BIP Intro To R Workshop

BIP Summer Series

Dr. Joscelin Rocha-Hidalgo

This workshop will be recorded

If you don't want your image or voice to be part of the recording:

1. Keep your camera off
2. Keep yourself muted and ask questions in the chat/Q&A box only

Fill out this quick survey to build our dataset

05:00

Recording will start now!

Agenda

1. Logistics
2. Data Analysis Pipeline
3. Intro to R and Importing Datasets
4. Reproducibility & RMarkdown
5. Data Manipulation/Wrangling
6. Data Visualization

Who am I?

My name is Joscelin Rocha-Hidalgo.

I finished my Ph.D. at Georgetown University and currently working at Penn State University as a Post-Doctoral Scientist.

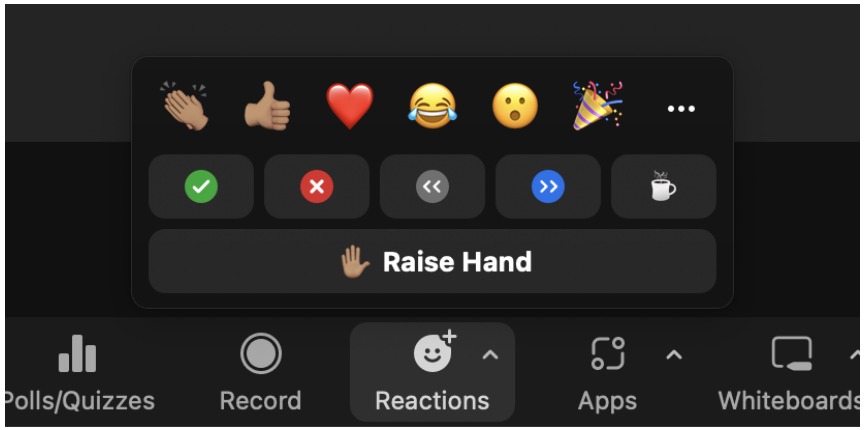
Twitter/X: @JoscelinRocha

email: JR1679@georgetown.edu

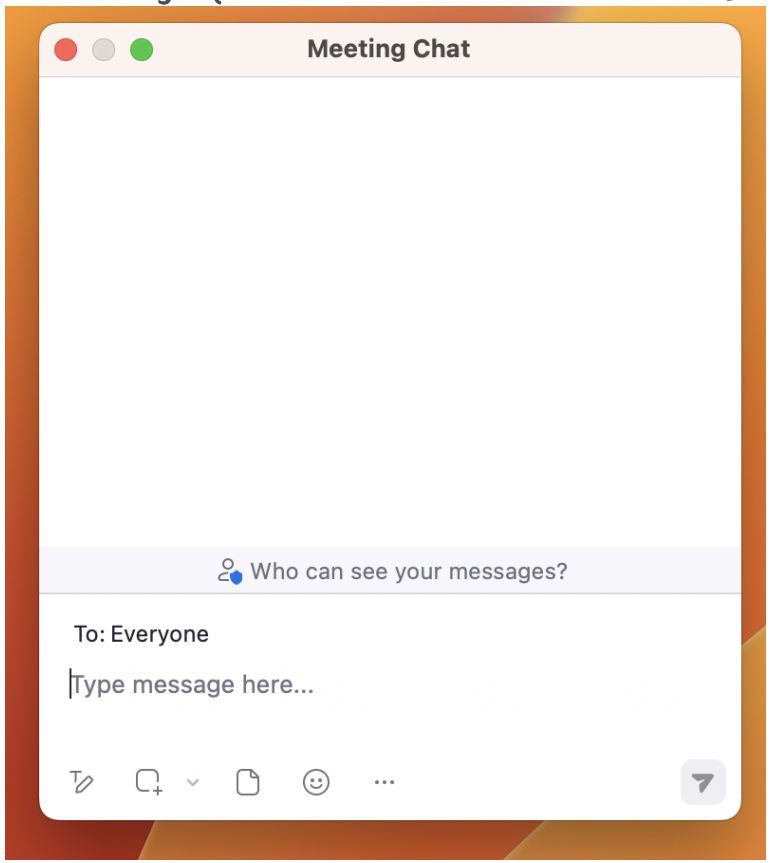
Logistics

Feel Free to:

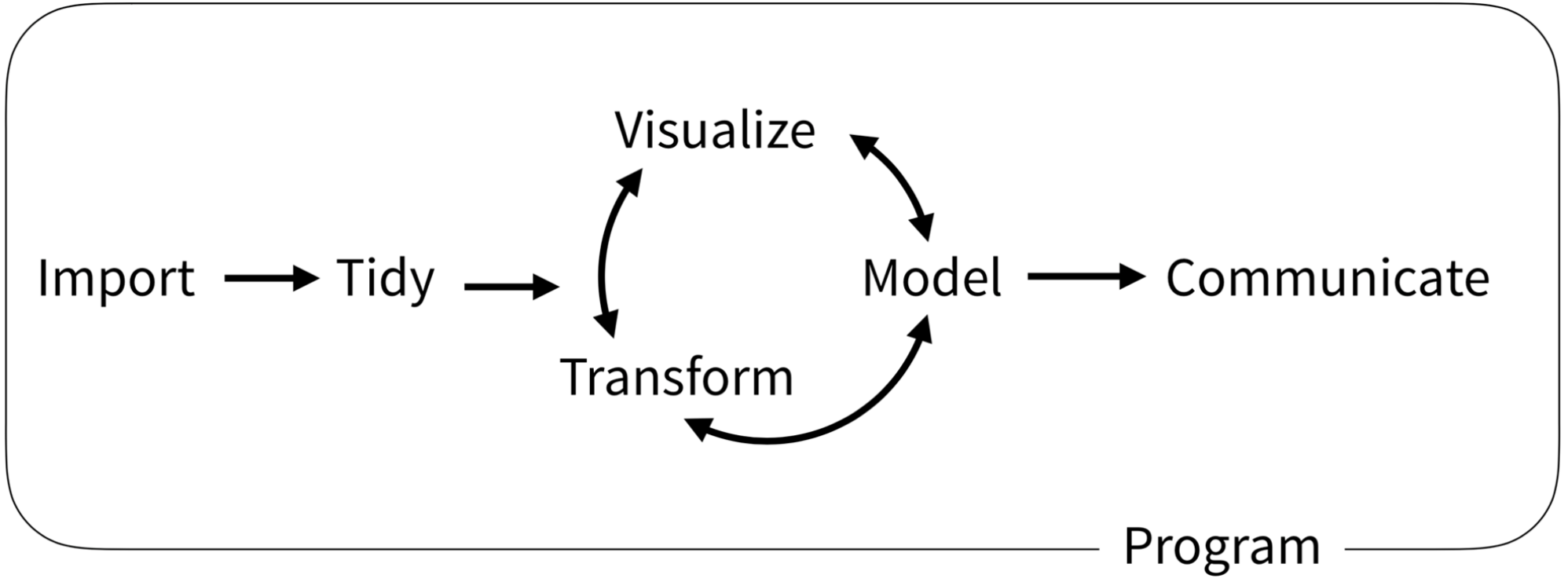
Provide nonverbal feedback or raise your hand



Ask any questions to the chat/Q&A box



Data Analysis Pipeline



From R for Data Science (https://r4ds.had.co.nz/introduction.html)

