

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.

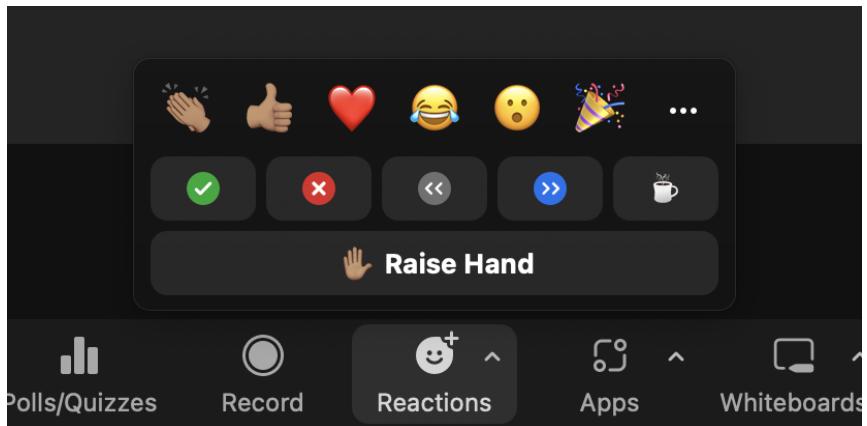
Twitter/X: @JoscelinRocha

email: JR1679@georgetown.edu

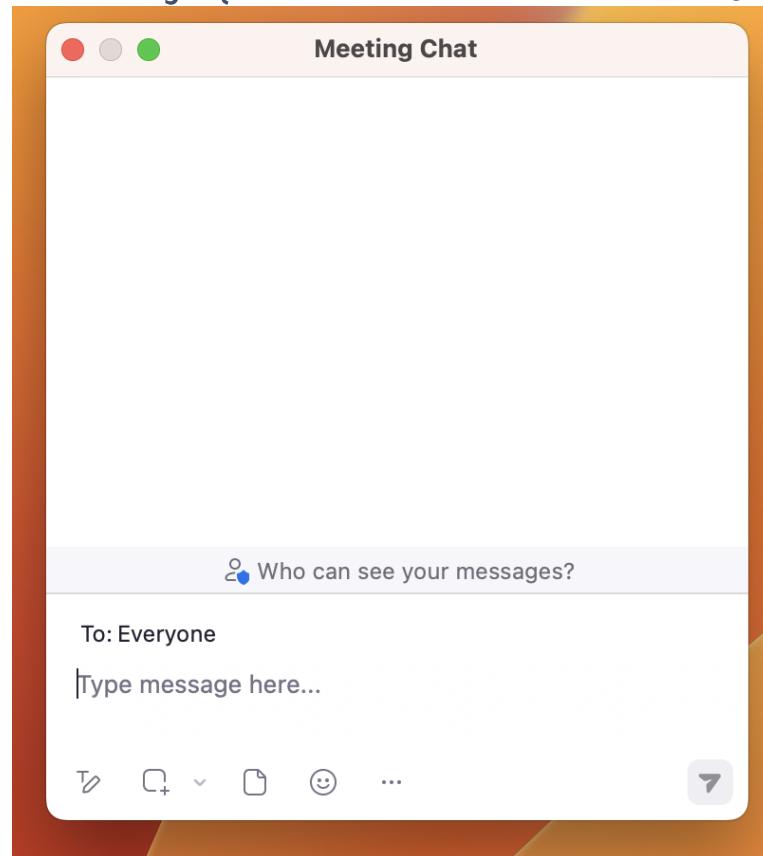
Logistics

Feel Free to:

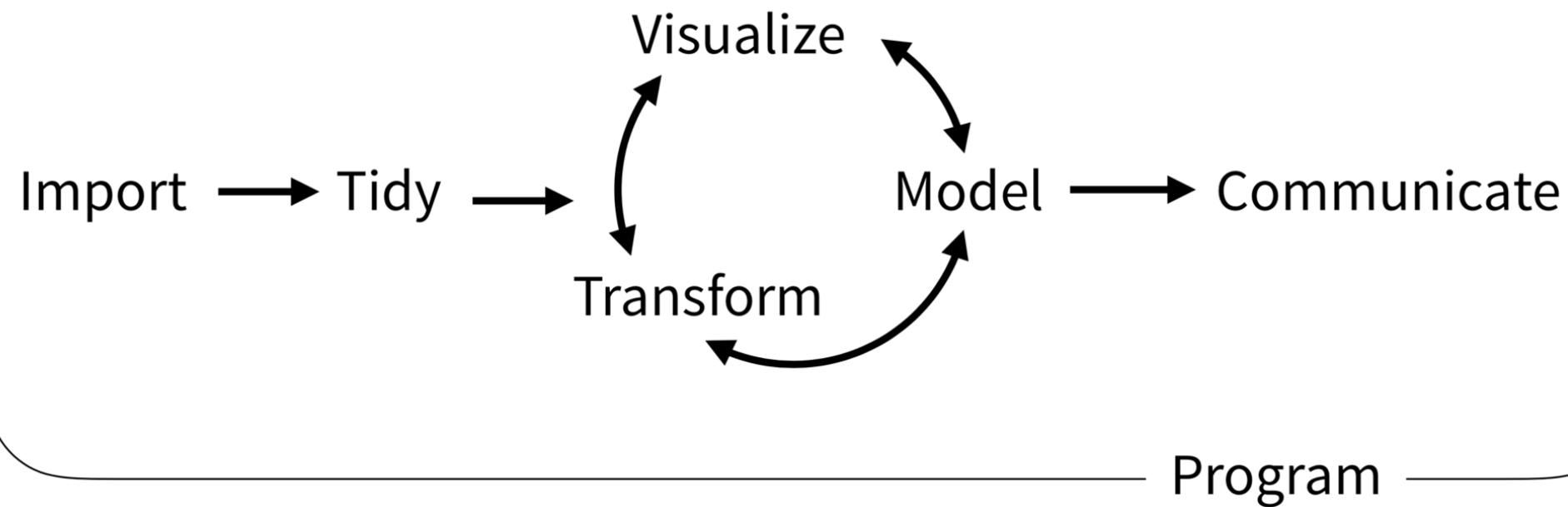
Provide nonverbal feedback or raise your hand



