Module 1: An Introduction and Motivation for R Programming

Jacob Jameson

Summer 2021

Welcome!

- Why are we here?
- What are we going to do?
- A quick introduction to R

Teaching Members

- Instructor: Jacob Jameson
- Experience:
  - 2nd year MSCAPP student
  - Data Analyst at the Center for Health and the Social Sciences
  - Previous TA for the Booth course Data Analysis in R and Python
- Research Interests: I am interested in studying the quantitative techniques that are used for medical decision-making at the individual and collective level.

You will also have several TAs who will be helping you along the way

Why learn coding?

Many public policy jobs and the Harris curriculum rely on programming

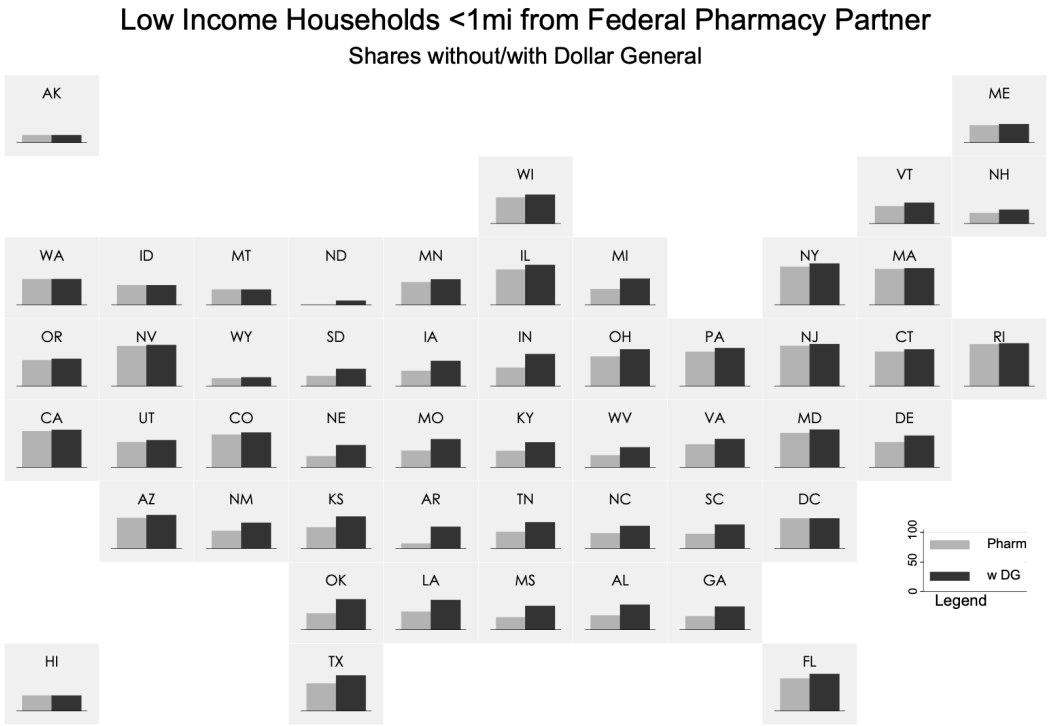
- to quickly engage with policy data
- to complete statistical analyses

Why R?

- Great data manipulation and visualization suite
- Strong statistical packages (e.g. program evaluation, machine learning)
- Complete programming language with low barriers to entry
- Open source and free

An Example

Quantifying the impact of a proposed federal partnership with the company Dollar General to serve as vaccination sites and comparing vaccine access with Dollar General to the current Federal Retail Pharmacy Partnership Program.



Notes: State-by-state data on the share of households earning less than $35K per year that are located less than a mile from a federal pharmacy partner. The grey bar represents the current pharmacy partners and the black bars add Dollar General as a partner.

What will we cover?