Introduction to R & Importing Datasets

R

Programming language

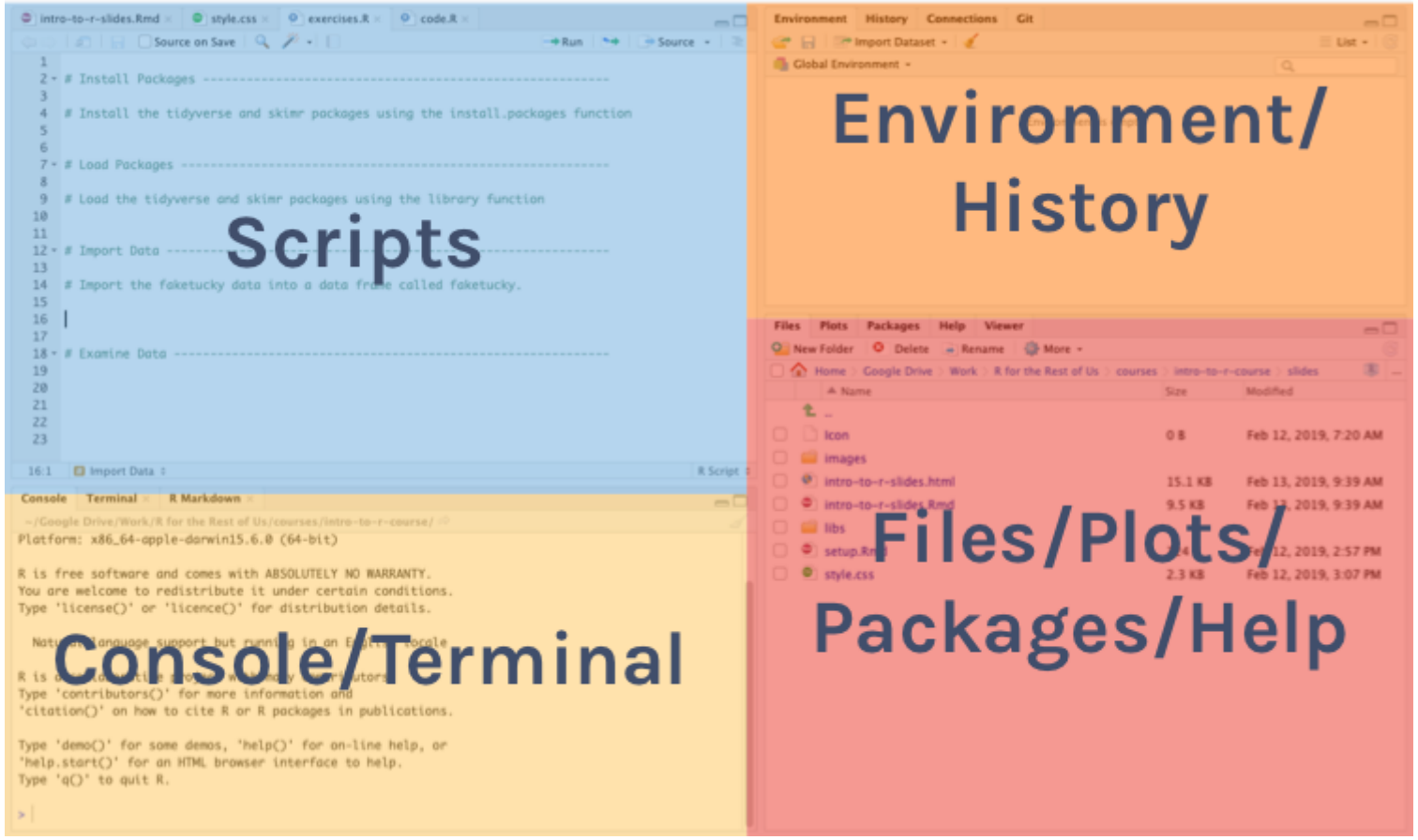
Besides the intuitiveness of this language, there is also a great community of folks using it for data science, and we are excited to welcome you all to the R community.

Rstudio

Interactive Development Environment (IDE)

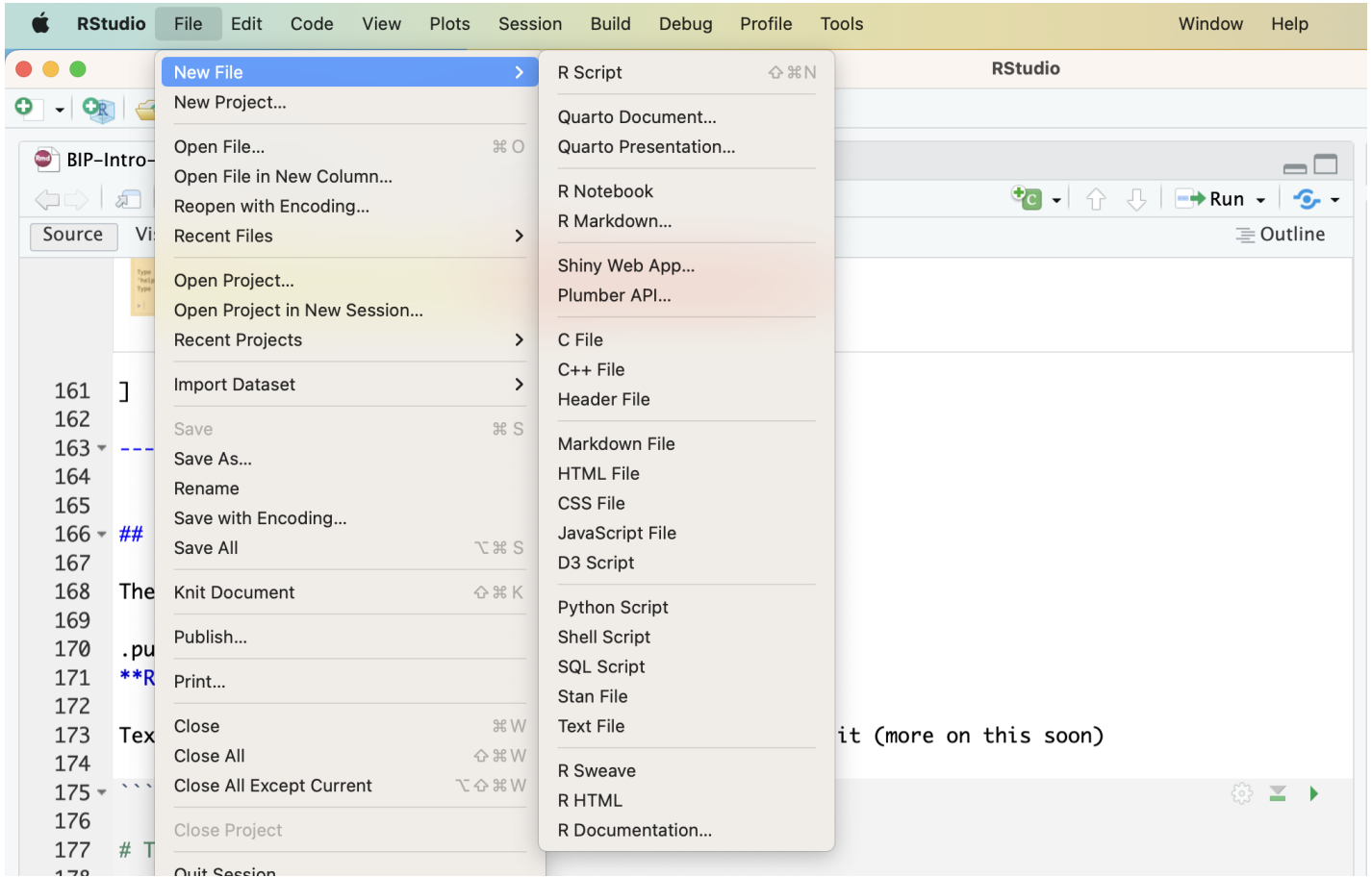
You can think of RStudio as a sophisticated text editor for writing R code. You can run RStudio on Mac, Windows, or on a server.

