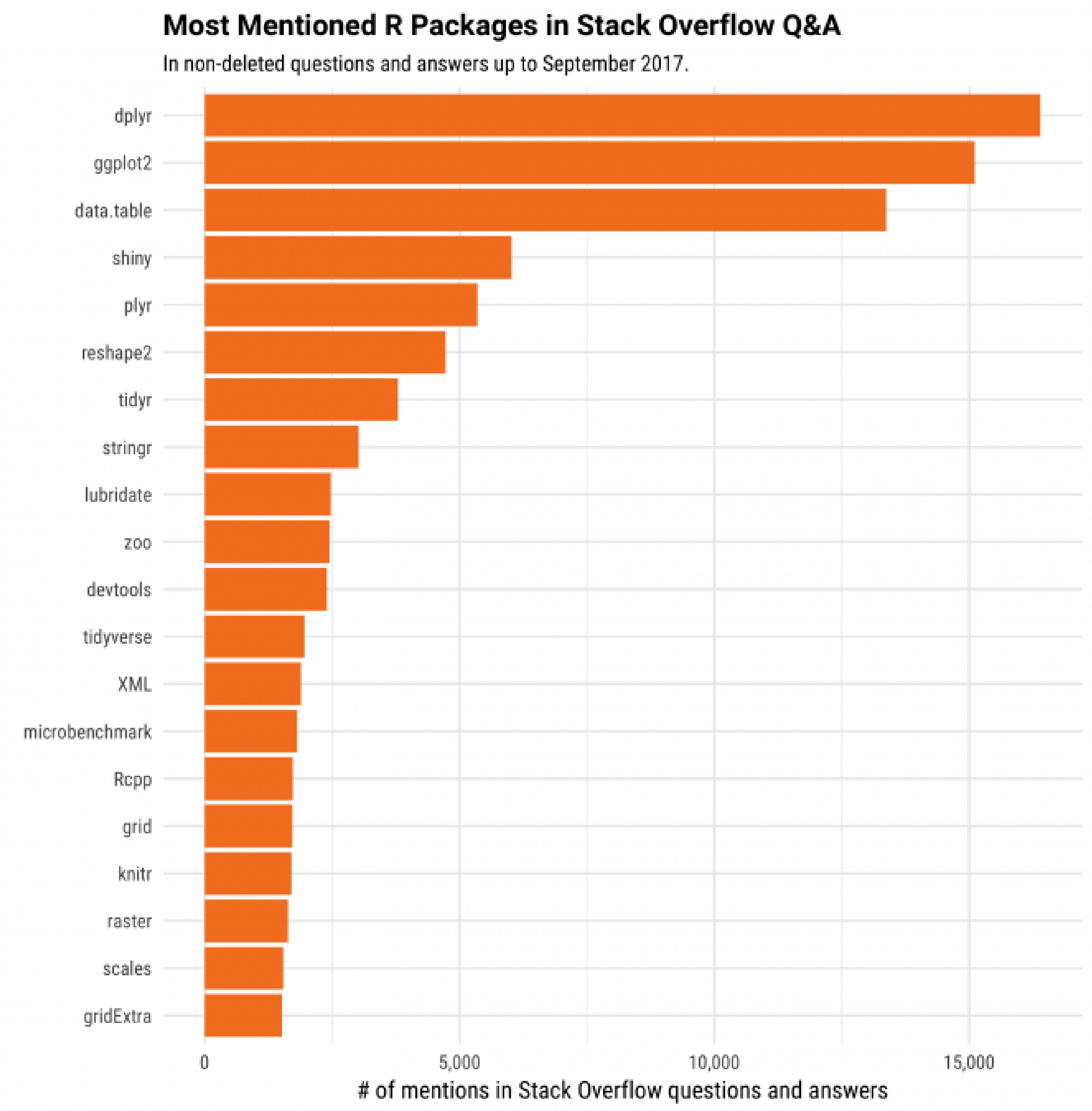
```
[1] 1.791759
```

```r
pi/2
```

```
[1] 1.570796
```

Packages and Libraries

- Since R is open source, users contribute **packages**
  - Really it's just users writing custom functions and saving them for others to use
- Load packages with `library()`[1]
  - e.g. `library("ggplot2")`
- If you don't have a package, you must first `install.packages()`[2]
  - e.g. `install.packages("ggplot2")`



[1] You can run a function from a previously-installed package without loading it with the syntax `package_name::function_name()`.

