THE UNIVERSITY OF CHICAGO
HARRIS SCHOOL
OF PUBLIC POLICY

Week 1

PPHA 30110 - Coding Lab for Public Policy

Welcome to the Coding Lab!

Today's **Agenda**

1. Intro to the Coding Lab + schedule
2. Why learn R?
3. Introduction to R and RStudio
4. Creating variables in R
5. What are packages and functions?
6. Manipulating vectors in R

Teaching **members**

- Instructor
  - Arthur Cheib - MPP

- Teaching Assistants
  - Eujene Yum - MSCAPP
  - Harish Sai - MPP
  - Khristel Zavaleta - MPP
  - Laís Kimie - MPP
  - Phoebe Collins - MSCAPP
  - Rodrigo Salazar - MSCAPP
  - Wesley Janson - MSCAPP

Why learn how to **code** in **R**?

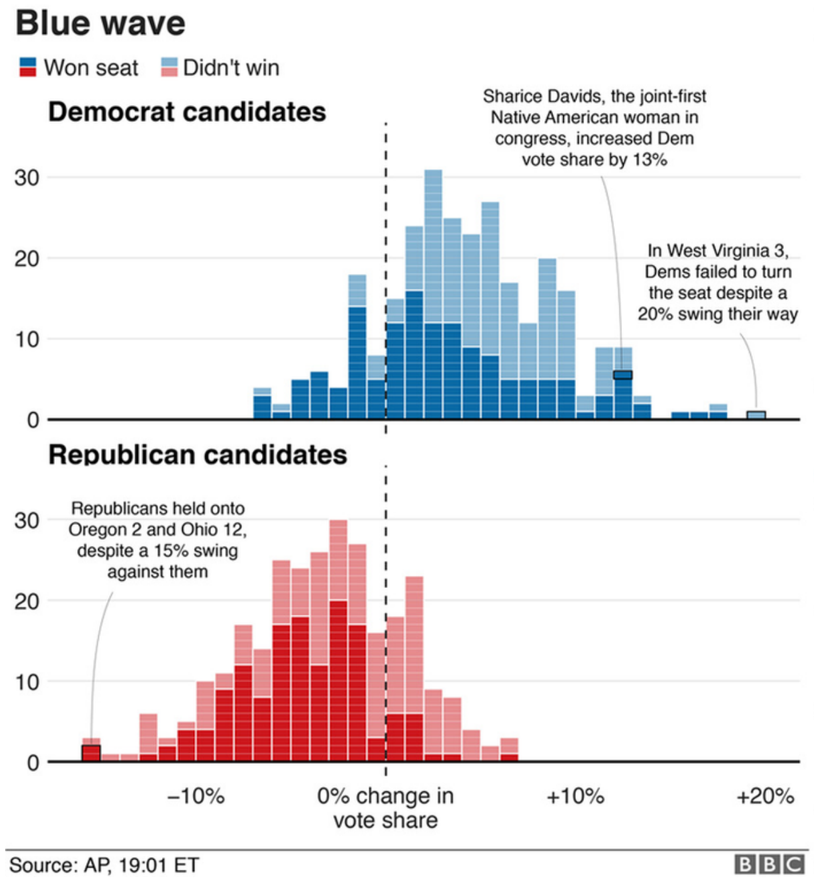
- R is an extremely powerful programming language and statistical software environment
  - Data manipulation + visualization suite
  - Strong statistical packages (e.g., program evaluation, machine learning)

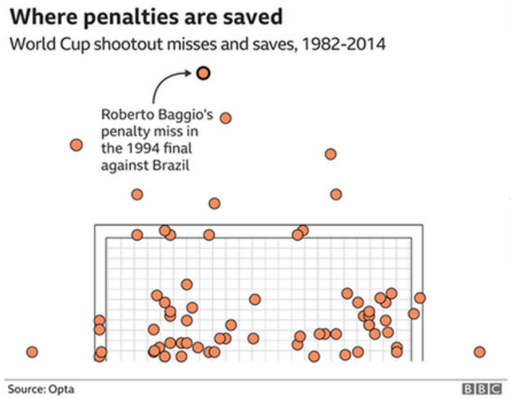
- Many public policy jobs and the Harris curriculum rely on programming
  - to quickly engage with policy data
  - to complete statistical analyses

