Data Science Basics in R, Day 1

Intro to statistical programming

Introductions

- Your name
- What you do for school, work, and/or fun
- Why you signed up for this class
- Have you ever used R before?

Housekeeping notes

- Take a break whenever you need one, we will also have a few structured breaks as a larger group
- Outlet locations
- Trash cans
- Try to come with a charged laptop
- If you have questions, you can find me at *sde31@georgetown.edu*

Housekeeping notes

All course materials are on github, we'll talk more about github starting tomorrow

https://github.com/seaneff/data-science-basics-2024

What to expect: Learning R

- Learning R is fun! And also frustrating.
- You won't be an expert by the end of these 4 days.
- But over time and as you practice, it gets easier!

What to expect: The next 4 days

- We'll balance slides/demos with hands-on exercises
- You decide how you learn best
  - listening with your computer away
  - laptop out and typing along
  - taking notes with paper/pen
- All of these slides, course notes, and other materials are publicly accessible on github

Workshop goals

This workshop will build literacy and basic proficiency in statistical programming, with a focus on the skills needed to conduct data analyses in professional healthcare and public health workplaces. We will cover the basics of data management, data cleaning, data visualization, and basic statistical calculations in R, and version control in github. Participants will leave with a small portfolio of relevant data visualizations and analyses completed using a real-world public health dataset.

Workshop goals

- Learning to program (in R) can be fun and creative, and doesn't have to be overwhelming or intimidating.
- Anyone can learn to write code.

Artwork by Allison Horst

Learning by doing

For some people, it's easier to learn by doing, typing, and making mistakes. I'm one of those people. Others prefer to listen, think, and work through problems later on their own. **In this workshop, we'll pause to do worked examples.** Sometimes these will be confusing. This is the point! We will learn together by trial and error. If you are more comfortable following along for now, feel free to just watch and try at home. But I really encourage you to try, the best way to learn R is to repeatedly do stuff wrong and then figure out the errors.

Learning by doing

Artwork by Allison Horst

Goals for today

- **Understand** what statistical programming is
- **Get acquainted** with Rstudio
- **Write your very first R code** (at least, of this workshop)
  - vectors
  - functions

  - accessing documentation

- **Explore github** to access course materials

Goals for today

- **Understand** what statistical programming is
- **Get acquainted** with Rstudio
- **Write your very first R code** (at least, of this workshop)
  - vectors
  - functions
  - accessing documentation
- **Explore github** to access course materials

What is statistical programming, and why should I care?

What is statistical programming?

Statistical programming is using code to clean, analyze, visualize, and interpret data.

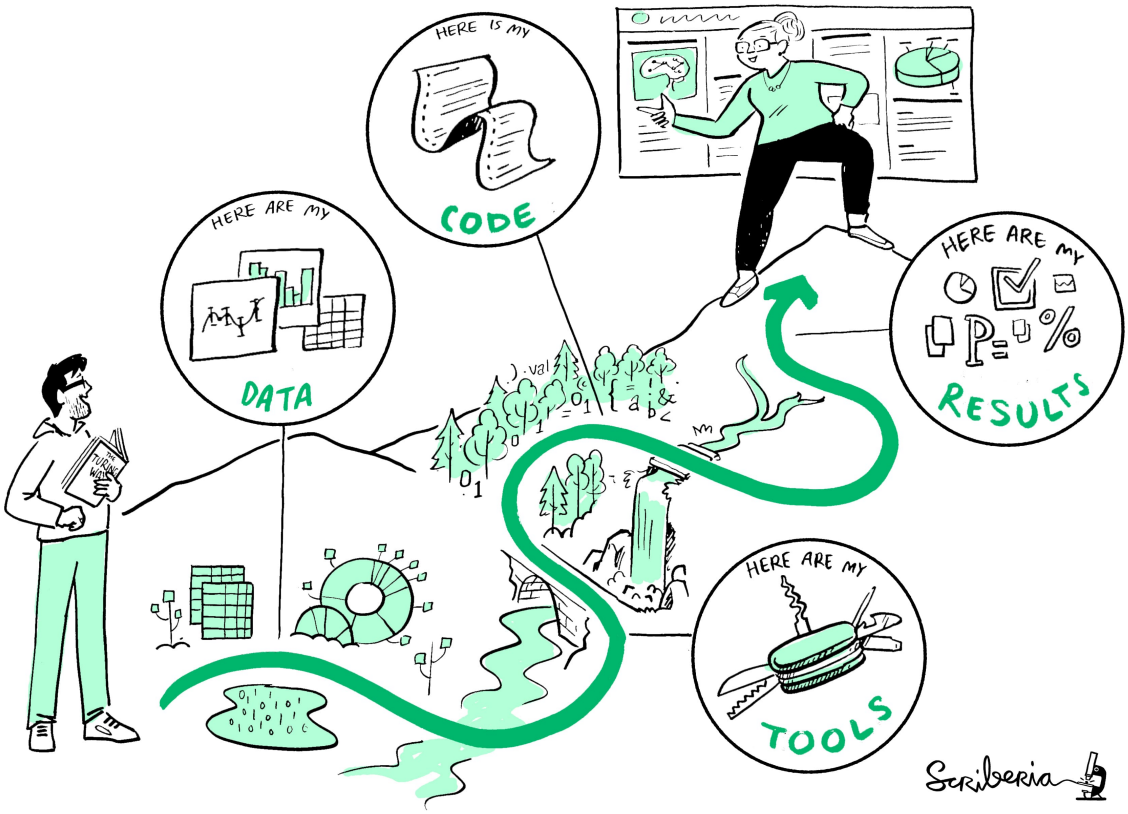
This illustration is created by Scriberia with The Turing Way community, used under a CC-BY 4.0 licence. DOI: 10.5281/zenodo.3332807

What is R?