[2] Yes, note the plural, even if it's just for one package.

Objects & Functions

- R is an **object-oriented** programming language, meaning you will always be:

1. creating `objects`
   - assign values to an object with `=` (or `<-`)[1]
2. running `functions` on `objects`
   - syntax: `my_function(my_object)`

```r
# make an object
my_object = c(1,2,3,4,5)

# look at it
my_object
```

```
[1] 1 2 3 4 5
```

```r
# find the sum
sum(my_object)
```

```
[1] 15
```

```r
# find the mean
mean(my_object)
```

```
[1] 3
```

[1] You can read this as "gets"

More About Functions

- Functions have **arguments**, the input(s)
- Some functions may have multiple arguments
- The argument of a function can be *another* function!

```r
# find the sd
sd(my_object)
```

```
[1] 1.581139
```

```r
# round everything in my object to two decimals
round(my_object,2)
```

```
[1] 1 2 3 4 5
```

```r
# round the sd to two decimals
round(sd(my_object),2)
```

```
[1] 1.58
```

Types of R Objects

Numeric

- `numeric` objects are just numbers[1]
- Can be mathematically manipulated

```r
x <- 2
y <- 3
x+y
```

```
[1] 5
```

```r
x*y
```

```
[1] 6
```

[1] If you want to get technical, R calls these `integer` (for whole numbers) or `double` if there are decimal values

Character

- `character` objects are “strings” of text **contained inside quote marks**
- Can contain spaces, so long as contained within quote marks

```r
name <- "Zahid Asghar"
address <- "Quaid-i-Azam University"

name
```

```
[1] "Zahid Asghar"
```

```r
address
```

```
[1] "Quaid-i-Azam University"
```

Logical

- `logical` objects are **boolean/binary** `TRUE` or `FALSE` indicators[1]
- Used a lot to evaluate **conditionals**:
  - `>`, `<`: greater than, less than
  - `>=`, `<=`: greater than or equal to, less than or equal to
  - `==`, `!=`: is equal to, is not equal to[2]
  - `&in&` : is a member of the set of $(\in)$
  - `&`: "AND"
  - `|`: "OR"

```r
z = 10 # set z equal to 10

z==10 # test is z equal to 10?
```

```
[1] TRUE
```

```r
"red"=="blue" # test is red equal to blue?
```

```
[1] FALSE
```

```r
z > 1 & z < 12 # test is z > 1 AND < 12?
```

```
[1] TRUE
```

```r
z <= 1 | z==10 # test is z >= 1 OR equal to 10?
```

```
[1] TRUE
```

[1] Technically, under the hood, `R` is actually storing them as `numeric`: `1 = TRUE`, `0 = FALSE`!

[2] One `=` *assigns* a value (like `<-`). Two `==` *evaluates* a conditional statement!

Factor

- `factor` objects contain **categorical** data - membership in mutually exclusive groups
- Look like `character` strings, behave more like `logical`s, but with *more than two options*

```
 [1] junior    senior    freshman  freshman  sophomore sophomore junior
 [8] junior    junior    senior
Levels: freshman sophomore junior senior
```

- We’ll make much more extensive use of them later

```
 [1] junior    senior    freshman  freshman  sophomore sophomore junior
 [8] junior    junior    senior
Levels: freshman < sophomore < junior < senior
```

Data Structures

Vectors

- `vector` the simplest type of object, just a collection of elements
    - All elements must be the same data type!
- Make a vector using the **combine/concatenate** `c()` function

```r
# create a vector named vec
vec <- c(1,"orange", 83.5, pi)

# look at vec
vec
```

```
[1] "1"                    "orange"               "83.5"
"3.14159265358979"
```

Dataframes I

- `data.frame` or `tibble`: what we'll always be using; think like a **"spreadsheet"**:
  - Each **column** is a vector (variable) of data all the same type
  - Each **row** is an observation (pair of values for all variables)

```r
library(ggplot2)
diamonds
```

```
# A tibble: 53,940 × 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
# … with 53,930 more rows
```