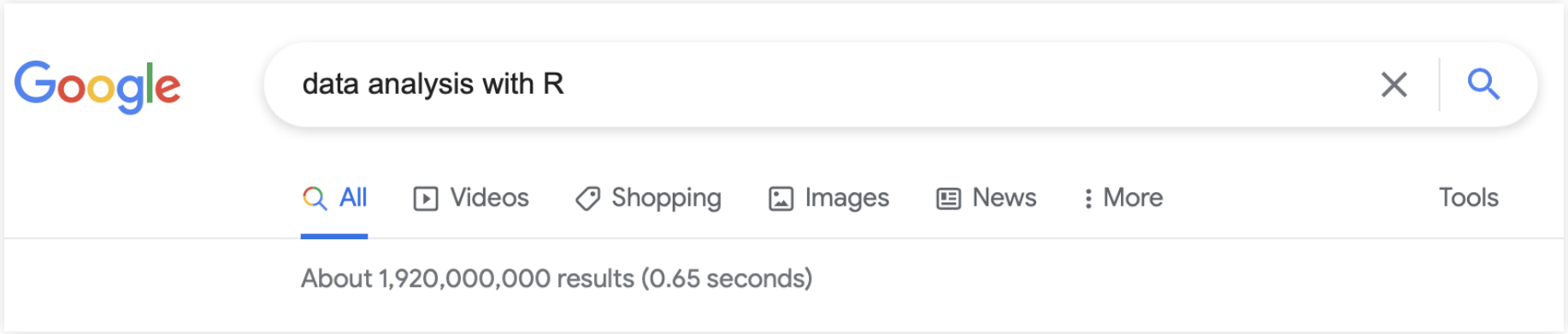
Coding Camp:

1. Motivation/Installation of R
2. Installing Packages and Reading Data
3. Basic Data Manipulation
4. Markdown Files, Knitting, and Data Structures
5. More on Data Manipulation and Analysis
6. Data Cleaning and Debugging

This is just the beginning of your programming journey!

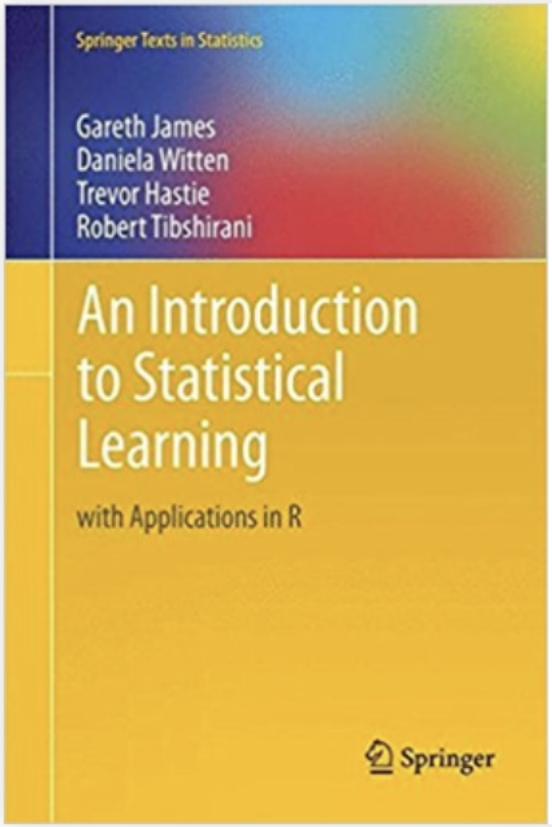
A Curated Tour

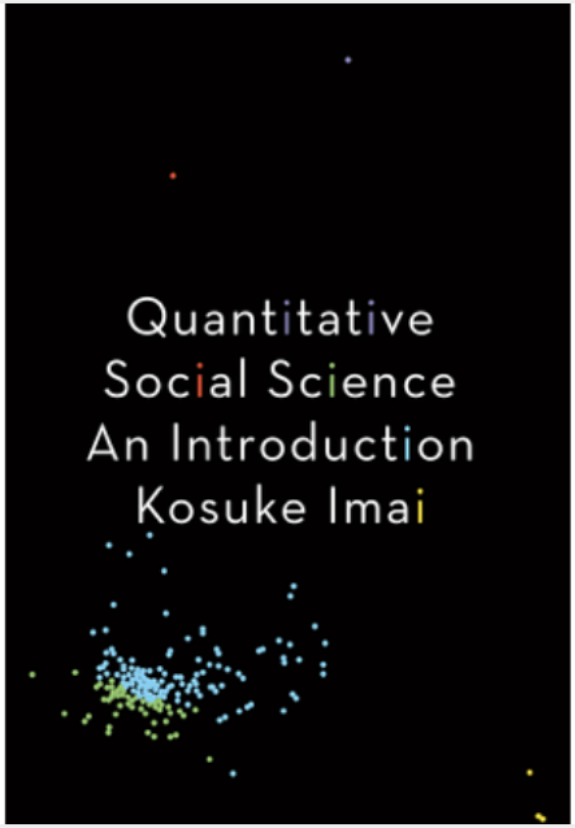
- There is a lot of information out there