The R logo is © 2016 The R Foundation

What is RStudio?

Artwork by Allison Horst



- R is a programming language for statistical computing
- Created by Ross Ihaka and Robert Gentleman in 1996
- R is open-source and free
- Many people use R in different ways and for different purposes, but it's defined specifically for data analysis and visualization (unlike other open-source languages like python)

- R Studio is, put simply, a place to write and run R code
- It's an IDE (integrated development environment) and supports both R and python
- R is open-source and free
- It's also free (with enterprise upgrades)

Why learn R?

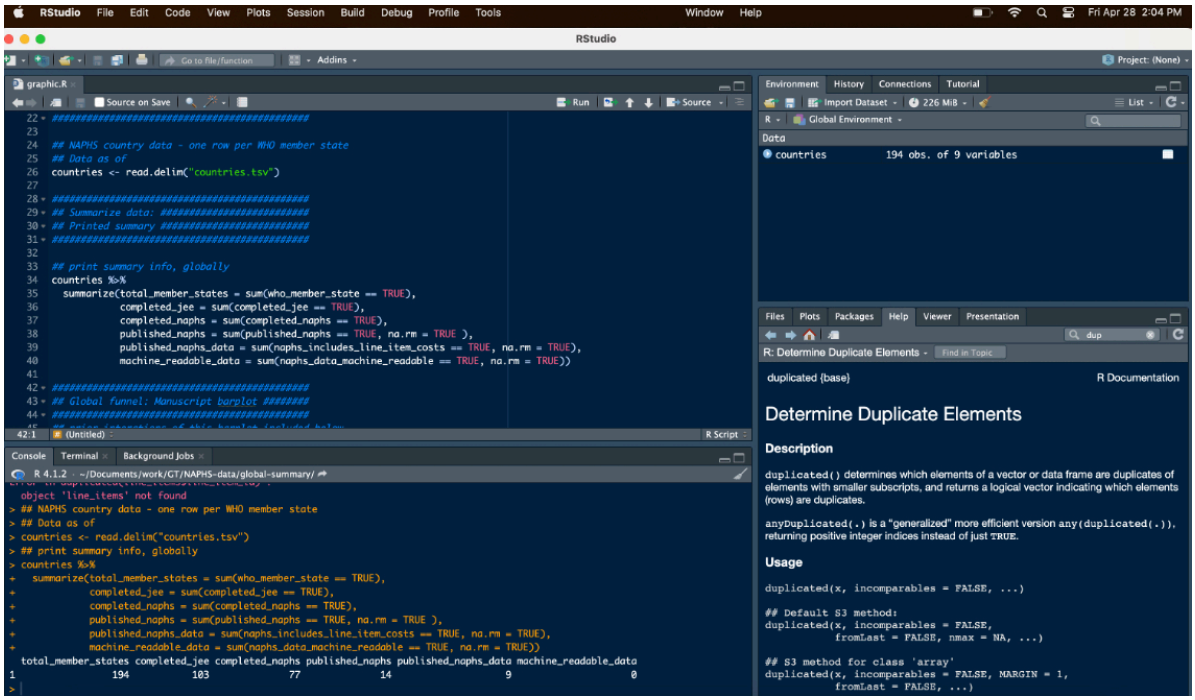
- Learning R helps you understand your data and understand how analysis works, whether you're a researcher, a data scientist, or someone who collaborates with folks who do analysis
- Coding helps you think rigorously about your questions
- It's free (vs. other more expensive tools like SAS or SPSS)
- Sharable, reproducible code and research
- Lots of academics/companies/agencies use it
- It's fun (honestly)