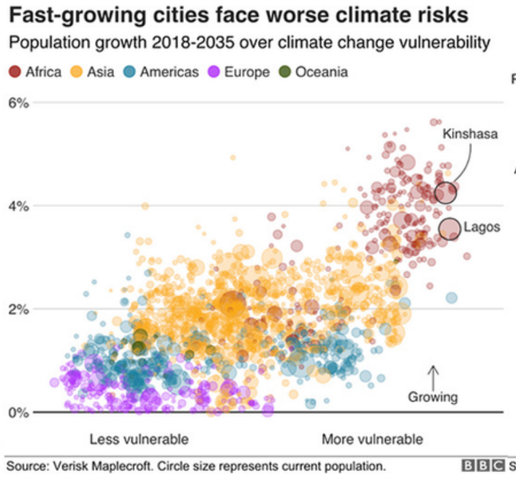
- Open source and free

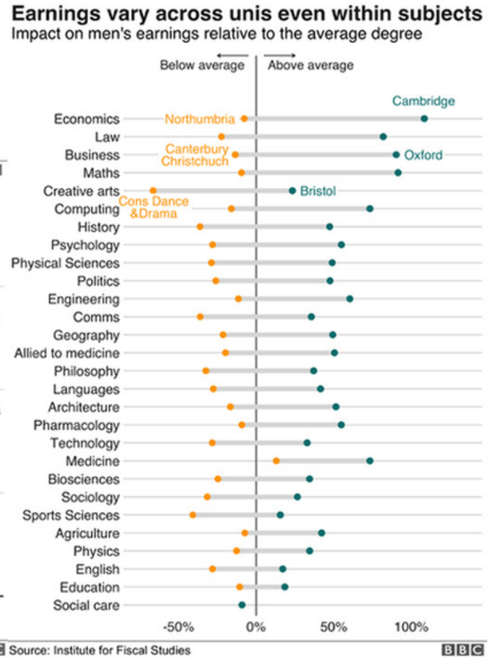
Why learn **coding** in **R**?