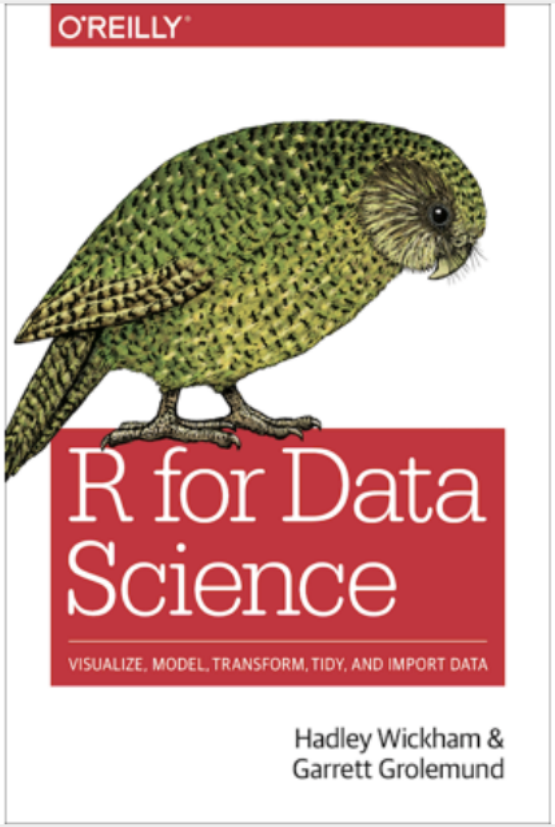


- We will only cover some of it
- If you think we are covering too much, also let us know