Goals for today

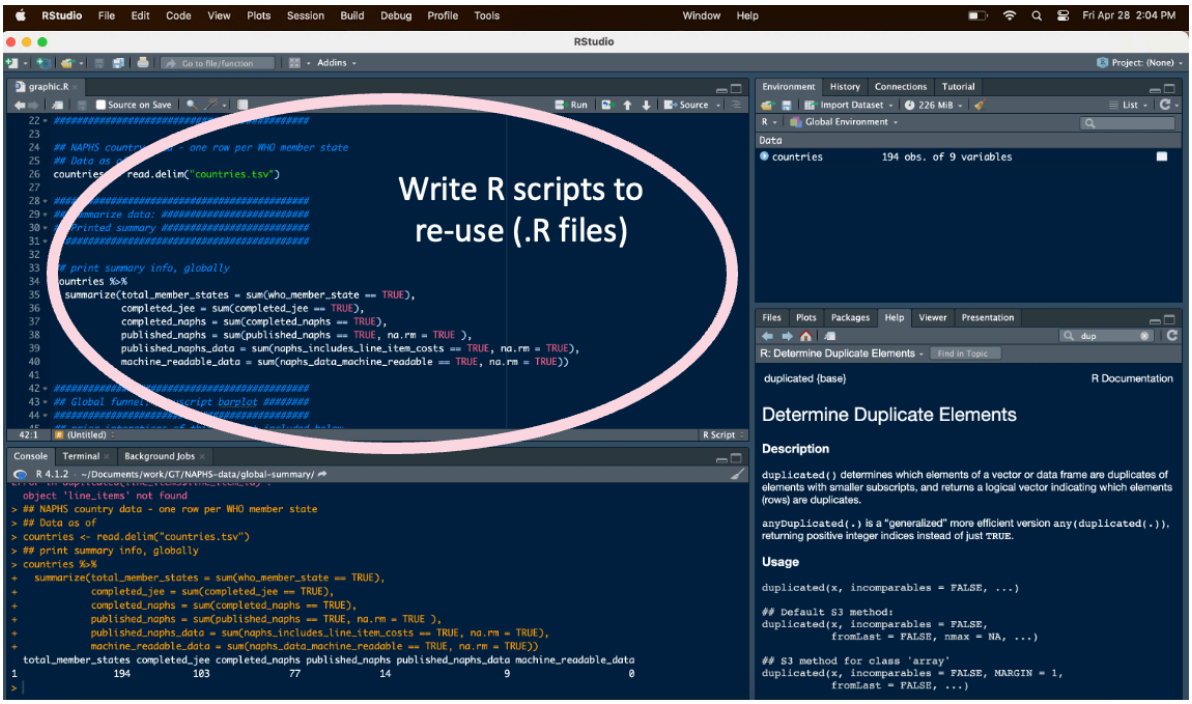
- **Understand** what statistical programming is
- **Get acquainted** with Rstudio
- **Write your very first R code** (at least, of this workshop)
  - vectors
  - functions
  - accessing documentation
- **Explore github** to access course materials