- **Graphs (Coding Camp)**



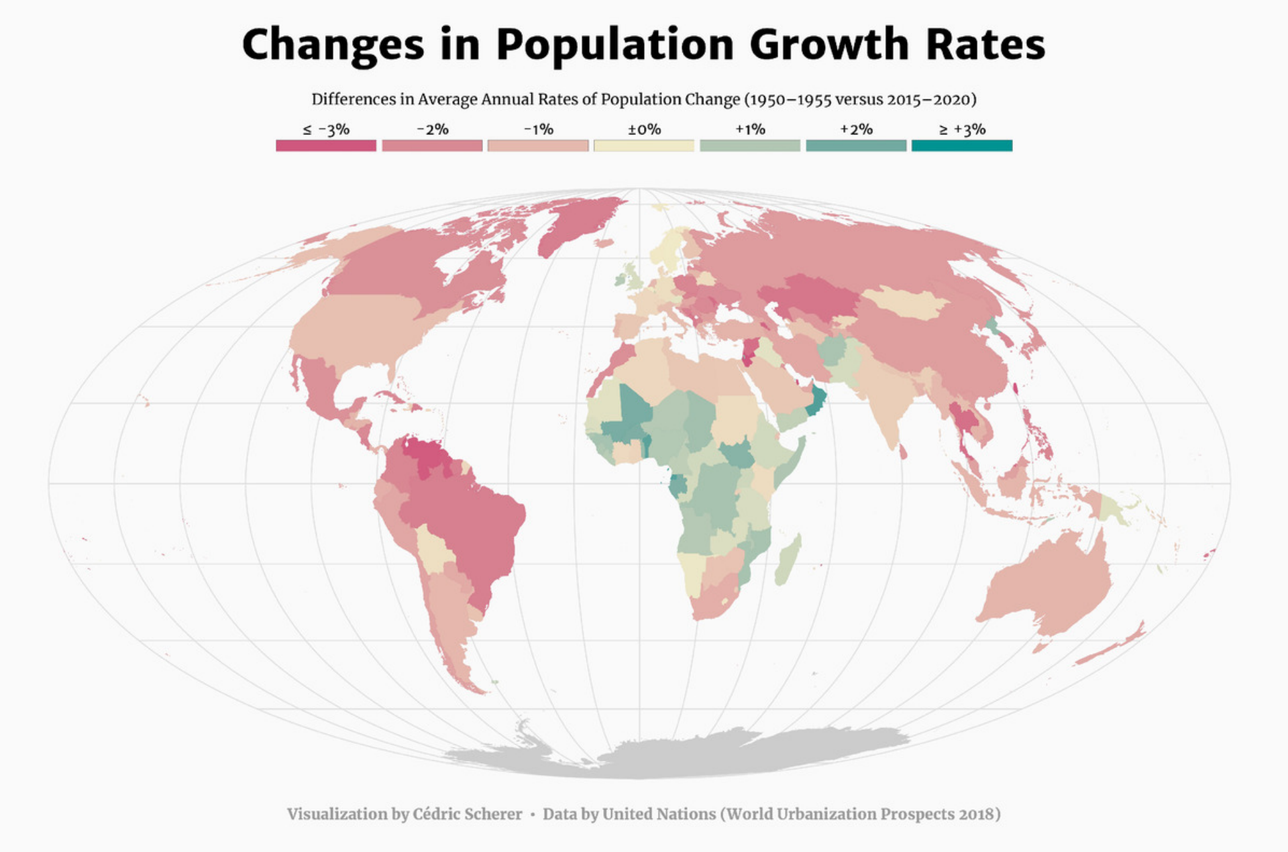






Why learn **coding** in **R**?

- **Maps (data and programming in R I)**



Course **prerequisites**

R (programming language)

RStudio (IDE)

How to download and install?

Check Canvas!

Course review

1. **Introduction to R**

Course review

1. Introduction to R

2. **Importing and Transforming data in R**

Course review

1. Introduction to R
2. Importing and Transforming data in R
3. **Transforming data & creating graphs in R**

Course review

1. Introduction to R
2. Importing and Transforming data in R
3. Transforming data & creating graphs in R
4. **Reshaping data & Control flow in R**

Course review

1. Introduction to R
2. Importing and Transforming data in R
3. Transforming data & creating graphs in R
4. Reshaping data & Control flow in R
5. **Joining data & for loops in R**

What are the **goals** here?

1. **Generous introduction to R**
2. **Set you ready for the core (everything R related)**
3. **Know enough R to perform mid-level data analysis (RA positions)**

Learning **philosphy**

1. **We learn coding by experimenting with code**

Learning **philosphy**

1. We learn coding by experimenting with code

2. **Coding requires a different modality of thinking:** just like learning a new language!