Tour of RStudio



File Types

There are multiple files types you could create using the Rstudio interface



File Types

But there are **two main file types** that you'll work with:

R scripts (.R)

Text is assumed to be executable R code unless you comment it (more on this soon)

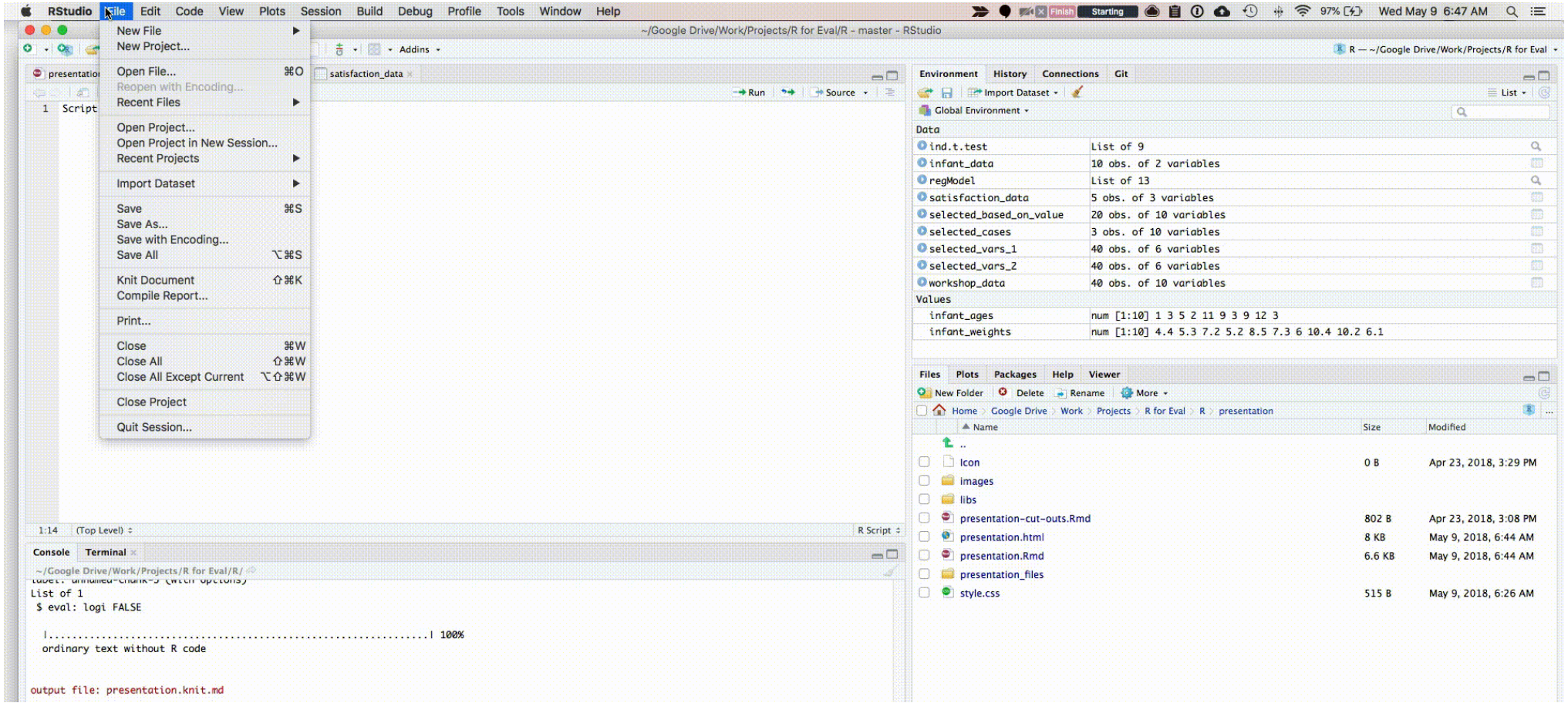
```
# This is a comment

data <- read_csv("data.csv")
```

RMarkdown files (.Rmd)

Text is assumed to be text unless you put it in a code chunk (more on this soon)

R Scripts

Create new script file: File -> New File -> R Script



How to Run Code

Run the code:

control + enter on Windows,

command + enter on Mac keystrokes or use Run button

Comments

Do them for others and for your future self.

```
# Show the first 5 rows of my data

head(data, n = 5)
```

Install Packages

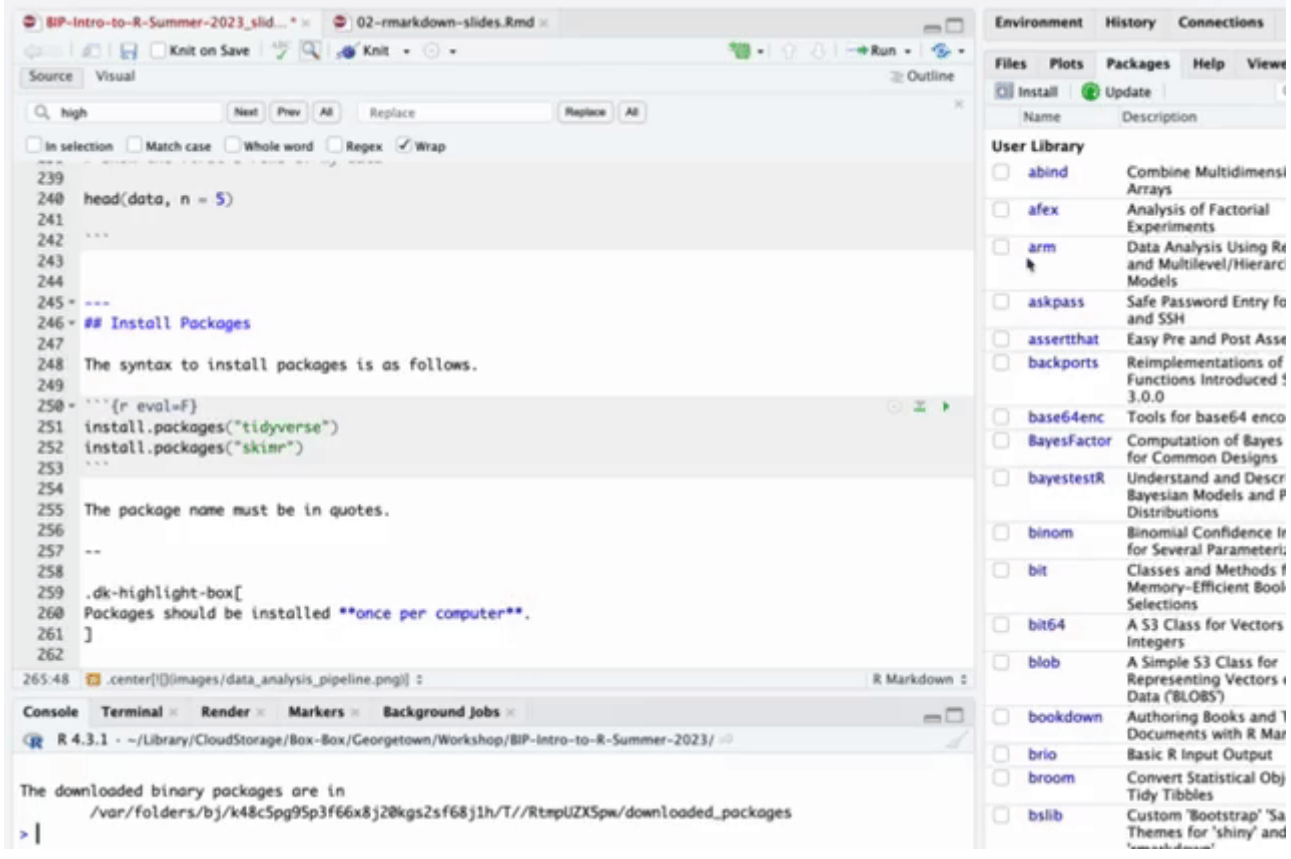
The syntax to install packages is as follows.

```
install.packages("tidyverse")
install.packages("skimr")
```