Dataframes II

- Dataframes are really just combinations of (column) vectors
- You can make data frames by combinining named vectors with `data.frame()` or creating each column/vector in each argument

```
# make two vectors
fruits = c("apple","orange","pear","kiwi","pineapp
numbers = c(3.3,2.0,6.1,7.5,4.2)

# combine into dataframe
df = data.frame(fruits,numbers)

# do it all in one step (note the = instead of <-)
df = data.frame(fruits=c("apple","orange","pear","
               numbers=c(3.3,2.0,6.1,7.5,4.2))

# look at it
df
```

```
     fruits numbers
1     apple     3.3
2    orange     2.0
3      pear     6.1
4      kiwi     7.5
5 pineapple     4.2
```

Working with Objects

Objects: Storing, Viewing, and Overwriting

- We want to store things in objects to run functions on them later
- Recall, any object is created with the assignment operator `=` or `<-`

```r
my_vector = c(1,2,3,4,5)
```

- R will not give any output after an assignment

Objects: Storing, Viewing, and Overwriting

- *View* an object (and list its contents) by typing its name

```
my_vector
```

```
[1] 1 2 3 4 5
```

- objects maintain their values until they are assigned different values that will *overwrite* the object

```
my_vector = c(2,7,9,1,5)
my_vector
```

```
[1] 2 7 9 1 5
```

Objects: Checking and Changing Classes

- Check what type of object something is with `class()`

```r
class("six")
```

```
[1] "character"
```

```r
class(6)
```

```
[1] "numeric"
```

- Can also use logical tests of `is.()`

```r
is.numeric("six")
```

```
[1] FALSE
```

```r
is.character("six")
```

```
[1] TRUE
```

Objects: Checking and Changing Classes

- Convert objects from one class to another with `as.object_class()`
  - Pay attention: you can’t convert non-numbers to `numeric`, etc!

```r
as.character(6)
```

```
[1] "6"
```

```r
as.numeric("six")
```

```
[1] NA
```

Objects: Different Classes and Coercion I

- Different types of objects have different rules about mixing classes
- Vectors can *not* contain different types of data
  - Different types of data will be "**coerced**" into the lowest-common denominator type of object

```
mixed_vector = c(pi, 12, "apple", 6.32)
class(mixed_vector)
```

```
[1] "character"
```

```
mixed_vector
```

```
[1] "3.14159265358979" "12"               "apple"            "6.32"
```

Objects: Different Classes and Coercion II

- Data frames can have columns with different types of data, so long as all the elements in each column are the same class[1]

```r
df
```

```
     fruits numbers
1     apple     3.3
2    orange     2.0
3      pear     6.1
4      kiwi     7.5
5 pineapple     4.2
```

```r
class(df$fruits)
```

```
[1] "character"
```

```r
class(df$numbers)
```

```
[1] "numeric"
```

[1] Remember each column in a data frame is a vector!

More on Dataframes I

- Learn more about a data frame with the `str()` command to view its structure

```r
class(df)
```

```
[1] "data.frame"
```

```r
str(df)
```

```
'data.frame':	5 obs. of  2 variables:
 $ fruits : chr  "apple" "orange" "pear" "kiwi" ...
 $ numbers: num  3.3 2 6.1 7.5 4.2
```

More on Dataframes II

- Take a look at the first 5 (or `n`) rows with `head()`

```r
head(df)
```

```
     fruits numbers
1     apple     3.3
2    orange     2.0
3      pear     6.1
4      kiwi     7.5
5 pineapple     4.2
```

```r
head(df, n=2)
```

```
  fruits numbers
1  apple     3.3
2 orange     2.0
```