Learning **philosphy**

1. **We learn coding by experimenting with code**

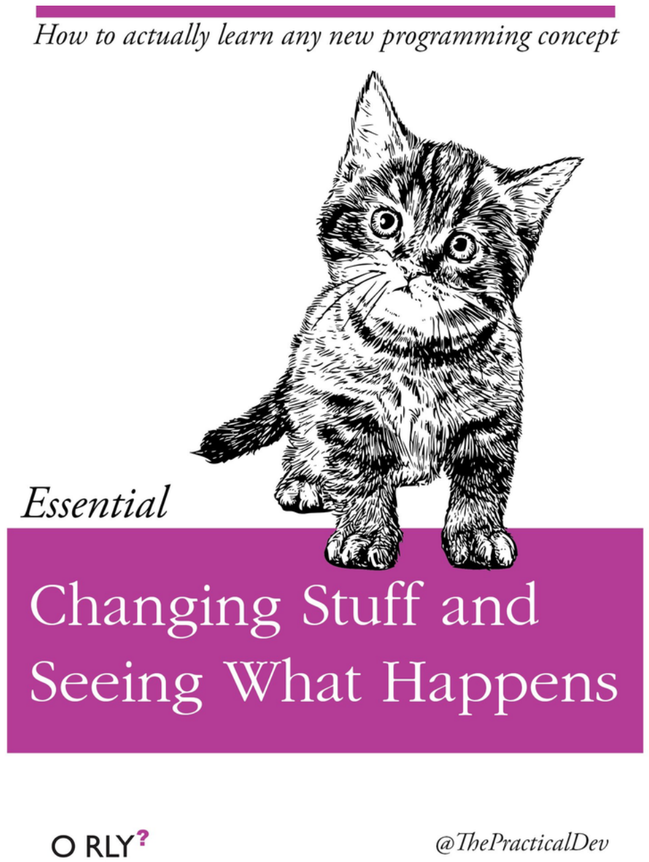
2. **Coding requires a different modality of thinking:** just like learning a new language!

3. **Coding will be frustrating at the beginning!**

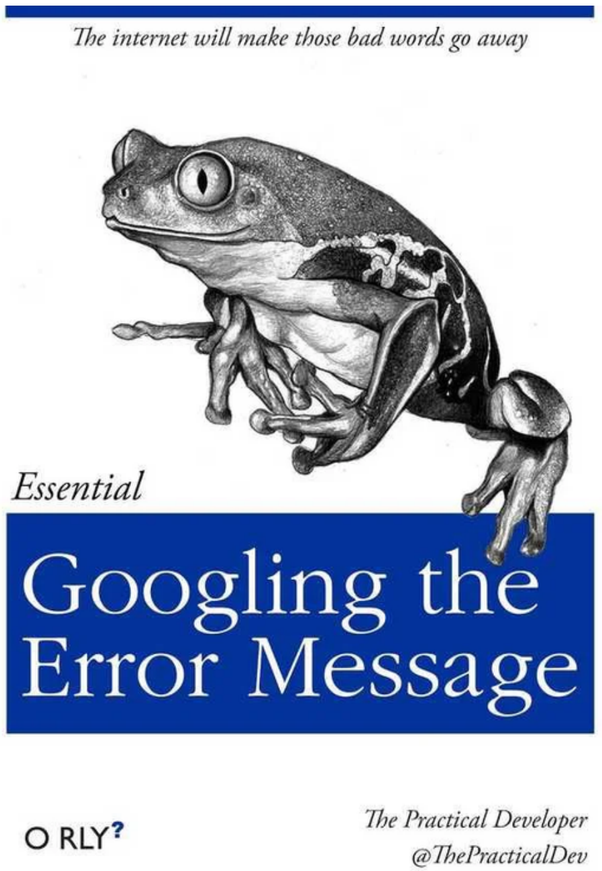
Learning **philosphy**

1. **We learn coding by experimenting with code**
2. **Coding requires a different modality of thinking:** just like learning a new language!
3. **Coding will be frustrating at the beginning!**
4. **Coding lab is for you!**