Getting acquainted with RStudio

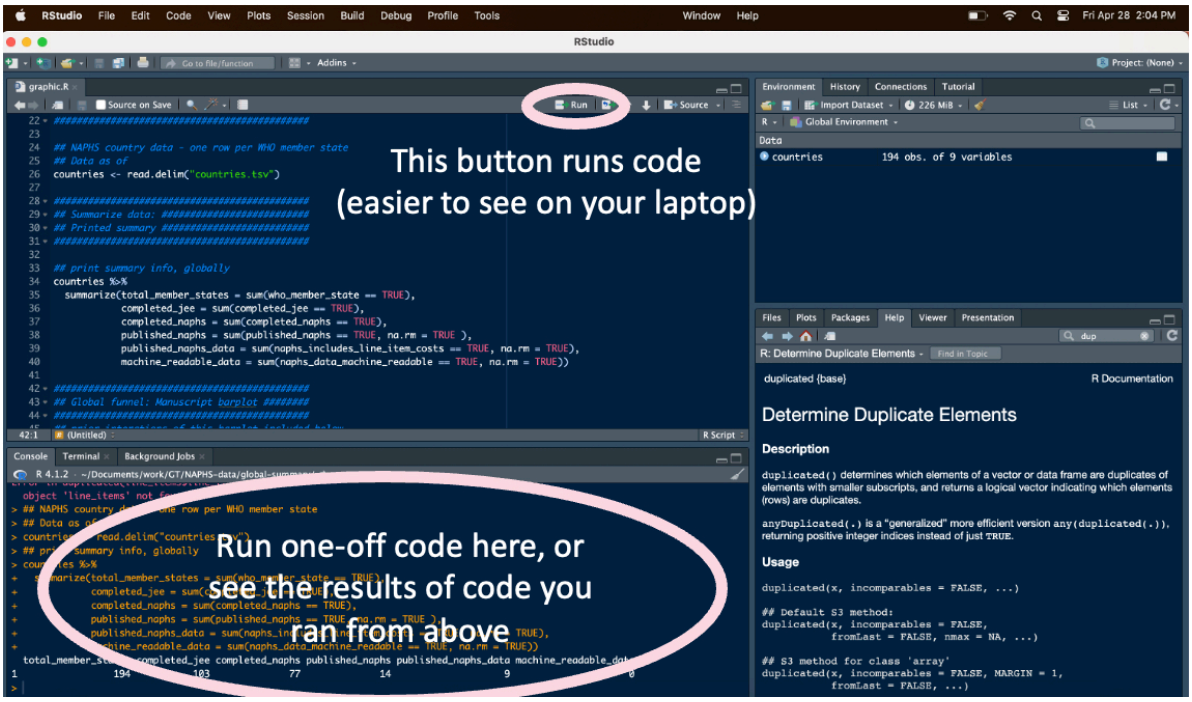
R Studio Console



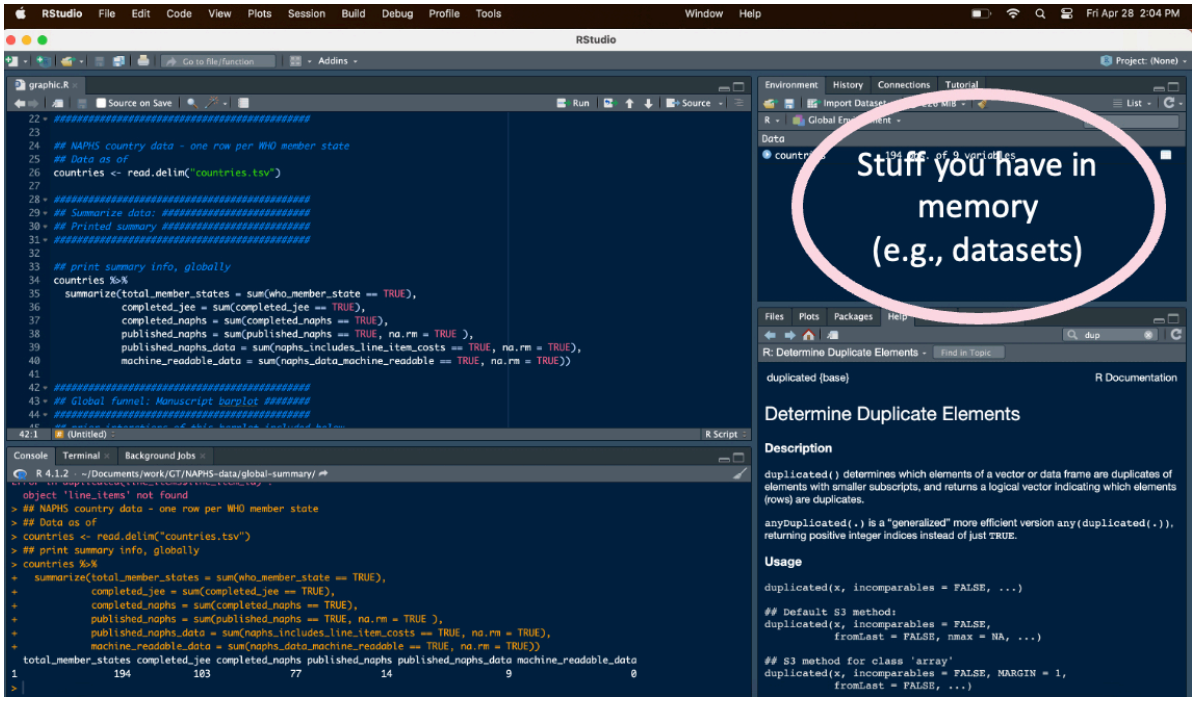
R Studio Console



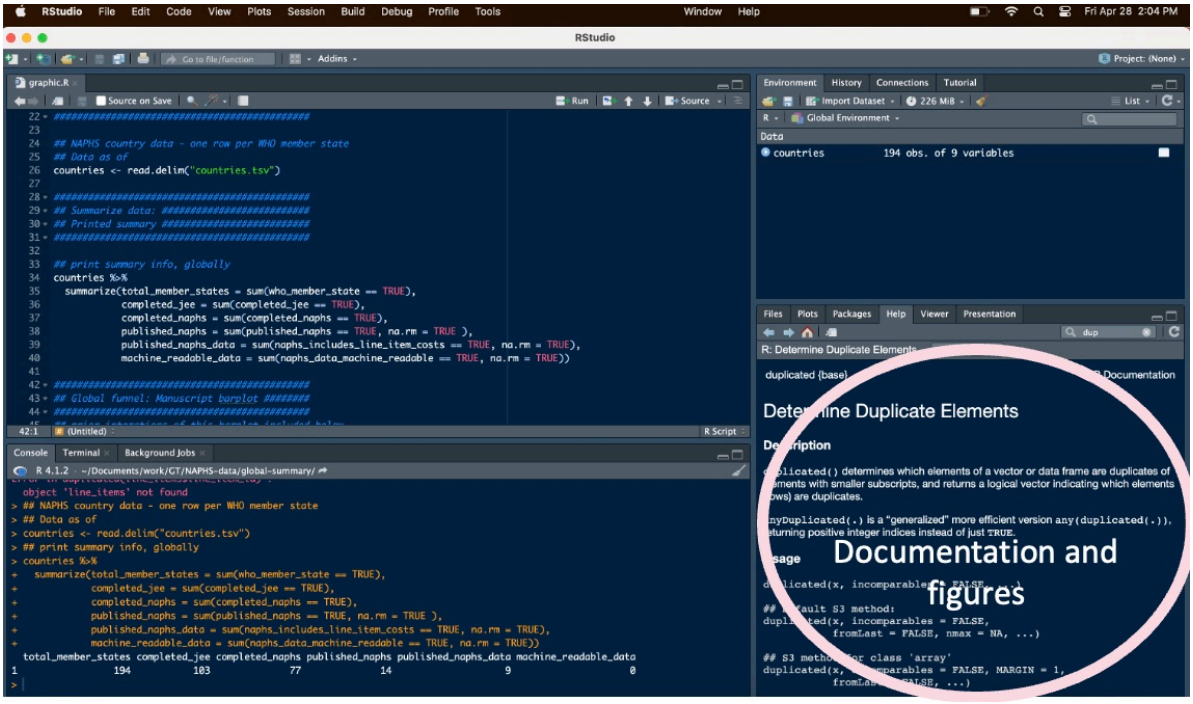
R Studio Console



R Studio Console



R Studio Console



Changing R Studio Appearance

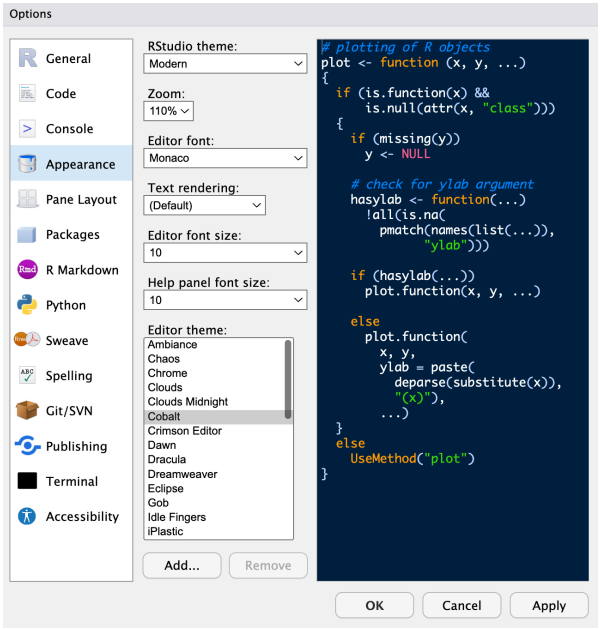
15 minute break (and a note on worked examples)

- If you haven't already, try to download R and Rstudio before tomorrow's class. I'll be around by Zoom or email if you have any questions, and can help troubleshoot.
- Today, we'll do some worked examples sharing my laptop. If you already have R and Rstudio installed on your laptop, feel free to follow along there.

Goals for today

- **Understand** what statistical programming is
- **Get acquainted** with Rstudio
- **Write your very first R code** (at least, of this workshop)
  - vectors
  - functions
  - accessing documentation
- **Explore github** to access course materials

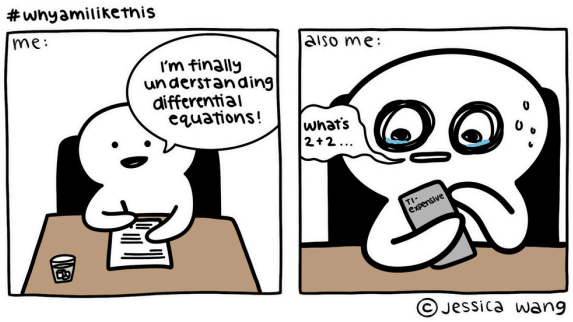
- Navigate from the top bar
  - -> Tools
  - -> Global options
  - -> Appearance
  - (then click “apply”)



R Basics

Using R as a calculator

- R can do everything a basic calculator can do
- Using R as a calculator is a great first step



Comic by Jessica Wang

Using R as a calculator

```
## Addition
1+1
```

```
[1] 2
```

```
## Subtraction
8-10
```

```
[1] -2
```

```
## Multiplication
5*2
```

```
[1] 10
```

```
## Division
88/44
```

```
[1] 2
```

Using R as a calculator

```
## Logarithms
log(1)
```

```
[1] 0
```

```
## Square root
sqrt(64)
```

```
[1] 8
```

```r
## Exponents
8^2
```

```
[1] 64
```

```r
## Absolute value
abs(-1)
```

```
[1] 1
```

```r