The package name must be in quotes.

Packages should be installed **once per computer**.

Another way to install packages



Load Packages

To load packages, use the following syntax:

```
library(tidyverse)
library(skimr)
```

Packages should be loaded **once per session**.

Quick Activity:

What Kind of File Type Do You Deal With the *Most*?

02:00

Import Data

CSV (Comma Separated Values) Files

```
# Using base R
data <- read.csv("file.csv")

# OR

# Using the readr package (part of the tidyverse)
library(readr)
data <- read_csv("file.csv")
```

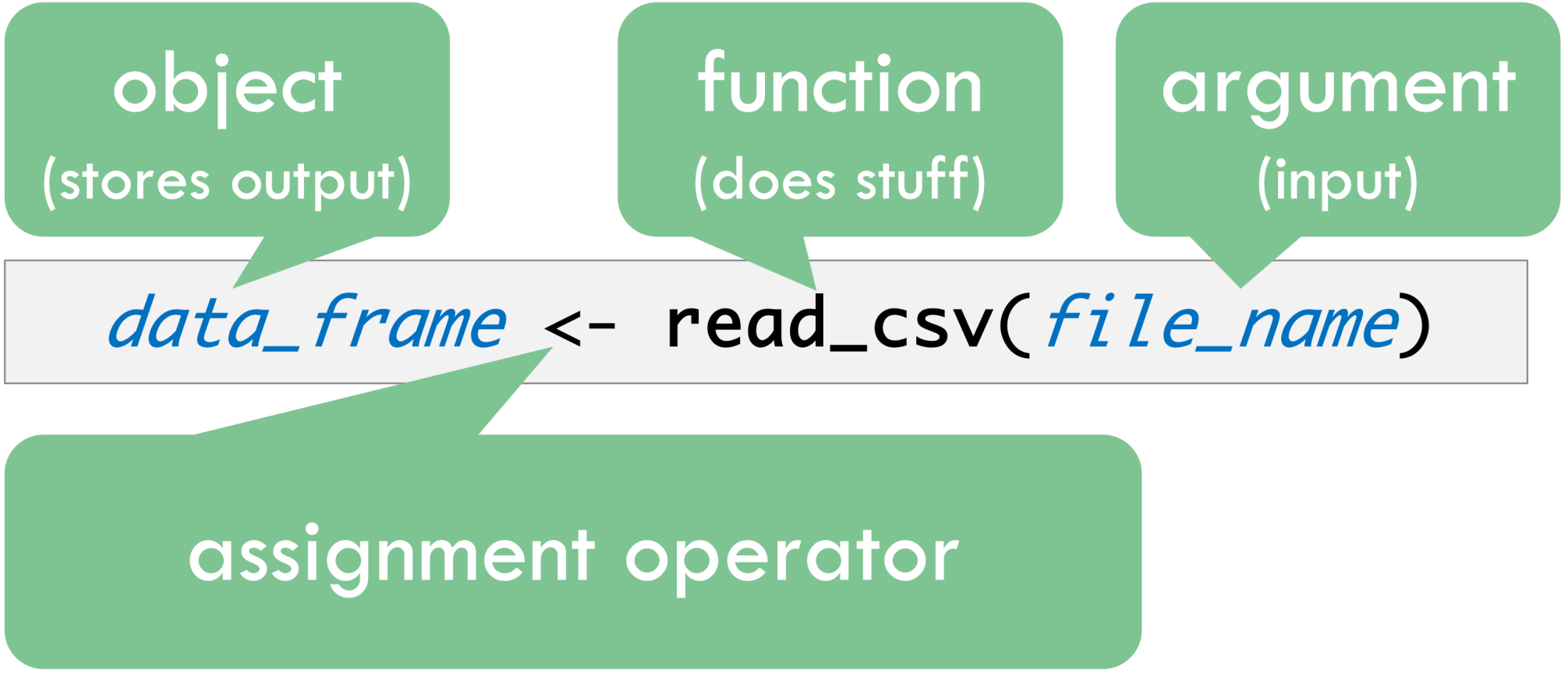
Let's explore this sintax together...

This code construct is exceedingly common in R, so I want to spend a few minutes exploring it together

read_csv() is a function. Remember that functions are defined in packages. We loaded the Tidyverse package to be able to use the read_csv function.

- The input that goes into a function is called an argument. The argument to a function gets put in parentheses.
- A function can have zero, one, or many arguments. If there is more than one argument, we use commas to separate them. We'll see examples of that later.

- The read_csv function outputs a data frame (think table), but if we want to *capture* that data frame inside of a named object, we need to specify that explicitly.
- It is a great idea to capture the output of a function in an object so it can be used as the input for other functions



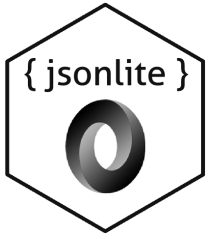
- To put the output of the read_csv function into a named object, we use the assignment operator.
- The assignment operator is a smaller-than sign followed by a dash or minus and looks kind of like an arrow pointing left.

What questions do you have so far?

00:10

Excel Files

```
# Using the readxl package (part of the tidyverse)
library(readxl)

data <- read_excel("file.xlsx")
```



JSON (JavaScript Object Notation) Files

```
# Using the jsonlite package
library(jsonlite)

data <- fromJSON("file.json")
```

XML (eXtensible Markup Language) Files

```
# Using the XML package
library(XML)

data <- xmlParse("file.xml")
```

SPSS Files

```
# Using the haven package (part of the tidyverse)
library(haven)

data <- read_sav("file.sav")
```

SAS Files

```
# Using the haven package (part of the tidyverse)
library(haven)

data <- read_sas("file.sas7bdat")
```

Stata Files

```
# Using the haven package (part of the tidyverse)
library(haven)

data <- read_dta("file.dta")
```

It can connect with multiple databases:

- Microsoft SQL Serve
- MonetDB
- MongoDB
- MySQL
- Netezza
- Oracle
- Amazon Redshift
- Apache Hive
- Apache Impala
- Athena
- Cassandra
- Google BigQuery
- Other Databases
- PostgreSQL
- SQLite
- Salesforce
- Teradata

https://solutions.posit.co/connections/db/databases/

SQLite Databases

```r
# Using the DBI and RSQLite packages
library(DBI)
library(RSQLite)

con <- dbConnect(RSQLite::SQLite(), "file.db")
data <- dbGetQuery(con, "SELECT * FROM table_name")
dbDisconnect(con)
```

R is Case Sensitive

R is **case sensitive** so choose one of the following for all objects and **be consistent**.