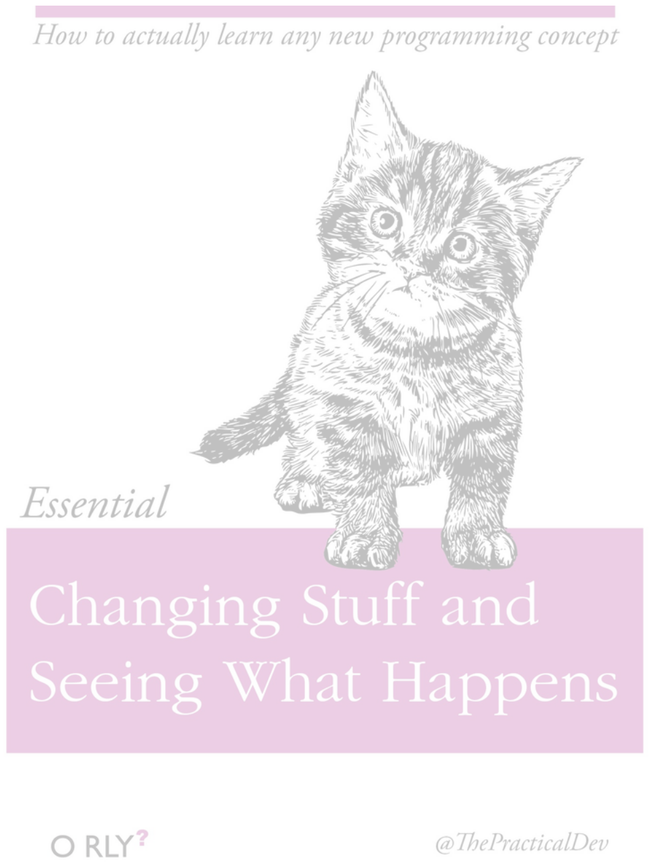
Textbook & **Resources**

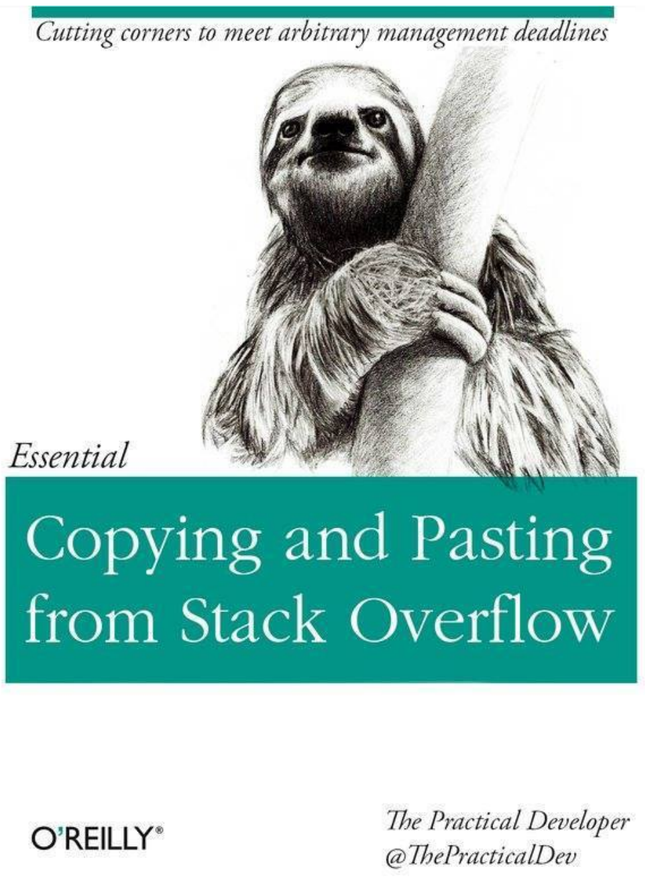


Textbook & **Resources**



Textbook & **Resources**





Questions

(?)

Ready?

Ready?

Almost!

Meet your **neighbour!**

1. **What is their experience with R?**
2. **What is their expectation about learning R?**
3. **The classic: why did you choose Harris?**