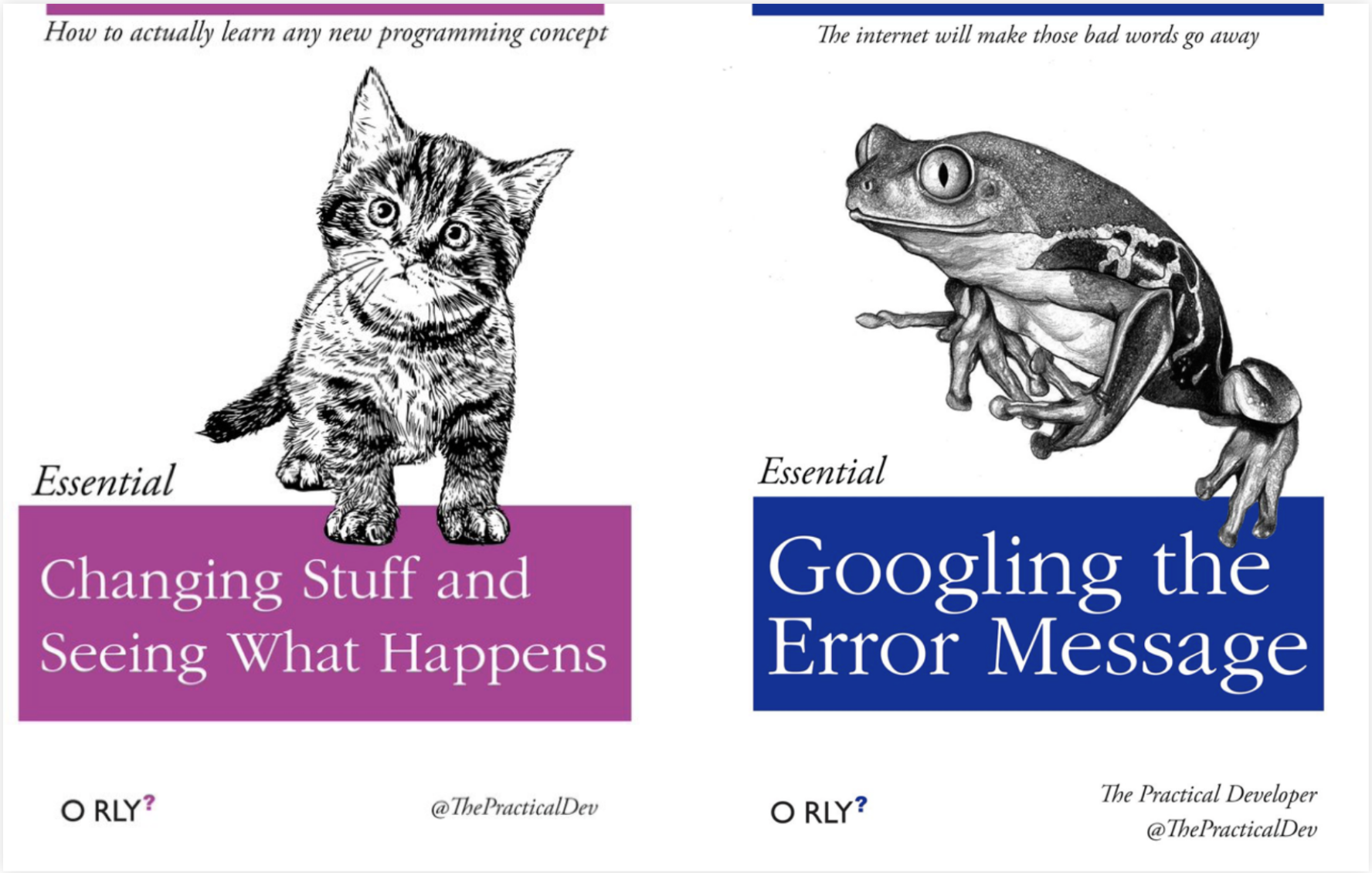
Textbooks and Resources



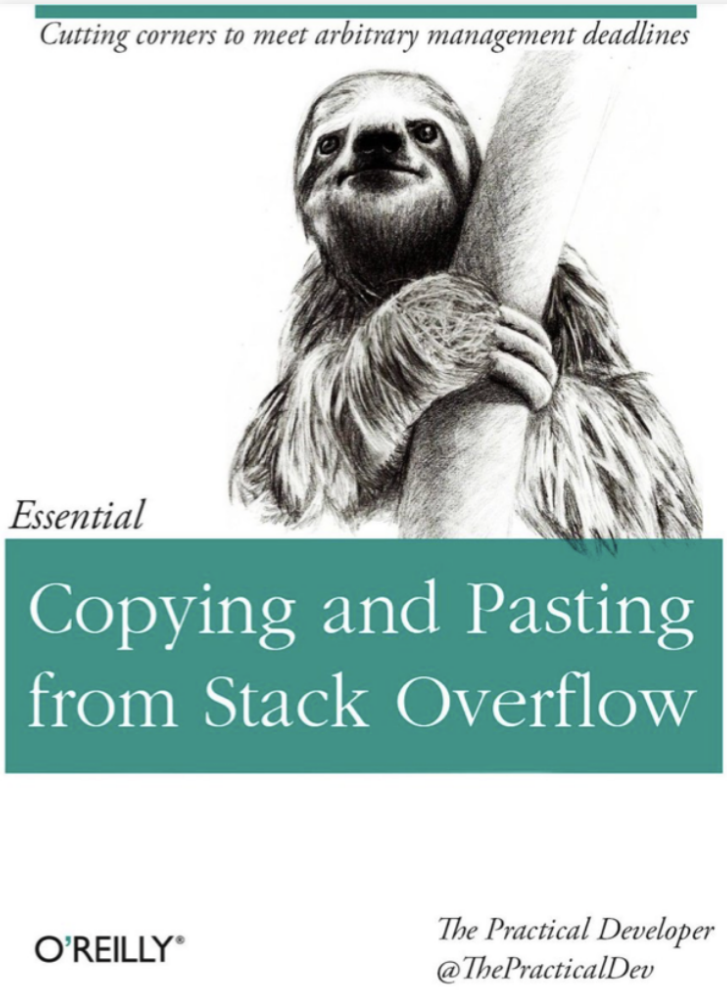


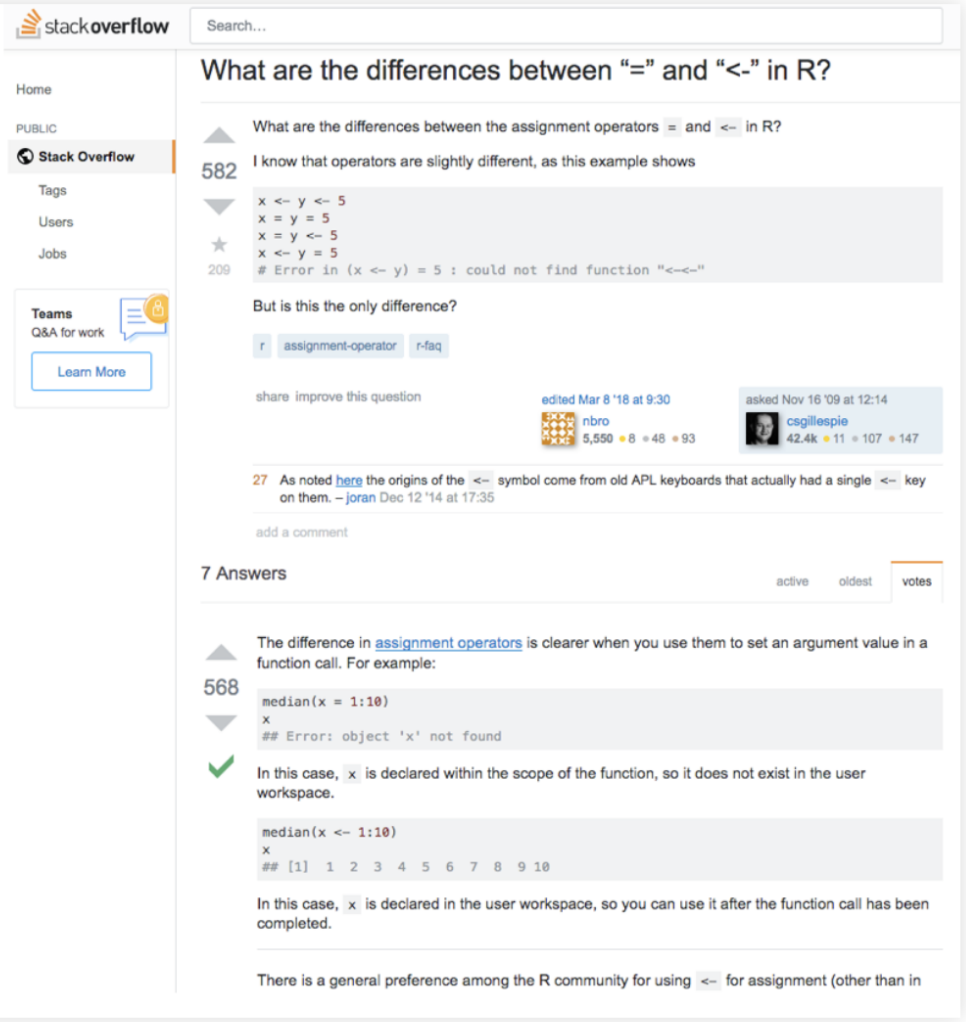


Textbooks and Resources



Textbooks and Resources





Learning philosophy

- We learn coding by experimenting with code.
- Coding requires a different modality of thinking
- Coding can be frustrating
- We develop self-sufficiency by learning where to get help and how to ask for help
- Coding lab is for you.

How will we progress?

- Video lectures:
  - Have R open. Pause regularly.
  - Focus on main idea first.
- Practice in labs (most important part):
  - You learn coding by coding.
  - Break up into small groups and work on problems with peer and TA support
- Q and A (live session):
  - Please send me questions ahead of class
  - May include additional practice problems.

A Quick Introduction to R and R Studio

Objectives

- Discuss what Rstudio is
- Introduce minimal information to get started working with R
- Expose you to the different data types in R
- Learn different types of operators

What is RStudio?