Option	Example
snake_case	student_data
camelCase	studentData
periods.in.names	student.data

Directories

If the data file is in the working directory, you only need to specify its name.

```
chds6162_data <- read_csv("chds6162_data.csv")
```

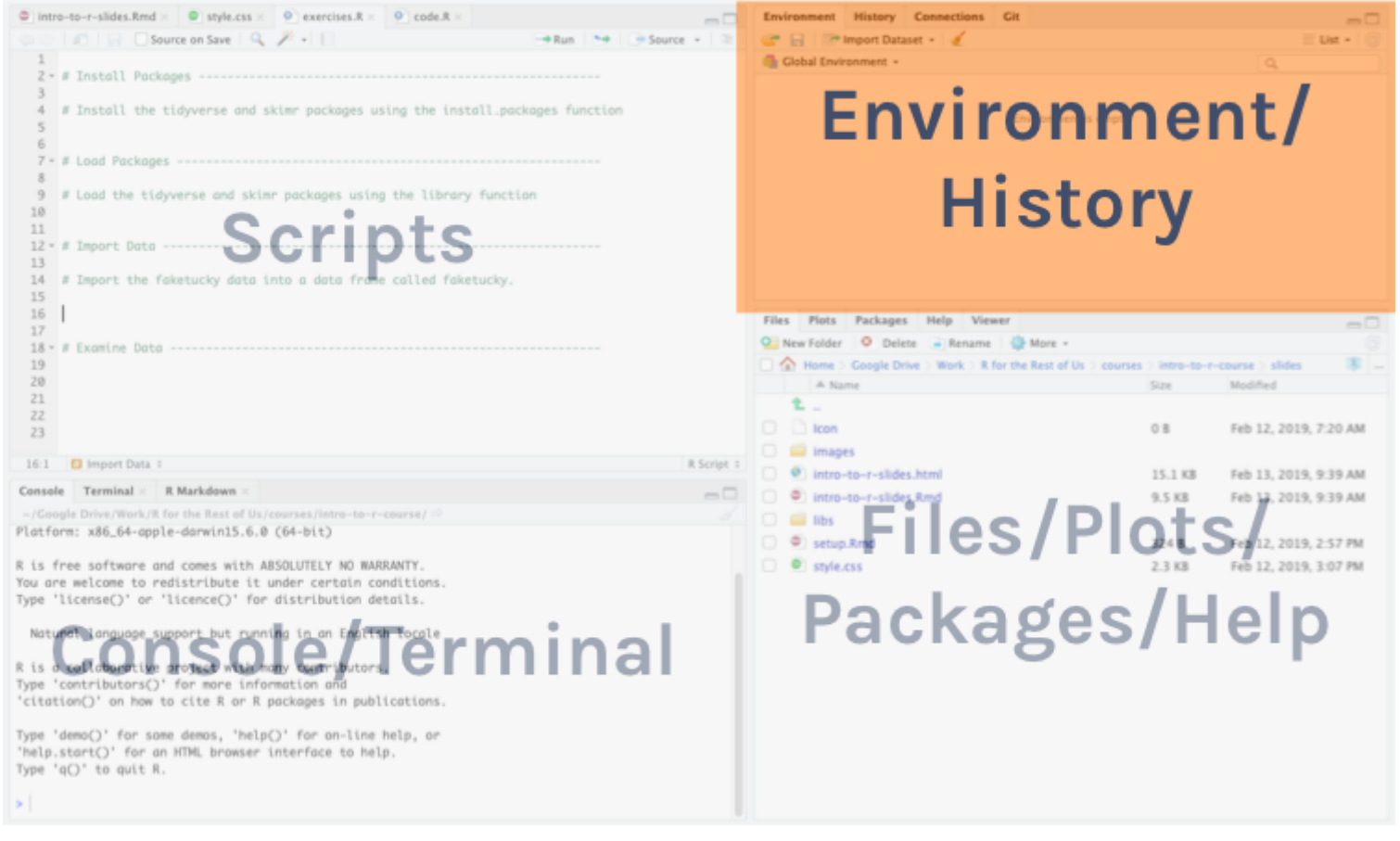
If the data file is not in the working directory, you need to specify full path name.

```
chds6162_data <- read_csv("data/chds6162_data.csv")
```

Using an RStudio project sets your working directory to the folder where your project lives so you only need to specify the location relative to that

Where Does our Data Live?

Data we have imported is available in the environment/history pane.



?function

Use the ? to get help about anything you're confused about

```
?read_csv
```

Recap

Recap

- **Packages** extend the functionality of R. They need to be installed once per computer and loaded each session.
- some Packages can be found in **CRAN (The Comprehensive R Archive Network)** and some need to be manually "activated" or "installed" to your device from github repos or other sites.
- **Functions** do stuff. They accept **Arguments** to define parameters. We can store the output of functions in **Objects** using the **assignment operator** (<-).
- **Importing Data** is the first step data analysis pipeline. **read_csv()** is a function from the **tidyverse** that we can use for importing data.

What questions do you have so far?

00:10

Reproducibility

Reproducibility & R

- One of the most powerful aspects of working in the R environment is the ability to conduct reproducible data analyses and data cleaning (those analyses and processes that can be shared, revised, repurposed and reproduced) by others.
- **Point-and-click is not reproducible**...unless...the interactive tool you are using also records those actions (**explore: JAMOVI or JASP**)

But it doesn't always help others...

Sometimes (most times) it helps YOU

Consider the following statements and ask yourself if they sound familiar:

- Can I redo the analysis from last month with this month's data?
- Why do the data in Table X not seem to agree with Figure X?
- Why did I decide to omit these 10 rows from my analysis?
- Why did i recode these variables this way?
- Which tests did my PI/Supervisor suggest to run and which ones did I come up with instead?

**Your closest collaborator is YOU from last month or week. You can do your self a huge favor by using tools that promote reproducibility.

Reproducibility & RMarkdown

RMarkdown (.rmd)

- R Markdown provides us with features and tools to tackle the reproducibility problem.
- In R Markdown, we can craft computer code mixed in with narrative annotation that documents the purpose of the code and details about the decisions we made in our analysis.