Introduction to R / Rstudio

R & RStudio

- **Is there any difference?**

R & RStudio

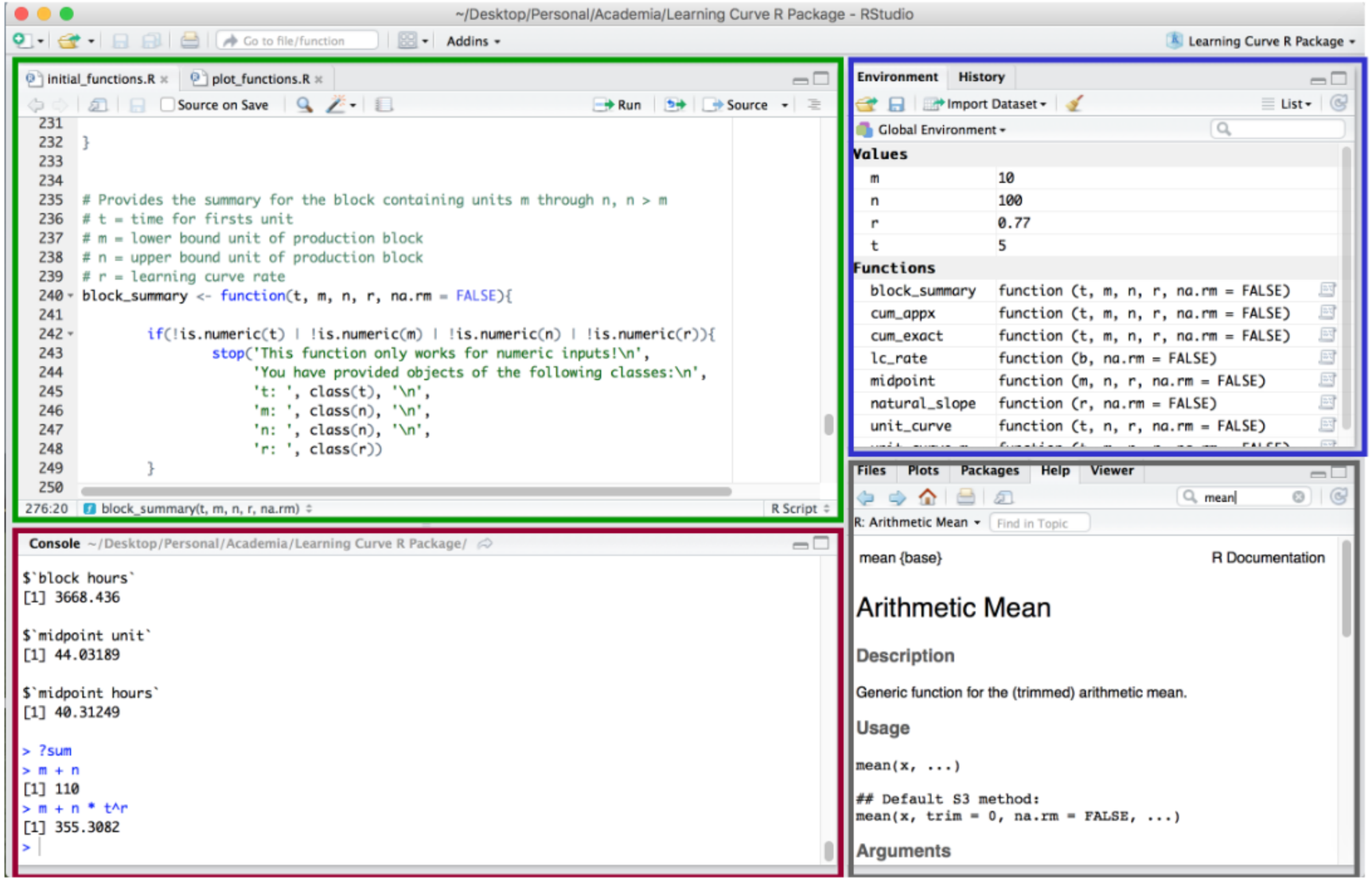
- **Is there any difference?**

  Let's check!

R & RStudio

Script files
- Saves your script
- Allows code & comments
- Can have multiple files open at a time

Console/Command line
- Can use as calculator
- Does not save code
- This is where your output is displayed



Workspace environment
- Holds your objects
- Can review history

Misc - Displays:
- files in working directory
- plots when produced
- help files/search

#1 Try it yourself!

- **Open this Week .R file (script) that you've downloaded from Canvas** `(week1-lecture.R)`

- Below where you can see in the script *"Try it yourself 1 - slide 20"*, write and run the following code - where in the IDE (RStudio) can the outputs be found?

```
1+1

?sum

sum(1,2,3,4,5)

mtcars
```

```
## R as a Calculator (try it!)
## R obeys PEMDAS convention

22 + 3 * 2

5 * 3 / 10 - 5

2 ^ 3 - 5
```

Creating variables in R

Creating **variables** in R

- We can think variables as a **container** with a name

```
my_variable

my_age

dataframe_harris
```

- We can also think of it as a way to interact with R in **his own language...**

Creating **variables** in R

- We can think variables as a **container** with a name

```
my_variable

my_age

dataframe_harris
```

- We can also think of it as a way to interact with R in **his own language...**

- How to create them?