## Order of operations
(432+4)/2
```

```
[1] 218
```

Now you try!

- Use R to do some basic math
  - Add two numbers together
  - Multiply three or more numbers
  - Take the square root of a number

Objects

- An object is something you save to R's working memory
- It can be almost anything
  - A string (e.g., your name)
  - A number (e.g., 3.14)
  - A dataset (e.g., that file you have in Excel)
- We assign objects using a little arrow with the syntax (<-)
- When doing data analysis, the most common object you'll probably save is a dataframe, like an Excel or .csv file that you can access from within R (more on this later)

Objects

```
## Create an object named "my_first_object" with value 3.14
my_first_object <- 3.14
## print out the object we created
my_first_object
```

```
[1] 3.14
```

```
## Create an object named "my_second_object" with value 1+1
my_second_object <- 1+1
## print out the object we created
my_second_object
```

```
[1] 2
```

```
## Create an object named "my_third_object" with value "Steph"
my_third_object <- "Steph"
## print out the object we created
my_third_object
```

```
[1] "Steph"
```

Use (numeric) objects to do math

- Just like we did when we used R as a calculator, you can also use numeric objects to do math
- When you do this, the objects themselves don't change unless you explicitly re-assign them to new variables

Use (numeric) objects to do math

```
## reminder: print out value of my_first_object
my_first_object
```

```
[1] 3.14
```

```
## reminder: print out value of my_second_object
my_second_object
```

```
[1] 2
```

```
## multiply together
my_first_object*my_second_object
```

```
[1] 6.28
```

Now you try!