RMarkdown (.rmd)

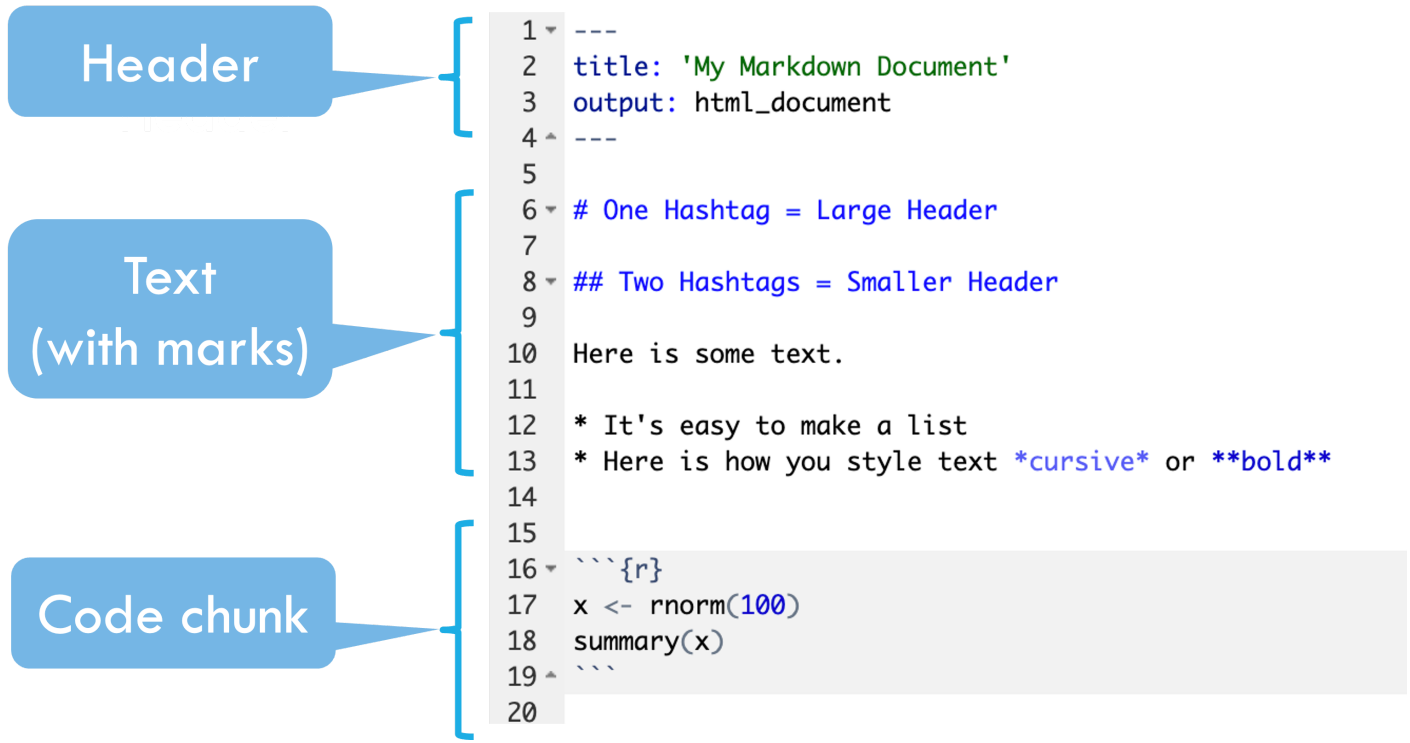
- R Markdown provides a lab notebook interface for analysis, visualization, and annotation of our work.
- It is quickly becoming the gold standard for reproducible data analysis.
- Today, I will teach you to use R Markdown and **encourage you to continue using it consistently in your future work**. In fact, This is my strongest recommendation today.

RMarkdown Overview

Every RMarkdown document has the following:

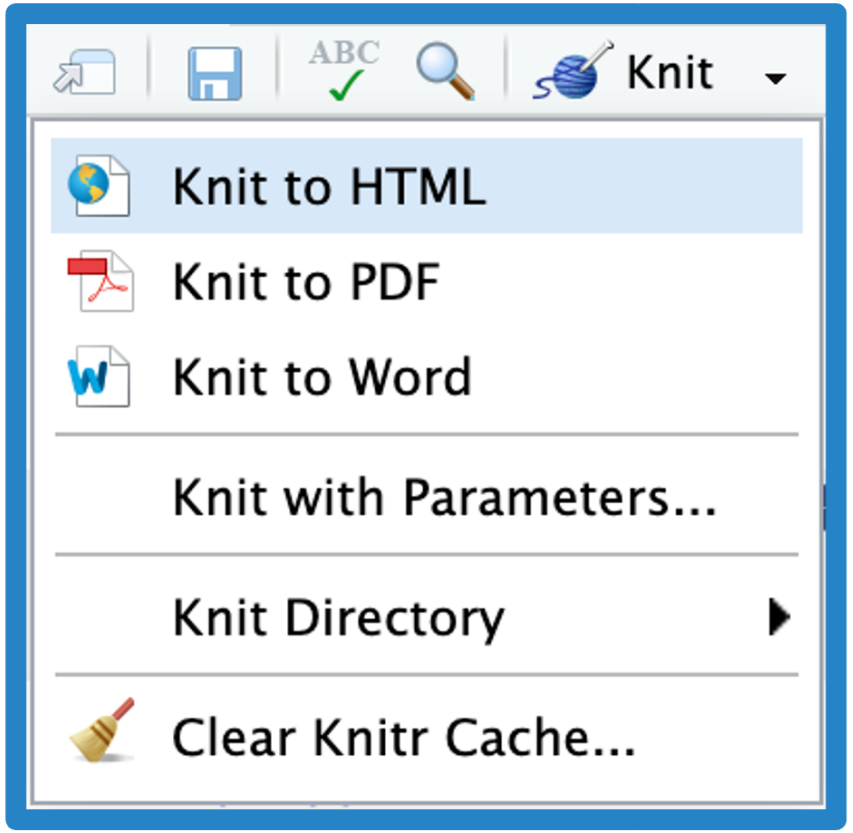
Or much easier put:

Anatomy of an R Markdown Document

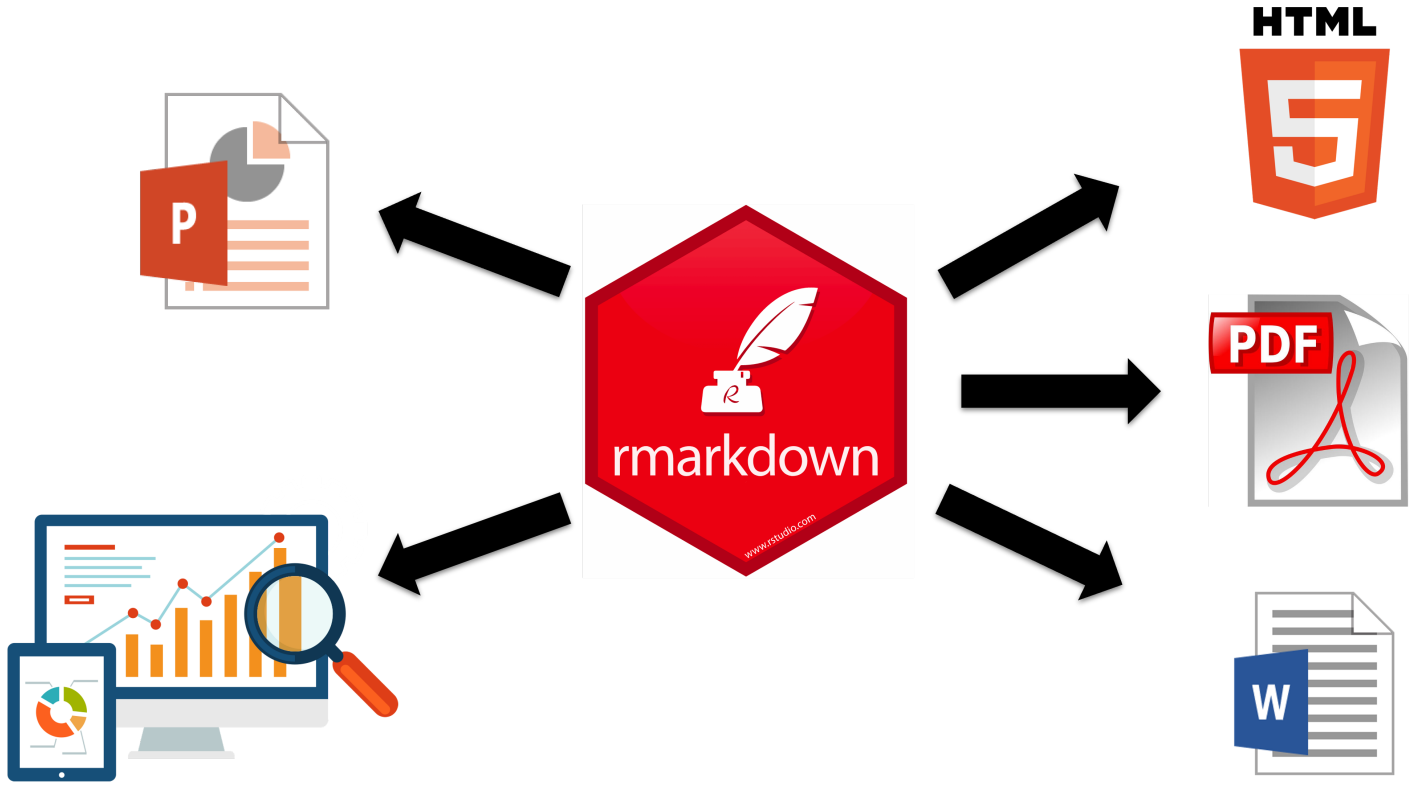


Knitting (aka Export)

```
1  ---
2  title: 'My Markdown Document'
3  output: html_document
4  ---
5
6  # One Hashtag = Large Header
7
8  ## Two Hashtags = Smaller Header
9
10 Here is some text.
11
12 * It's easy to make a list
13 * Here is how you style text *cursive* or **bold**
14
```



Knitting (aka Export)



In addition to HTML files, .rmd documents can be “knitted” into a number of additional formats including PDF, Microsoft Word, PowerPoint, and even interactive dashboards.

```
---
title: 'My Markdown Document'
output: html_document
---


# One Hashtag = Large Header

## Two Hashtags = Smaller Header

Here is some text.

* It's easy to make a list
* Here is how you style text *cursive* or **bold**


```{r}
x <- rnorm(100)
summary(x)
```


Including Plots

```{r, echo=FALSE}
hist(x)
```
```

My Markdown Document

One Hashtag = Large Header

Two Hashtags = Smaller Header

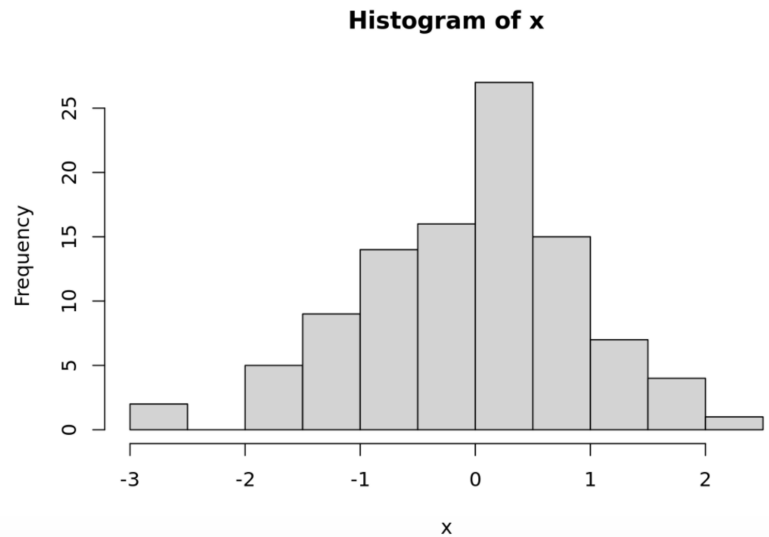
Here is some text.

- It's easy to make a list
- Here is how you style text *cursive* or **bold**

```
x <- rnorm(100)
summary(x)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.99204 -0.64726  0.14853 -0.02832  0.58218  2.07410
```

Including Plots



What questions do you have so far?

00:10

Anatomy of an RMarkdown Document

- The first is the **Header** or **YAML** which includes pieces like the name of the document, the document's author, and the desired output format when the document is assembled.
- Where you add title, author, date, output options, etc.
- The second block is the **text**. R Markdown documents can be marked in ways that promote readability with various formatting styles.
- Finally, there are **code chunks**. Code chunks include R code (or python, ruby, etc.) that can be executed to output results.

RMarkdown Documents have particular "rules" for how to format the Text sections

Markdown

```
Text with **some words in bold**
and *some words in italics*
```

Output

Text with **some words in bold** and *some words in italics*

Headers

Markdown

```
# First-Level Header

## Second-Level Header

### Third-Level Subheader
```

Output

First-Level Header

Second-Level Header

Third-Level Subheader

Lists

Markdown

```
- Unordered bulleted list item
- Unordered bulleted list item

1. Numbered list item
1. Numbered list item
```

Output

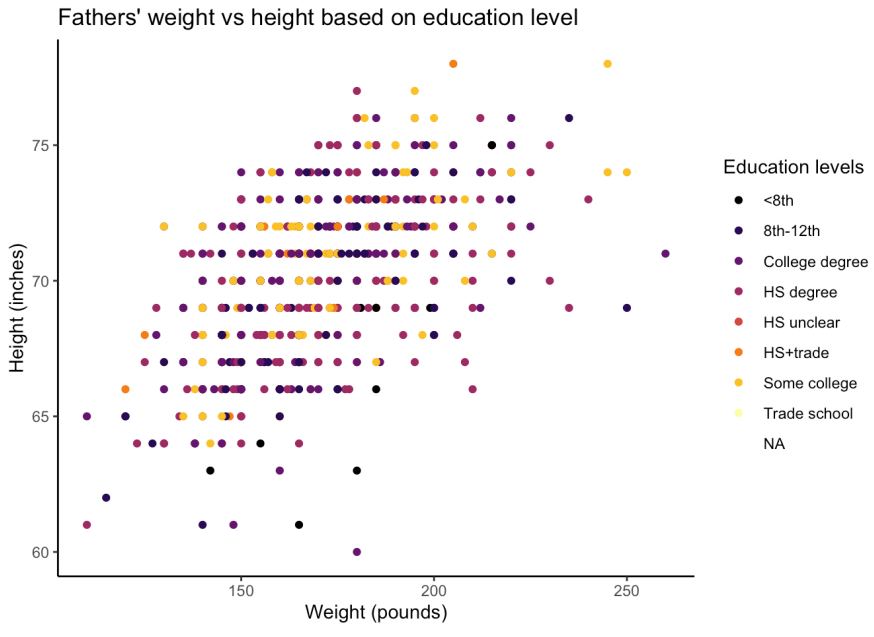
- Bulleted list item #1
- Bulleted list item #2