The **assignment operator!**

```
<-
```

```
my_variable <- 5

my_age <- 30

harris_text <- 'Harris'
```

Doing **basic operations** with my variables

- **Using variables you have created for doing for math**

```
## Creating our variable

my_number <- 5

## Guess the output if we run this:

my_number
```

```
22 + 3 * 2

my_number  * 3 / 10 - my_number

2 ^ 3 - my_number
```

Variable reassignment

- What if we want to **update** the value of our variable?

```
## Creating our variable
my_number <- 5


## Updating the value of our variable
my_number <- 10 * my_number
```

Variable reassignment

- What if we want to **update** the value of our variable?

```
## Creating our variable
my_number <- 5


## Updating the value of our variable
my_number <- 10 * my_number
```

- What if we do this instead?

```
## Creating our variable
my_number <- 5


## Updating the value of our variable
10 * my_number
```

Variable reassignment

- What if we want to **update** the value of our variable?

```
## Creating our variable
my_number <- 5

## Updating the value of our variable
my_number <- 10 * my_number
```

CORRECT!

- What if we do this instead?

```
## Creating our variable
my_number <- 5

## Updating the value of our variable
10 * my_number
```

WRONG!

#2 Try it yourself!

Open this Week .R file (script) that you've downloaded from Canvas `(week1-lecture.R)`

What are **functions?**

Functions

- **Functions!**
  - Are objects in R that completes a **task/operation** with the **inputs** we provide and gives us an **output**

inputs / arguments

task

outputs

- The **sum** function

Basic syntax:

```
sum(arg1, arg2, ...)
```

```
# Using the sum function
sum(1,2,3,4,5)


## [1] 15
```

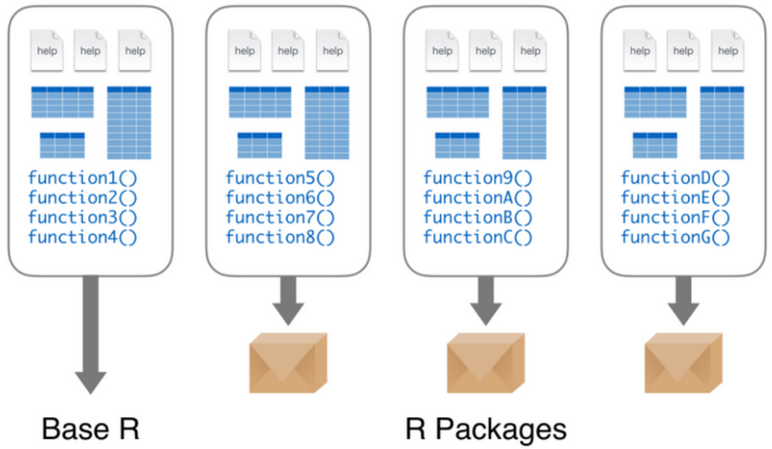
… and where are the
functions coming from?

Packages

Packages!

The fundamental unit of shareable code is the **package**.

- CRAN: 13,500+
- Bioconductor: 1,500+
- GitHub: Many more plus beta versions for updated packages not yet published



Source: Bradley Boehmke - access his GitHub to check his amazing content for learning R

Installing packages

1x per computer

```
## Installing the readxl package
install.packages('readxl')
```

Loading packages

1x per R session

```
## Loading the readxl package
library(readxl)
```

#3 Try it yourself!

Open this Week .R file (script) that you've downloaded from Canvas `(week1-lecture.R)`

What are

Logical operators?