- Pick your favorite number, and save it as an object
- Pick another number, and save it as another object
- Do one basic calculation (e.g., addition) with your objects

Artwork by Allison Horst

Data types in R

- **numeric:** a number (e.g., -1, 0, 893243.343)
- **logical:** TRUE or FALSE (no quotations)
- **character:** letters and words (tricky – or a number stored as a string)

Data types in R

- **numeric:** a number (e.g., -1, 0, 893243.343)
- **logical:** TRUE or FALSE (no quotations)
- **character:** letters and words (tricky – or a number stored as a string)
- the function `is()` will help us learn the data type of a given object

```
is(3.14)
```

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

```
is(TRUE)
```

```
[1] "logical" "vector"
```

```
is("Steph")
```

```
[1] "character"           "vector"              "data.frameRowLabels"
[4] "SuperClassMethod"
```

Numeric data

```
my_first_object
```

```
[1] 3.14
```

Numeric data

```
my_first_object
```

```
[1] 3.14
```

```
is(my_first_object)
```

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

Character data

```
policy <- "International Health Regulations (IHR)"
```

Character data

```
policy <- "International Health Regulations (IHR)"
policy
```

```
[1] "International Health Regulations (IHR)"
```

```
is(policy)
```

```
[1] "character"           "vector"              "data.frameRowLabels"
[4] "SuperClassMethod"
```

Logical data

```
logical_example <- TRUE
```

Logical data

```
logical_example <- TRUE
logical_example
```

```
[1] TRUE
```

```
is(logical_example)
```

```
[1] "logical" "vector"
```

Check your understanding

```
is(-1)
```

Check your understanding

```
is(-1)
```

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

Check your understanding

```
is(-1)
```

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

```
is(FALSE)
```

Check your understanding

```
is(-1)
```

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

```
is(FALSE)
```

```
[1] "logical" "vector"
```

Check your understanding

```
is(-1)
```

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

```
  is(FALSE)
```

```
[1] "logical" "vector"
```

```
  is("What is this?")
```

Check your understanding

```
  is(-1)
```

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

```
  is(FALSE)
```

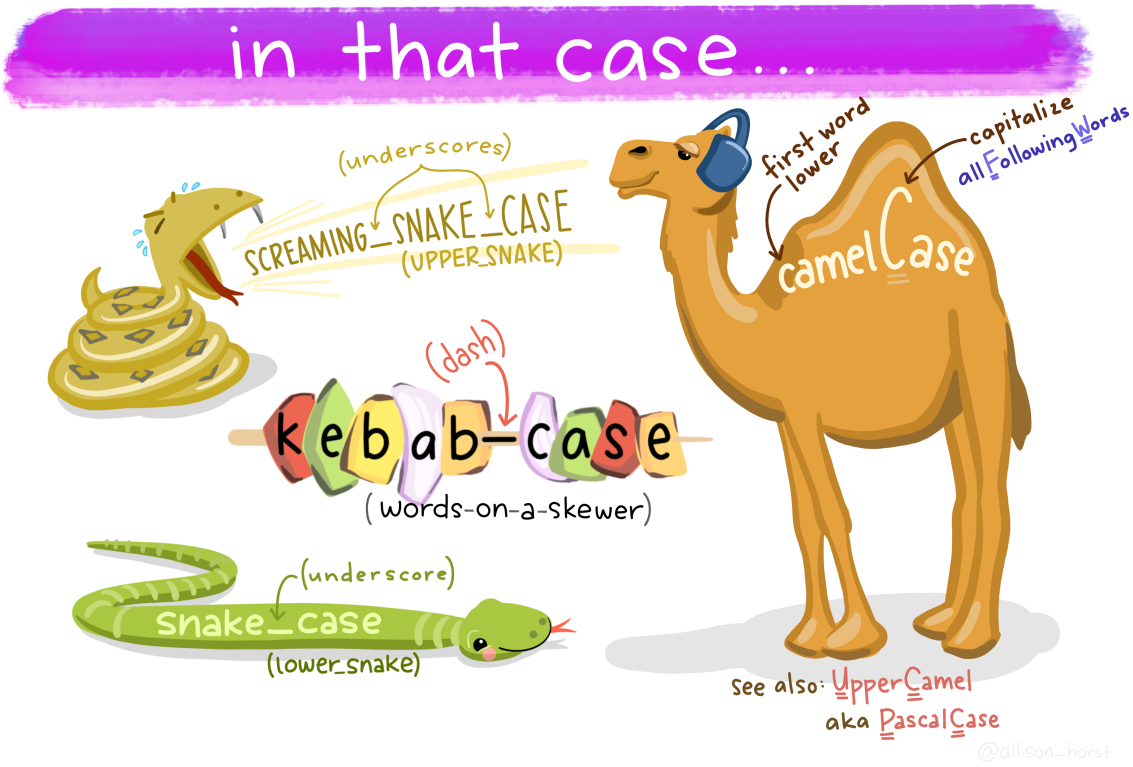
```
[1] "logical" "vector"
```

```
  is("What is this?")
```