More on Dataframes III

Get summary statistics[1] by column (variable) with `summary()`

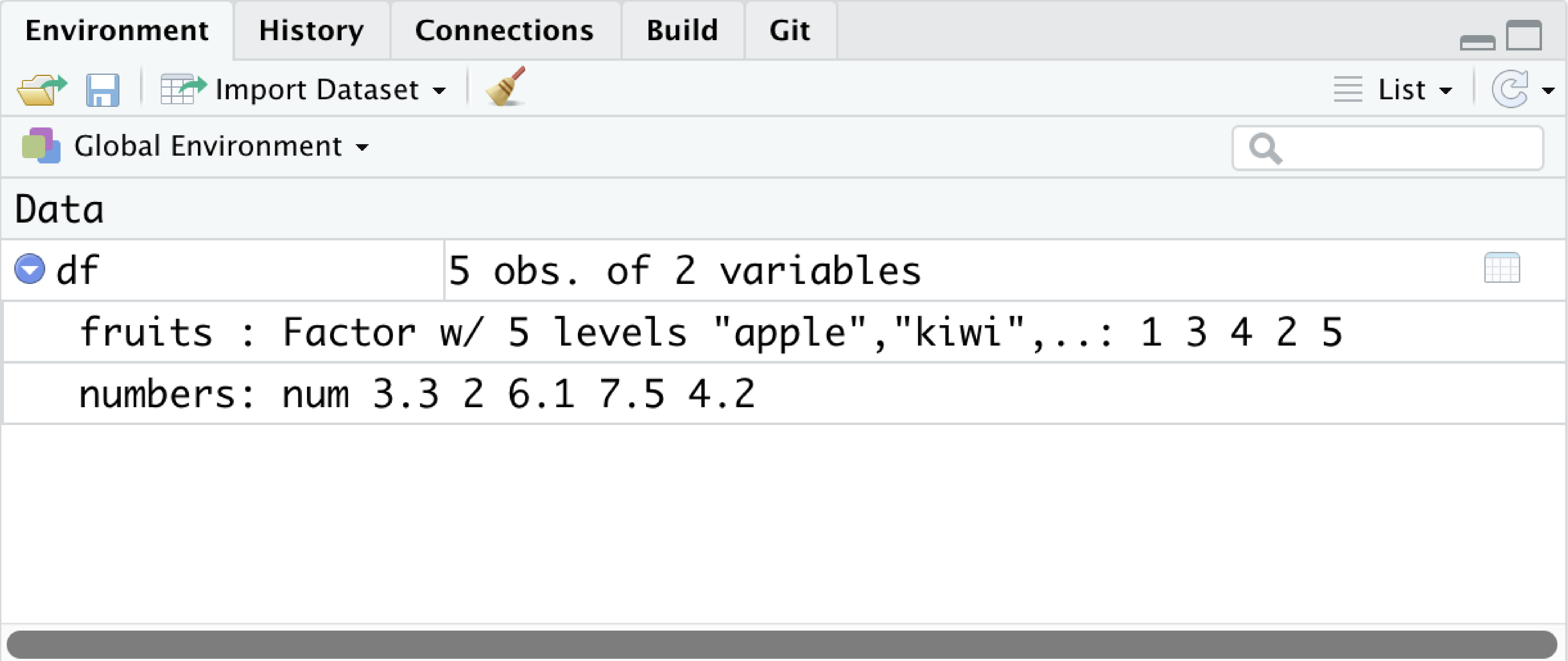
```r
summary(df)
```

```
    fruits             numbers     
 Length:5           Min.   :2.00  
 Class :character   1st Qu.:3.30  
 Mode  :character   Median :4.20  
                    Mean   :4.62  
                    3rd Qu.:6.10  
                    Max.   :7.50  
```

[1] For `numeric` data only; a frequency table is displayed for `character` or `factor` data.