Logical **Operators** & **Booleans**

- **Most programming languages work with logical comparisons - R is not different**

```
## Let's check: is one equal to one?

1 == 1

## Is five greater than 4?

5 > 4
```

```
## Is my_var not equal to 10?

my_var <- 8

my_var != 10
```

Logical **Operators** & **Booleans**

- **Some logical operators ...**

operator	definition
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
x & y	x AND y

Logical **Operators** & **Booleans**

operator	definition
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
x & y	x AND y

- **Some more logical operators ...**

operator	definition
x | y	x OR y
is.na(x)	test if x is NA
!is.na(x)	test if x is not NA
x %in% y	test if x is in y
!(x %in% y)	test if x is not in y
!x	not x

Compound Conditions

- **Sometimes we want R to evaluate more than one condition to check if a statement is true or false**

  Condition 1: 5 **>** 4

  Condition 2: 2 * 5 **>** 9

Simultaneously true ("& "operator)

```
5 > 4 & 2 * 5 > 9
```

At least one is true ("|" operator)

```
5 > 4 | 2 * 5 > 9
```

Data types
&
Vectors

Date **Types**

- **R handles uniquely each type of data...**

... and there are six different types of them. But we will be working with four mainly.

Character string (single or double quotes)

```
'Harris'
```

Date **Types**

- **R handles uniquely each type of data...**

... and there are six different types of them. But we will be working with four mainly.

Character string (single or double quotes)

```
'Harris'
```

Integer

```
120L (real values without decimals)
```

Date **Types**

- **R handles uniquely each type of data...**

... and there are six different types of them. But we will be working with four mainly.

Character string (single or double quotes)

```
'Harris'
```

Numeric

```
10, 10.55, 1/3
```

Integer

```
120L (real values without decimals)
```

Date **Types**

- **R handles uniquely each type of data...**

... and there are six different types of them. But we will be working with four mainly.

Character string (single or double quotes)

```
'Harris'
```

Numeric

```
10, 10.55, 1/3
```

Integer

```
120L (real values without decimals)
```

Logical

```
TRUE, FALSE, NA
```

Date **Types**

- **R handles uniquely each type of data...**

... and there are six different types of them. But we will be working with four mainly.

Character string (single or double quotes)

```
'Harris'
```

Numeric

```
10, 10.55, 1/3
```

Integer

```
120L (real values without decimals)
```

Logical

```
TRUE, FALSE, NA
```

- **Ok, but how can we tell them apart?**

Vectors

- **Most basic R data structure**

  The vector:

```
 [1] 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
```

Vectors

- **Most basic R data structure**

  The vector:

```
 [1] 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
```

- **Properties**
  - One dimension: length() function
  - Only holds homogenous data types: class () function

Vectors

- **Most basic R data structure**

  The vector:

```
 [1] 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
```

- **Properties**
  - `One dimension: length() function`
  - `Only holds homogenous data types: class () function`

- **Main way to create them: c() function** (the 'c' stands for combine)

- **Most functions in R will return their outputs in a vector format**

Doing operations with **vectors**

- **What if we want to combine logicals and vectors?**
- **Suppose we wanted to check which elements in our numeric vector are >= 15:**

```
# Creating the vector - method 1
my_vec <- c(10,11,12,13,14,15,16,17,18,19,20)

# Creating the vector - method 2
my_vec <- c(10:20)

# Checking elements greater or equal to 15:
my_vec >= 15
```

Subsetting **vectors**

- **What if we wanted to get only a few elements of our vector?**

vector[element]

```
# Creating the vector
my_vec <- c(10:20)

# Getting the first element of our vector
my_vec[1]

# Getting the last element of our vector
my_vec[10]

# Getting elements three, four and five
my_vec[c(3,4,5)]
```

#4 Try it yourself!

Open this Week .R file (script) that you've downloaded from Canvas `(week1-lecture.R)`

What about Week 2?

Week's 2 **Agenda**

1. Importing data into R
2. What are dataframes?
3. Tidyverse
4. Transforming our dataset
5. The **pipe** operator

Thank you!

Week 1

PPHA 30110 - Coding Lab for Public Policy