R Studio is an "integrated development environment" for R.

- It provides a console to access R directly.
- A text editor to write R scripts and work with Rmds
- An environment and history tab that provide useful information about what objects you have in your R session
- A help / plots / files / packages etc. section

If you have not already downloaded RStudio– do so now!

Let's Take a tour in Rstudio

Variables

We can think of a variable as a container with a name, such as `x`, `current_temperature`, or `subject_id` that contains one or more values. We can create a new variable and assign a value to it using = or ->.

```
x = 7
x
```

```
[1] 7
```

In the above bit of code, we created a variable called `x` and stored the value 7 inside of it.

Variables

We can think of a variable as a container with a name, such as `x`, `current_temperature`, or `subject_id` that contains one or more values. We can create a new variable and assign a value to it using = or ->.

```
my_number <- 5
my_number
```

```
[1] 5
```

In the above bit of code, we created a variable called `my_number` and stored the value 5 inside of it.

We can treat our variable like a regular number, and do arithmetic with it:

```
2 * x
```

```
[1] 14
```

```
my_number + 5
```

```
[1] 10
```

We can treat our variable like a regular number, and do arithmetic with it:

We can add comments to our code using the # character. It is useful to document our code in this way so that others (and us the next time we read it) have an easier time following what the code is doing.

```
#my_number is the variable I set equal to 5
my_number - 10
```

```
[1] -5
```

Atomic variable types in R

- character/string: “a”, “swc”
- numeric: 2, 15.5
- logical/boolean: TRUE (T), FALSE (F)
- special values: missing values (NA), impossible values (NaN), infinity (Inf), etc

Algebraic Operators

- +, -, *, and /. Also, %% (for mod), ^ (Exponentiation).

```
-7
7 + 5
7 - 5
7 %% 5
2^4
```

Operators can return special values

`Inf` is infinity. You can have either positive or negative infinity.

```
1/0
```

```
[1] Inf
```

```
2^1234
```

```
[1] Inf
```

`NaN` means Not a Number. It's an undefined value.

```
0/0
```

```
[1] NaN
```

Logical Operations

operator	definition	operator	definition
<	less than	x | y	x OR y
<=	less than or equal to	is.na(x)	test if x is NA
>	greater than	!is.na(x)	test if x is not NA
>=	greater than or equal to	x %in% y	test if x is in y
==	exactly equal to	!(x %in% y)	test if x is not in y
!=	not equal to	!x	not x
x & y	x AND y		

Comparison Operators

These are also binary operators; they take two objects, and give back a Boolean

```
7 > 5
7 < 5
7 >= 7
```

```
7 <= 5
7 == 5
7 != 5
```

Warning: == is a comparison operator, = is not!

`&` (and) and `|` (or)

- TRUE & FALSE -> FALSE
- TRUE & TRUE -> TRUE
- FALSE & FALSE -> FALSE
- TRUE | FALSE -> TRUE
- TRUE | TRUE -> TRUE
- FALSE | FALSE -> FALSE

```
(5 > 7) & (6 * 7 == 42)
```

```
[1] FALSE
```

```
(5 > 7) | (6 * 7 == 42)
```

```
[1] TRUE
```

```
(5 > 7) | (6 * 7 == 42) & (0 != 0)
```

```
[1] FALSE
```

Quiz

- What are the atomic (basic) data types in R?
- Will `(6 * 10 > 55 ) & (9 != 10-1) | (0 == 1)` return `TRUE`?

Recap: Introduction to R and RStudio

After going through this video, you should understand how to:

- navigate and use Rstudio's features
  - particularly, the console, the text editor and help
- assign objects to names with <- or =
- Understand the importance of data analysis in public policy

R Markdown

R Markdown is a file format for making dynamic documents with R.

An R Markdown document is written in markdown (an easy-to-write plain text format) and contains chunks of embedded R code, like the document below.

````
---
output: html_document
---

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more detai
ls on using R Markdown see .

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R co
de chunks within the document. You can embed an R code chunk like this:

```{r}
summary(cars)
```

You can also embed plots, for example:

```{r, echo=FALSE}
plot(cars)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
````