1. Numbered list item #1
2. Numbered list item #2

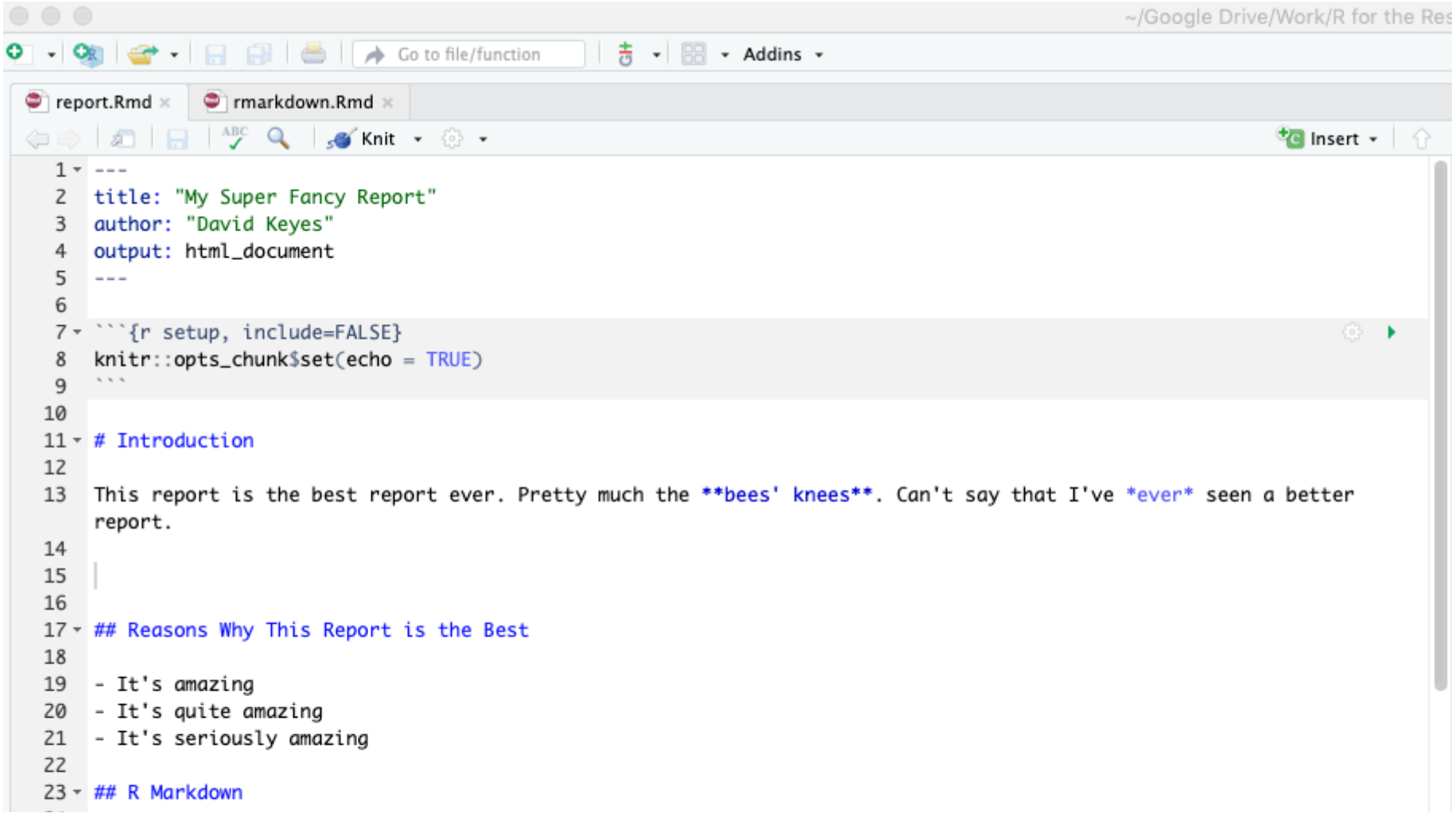
Code Chunk

They start with three backticks and {r} and end with three backticks.

```{r}
ggplot(data,aes(dwt,dht, color = ded_lbls)) +
  geom_point() + scale_color_viridis_d(option = "inferno") +
  labs (title = "Fathers' weight vs height based on education level",
        x = "Weight (pounds)",
        y = "Height (inches)",
        color = "Education levels") +
  theme_classic()

```



Insert a Code Chunk: Button



Insert a Code Chunk: Keyboard Shortcut

Mac

command+option+I

Windows

control+alt+I

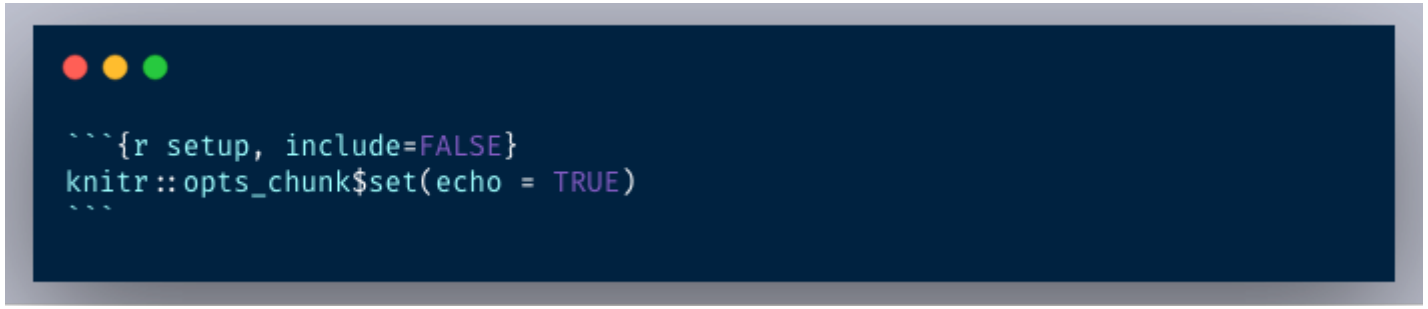
Chunk Options

Other options that we won't discuss today:

- **`warning`** (show any warnings that R throws)
- **`message`** (show any messages that R sends)
- **`fig.width`** (default figure width)
- **`fig.height`** (default figure height)
- **`echo`** (show the R code in the knitted report)
- and many more ...

Setup Code Chunk

A special code chunk with the text `setup` right after the `r`.



All chunk options can be set at the **global level** (in the setup code chunk) or at the **chunk level** (for individual chunks).

Options at the individual chunk level **override** global chunk options.

What questions do you have so far?

00:10

Data Manipulation

Remember this slide at the begining of the workshop?

Fill out this quick survey to build our dataset

That's our dataset for today. It has 12 variables we could clean:

variable	description
timestamp	Timestamp
books_2023	How many books have you read so far this 2023?
enjoy_cooking	On a scale of 1 to 10, how much do you enjoy cooking?
hrs_sleep	How many hours of sleep do you get on average per night?
superpower	Which superpower would you choose?
chosen_weekend_activity	If traveling was not an issue (you can teleport anywhere in a blink) What would be your preferred way to spend a free weekend?
least_fav_movie	What's your least favorite type of movie?
age_group	What is your age group?
gender	What is your gender?
state	Which state do you currently reside in?
race	What is your racial background?
latin_status	Are you Latino/a/e/x?

We will use packages from Tidyverse

Tidyverse Syntax

```
data %>%
  tabyl(age, ed)
```

- Did you notice I only referred to my dataframe once?
- The pipes! (%>%). They chain together a series of functions

The pipe

The pipe

I would read each pipe as "then." For example:

```
data %>%
  filter(age < 25) %>%
  group_by(ed) %>%
  summarize(mean_gestation = mean(gestation, na.rm = TRUE))
```

Keyboard shortcuts to add a pipe:

- command-shift-M (Mac)
- control-shift-M (Windows)

What questions do you have so far?

00:10

Let's create our mini report!