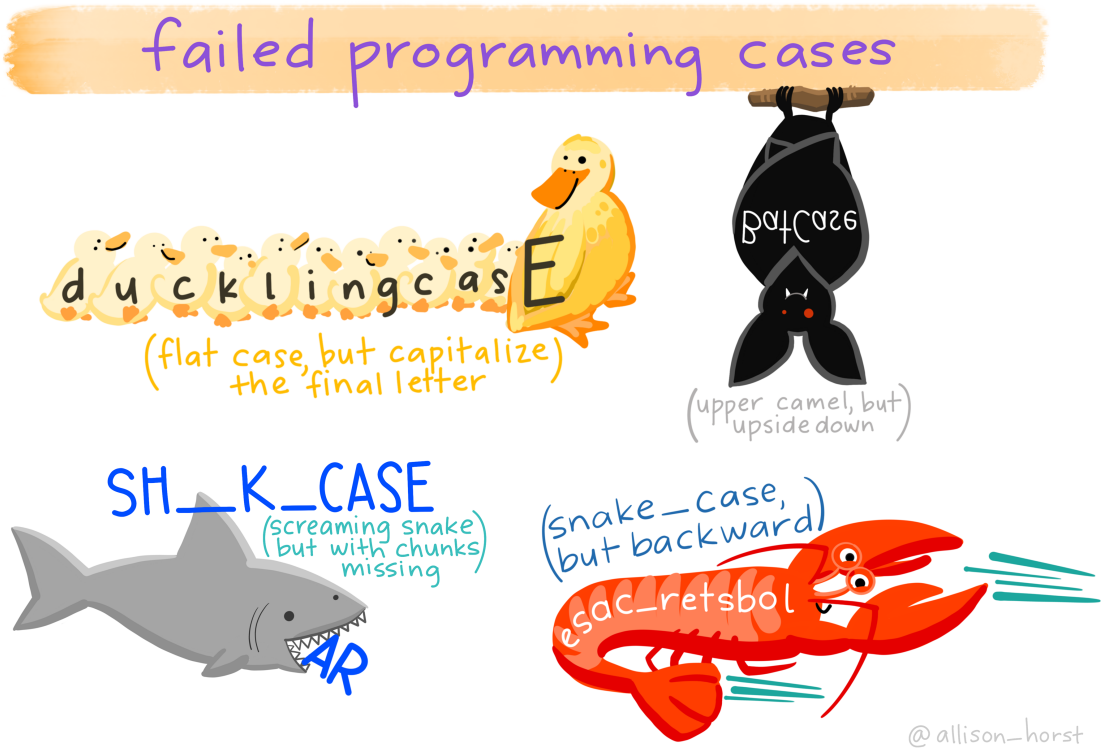
```
[1] "character"           "vector"              "data.frameRowLabels"
[4] "SuperClassMethod"
```

Rules for naming objects

- General naming requirement: a variable name can't start with a number or a dot (.)
- R is case sensitive ('A' is different than 'a')
- General rules of thumb: aim for consistency
  - snake_case
  - camelCase
  - whatever.this.is
- Chose a name you'll understand when you open your code the next day, or when someone else reviews it


in that case...
(underscores)
SCREAMING_SNAKE_CASE
(UPPER_SNAKE)
(dash)
kebab-case
(words-on-a-skewer)
(underscore)
snake_case
(lower_snake)
first word lower
capitalize allFollowingWords
camelCase
See also: UpperCamel aka PascalCase
@allison_horst

Artwork by Allison Horst



Artwork by Allison Horst

Now you try!

- Create three new objects, with any allowable names you want. Try to use a consistent naming style.
  - Numeric (we already did this, but practice is good)
  - Character
  - Logical

Vectors

- Vectors are grouped data elements in a specific order
- For example, data in a specific column in Excel
- When you've thought previously about data analysis, you probably think about vectors, even if you didn't use that name

Vectors

- Vectors are grouped data elements in a specific order
- For example, data in a specific column in Excel
- When you've thought previously about data analysis, you probably think about vectors, even if you didn't use that name

name	iso_3166	stanag_code	internet_cod	who_member_state
Afghanistan	AFG	AF | AFG | 004	AFG	TRUE
Albania	ALB	AL | ALB | 008	ALB	TRUE
Algeria	DZA	DZ | DZA | 012	DZA	TRUE
Andorra	AND	AD | AND | 020	AND	TRUE
Angola	AGO	AO | AGO | 024	AGO	TRUE
Antigua and Barbuda	ATG	AG | ATG | 028	ATG	TRUE
Argentina	ARG	AR | ARG | 032	ARG	TRUE
Armenia	ARM	AM | ARM | 051	ARM	TRUE
Australia	AUS	AU | AUS | 036	AUS	TRUE

Vectors

```
diseases <- c("HIV", "TB", "Malaria")
```

Vectors

```
diseases <- c("HIV", "TB", "Malaria")
diseases
```

```
[1] "HIV"     "TB"      "Malaria"
```

```
  is(diseases)
```

```
[1] "character"           "vector"              "data.frameRowLabels"
[4] "SuperClassMethod"
```

Vectors

```
  diseases <- c("HIV", "TB", "Malaria")
  diseases
```

```
[1] "HIV"     "TB"      "Malaria"
```

```
  is(diseases)
```

```
[1] "character"           "vector"              "data.frameRowLabels"
[4] "SuperClassMethod"
```

```
  values <- c(28, 84, 432)
```