More on Dataframes IV

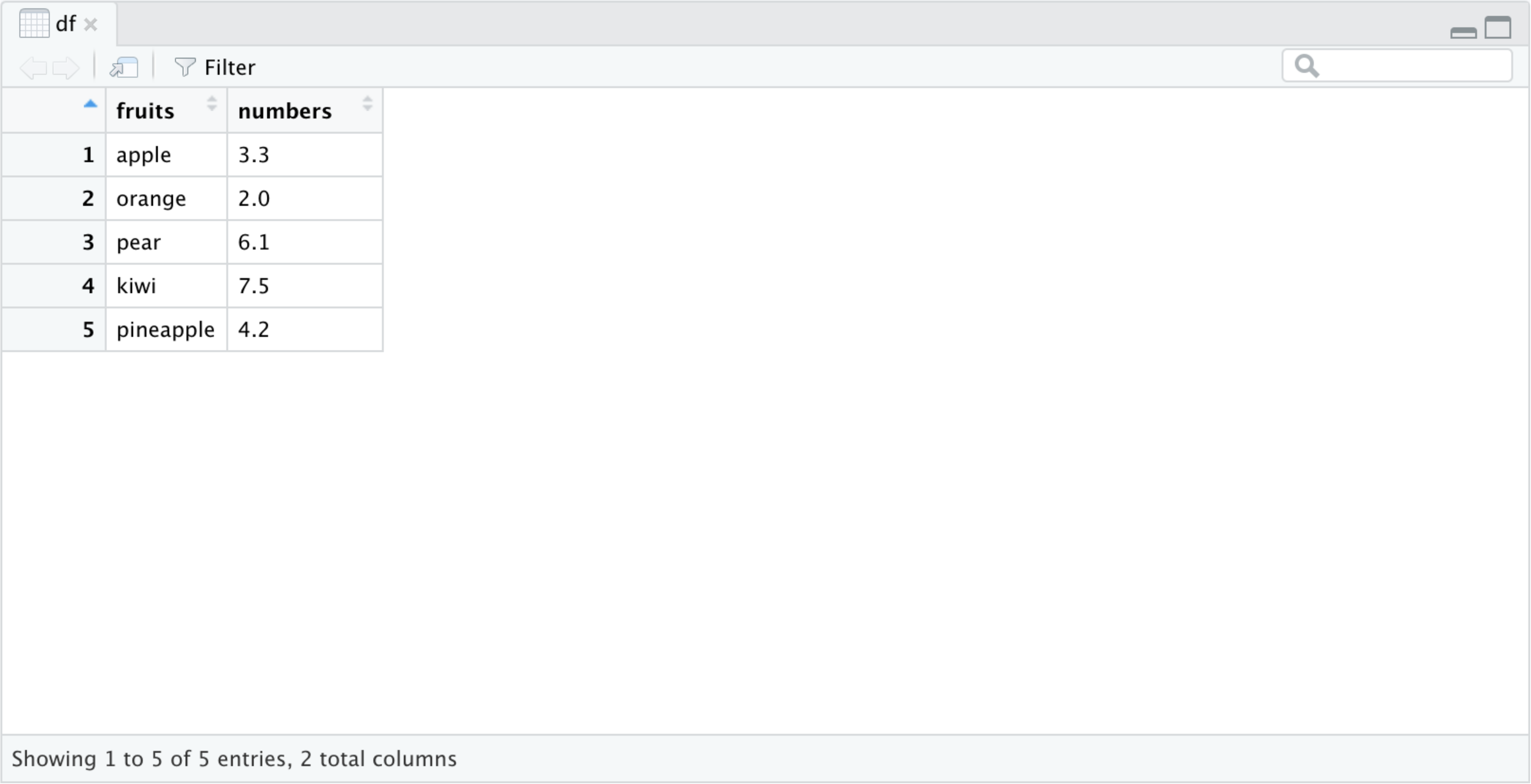
- Note, once you save an object, it shows up in the **Environment Pane** in the upper right window
- Click the blue arrow button in front of the object for some more information



```
df                  5 obs. of 2 variables
   fruits : Factor w/ 5 levels "apple","kiwi",..: 1 3 4 2 5
   numbers: num 3.3 2 6.1 7.5 4.2
```

More on Dataframes V

- `data.frame` objects can be viewed in their own panel by clicking on the name of the object in the environment pane
- Note you cannot edit anything in this pane, it is for viewing only



	fruits	numbers
1	apple	3.3
2	orange	2.0
3	pear	6.1
4	kiwi	7.5
5	pineapple	4.2

Showing 1 to 5 of 5 entries, 2 total columns

Functions Again I

- Functions in R are **vectorized**, meaning running a function on a vector applies it to *each* element

```r
my_vector = c(2,4,5,10) # create object called my_vector
my_vector # look at it
```

```
[1]  2  4  5 10
```

```r
my_vector+4 # add 4 to all elements of my_vector
```

```
[1]  6  8  9 14
```

```r
my_vector^2 # square all elements of my_vector
```

```
[1]   4  16  25 100
```

Functions Again II

- But often we want to run functions on vectors that *aggregate* to a result (e.g. a statistic):

```
length(my_vector) # how many elements?
```

```
[1] 4
```

```
sum(my_vector) # add all elements together
```

```
[1] 21
```

```
max(my_vector) # find largest element
```

```
[1] 10
```

```
min(my_vector) # find smallest element
```

```
[1] 2
```

```
mean(my_vector) # mean of all elements
```

```
[1] 5.25
```

```
median(my_vector) # median of all elements
```

```
[1] 4.5
```

```
var(my_vector) # variance of object
```

```
[1] 11.58333
```

```
sd(my_vector) # standard deviation of object
```

```
[1] 3.40343
```

Some Common Errors

- If you make a coding error (e.g. forget to close a parenthesis), R might show a + sign waiting for you to finish the command

```
> 2+(2*3
+
```

- Either finish the command– e.g. add)–or hit Esc to cancel