Vectors

```
  diseases <- c("HIV", "TB", "Malaria")
  diseases
```

```
[1] "HIV"     "TB"      "Malaria"
```

```
  is(diseases)
```

```
[1] "character"           "vector"              "data.frameRowLabels"
[4] "SuperClassMethod"
```

```
  values <- c(28, 84, 432)
  values
```

```
[1]  28  84 432
```

```r
is(values)
```

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

Vectors

- Vectors can contain strings, numbers, or other data types
- But they can't contain multiple data types at once

```r
c("HIV", "Malaria", 4325)
```

```
[1] "HIV"     "Malaria" "4325"
```

Vectors

- Vectors can contain strings, numbers, or other data types
- But they can't contain multiple data types at once

```r
c("HIV", "Malaria", 4325)
```

```
[1] "HIV"     "Malaria" "4325"
```

- What happened here?

Vectors

- Vectors can contain strings, numbers, or other data types
- But they can't contain multiple data types at once

```r
c("HIV", "Malaria", 4325)
```

```
[1] "HIV"     "Malaria" "4325"
```

- What happened here?
  - The number 4325 was converted into a string

Now you try!

- Make two vectors in R and assign them to objects.
  - Numeric
  - String

Vectorized Calculations

```
c(1,2,3,4,5)+1
```

```
[1] 2 3 4 5 6
```

Vectorized Calculations

```
c(1,2,3,4,5)+1
```

```
[1] 2 3 4 5 6
```

```
c(1,2,3,4,5)*2
```

```
[1]  2  4  6  8 10
```

Vectorized Calculations

```
c(1,2,3,4,5)+1
```

```
[1] 2 3 4 5 6
```

```
c(1,2,3,4,5)*2
```

```
[1]  2  4  6  8 10
```

```
c(1,2,3,4,5) + c(8,0,0,0,0)
```

```
[1] 9 2 3 4 5
```

Vectorized Calculations

```
c(1,2,3,4,5) + c(8,0)
```

```
Warning in c(1, 2, 3, 4, 5) + c(8, 0): longer object length is not a multiple
of shorter object length
```

```
[1]  9  2 11  4 13
```

Vectorized Calculations

```
c(1,2,3,4,5) + c(8,0)
```

```
Warning in c(1, 2, 3, 4, 5) + c(8, 0): longer object length is not a multiple
of shorter object length
```

```
[1]  9  2 11  4 13
```

- Pay attention to these warnings
- This is called *vector recycling* and it can cause major errors

Functions

- Functions are instructions to perform a task
- They are algorithms, or consistent set of rules
- R has built-in functions for many basic things
- Functions generally look like this: `function(object)`
- We can also "add on" extra functions by loading new libraries (we'll get to this later), or we can write our own functions to do whatever we want

Functions

- Most functions in R are vectorized
  - This means they act on all items in a vector
- Why does this matter?
  - If you misunderstand it, your math will be wrong
  - It's useful for basic calculations and analysis:
    - divide all numbers by 100 to calculate a %
    - multiply per-capita rates by total population

Functions

```
mean(c(1,2,3,4,5))
```

```
[1] 3
```

Functions

```
mean(c(1,2,3,4,5))
```

```
[1] 3
```

```
sd(c(1,2,3,4,5))
```

```
[1] 1.581139
```

Functions

```
mean(c(1,2,3,4,5))
```

```
[1] 3
```

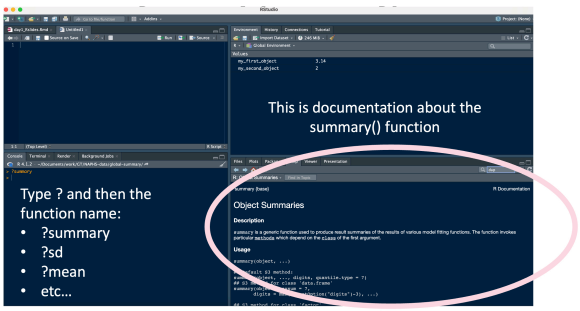
```
sd(c(1,2,3,4,5))
```

```
[1] 1.581139
```

```
summary(c(1,2,3,4,5))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       2       3       3       4       5
```

- Use `?summary` or `help(summary)` to learn more about functions
- Results will show in "help" section of RStudio



Learn more about functions

Now you try!

- Take the average of three or more numbers
- Use "?" to learn more about the function sd()

Goals for today

- **Understand** what statistical programming is
- **Get acquainted** with Rstudio
- **Write your very first R code** (at least, of this workshop)
  - vectors
  - functions
  - accessing documentation
- **Explore github** to access course materials

Accessing course materials

What is github?

- Have you ever saved a bunch of versions of a paper on your computer with different file names at different dates or times of day?
- Backups are useful to save progress, understand what we've done before, and look into problems/bugs
- Github is a tool do help do this with code

What is github?

We'll talk more about github later in this workshop. For now, I'd like you to be able to use it to access course materials any time you'd like to. https://github.com/seaneff/data-science-basics-2024

Todo: add graphics

See 2023 powerpoint