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.

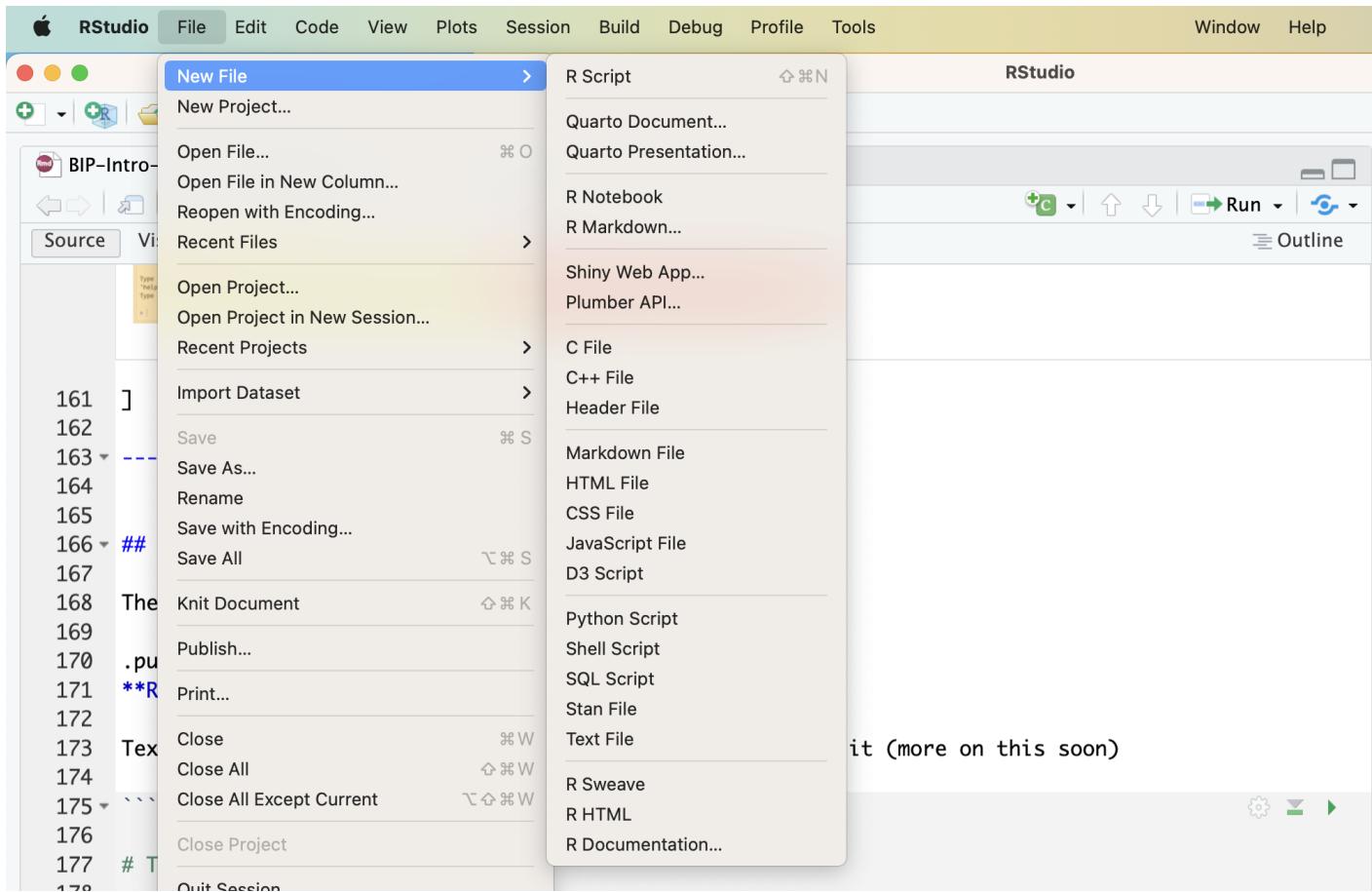
Tour of RStudio

The image shows the RStudio interface with four main panes:

- Scripts**: The top-left pane displays an R script with code for installing packages, loading tidyverse, importing faketyucky data, and examining data.
- Environment/History**: The top-right pane shows the Global Environment and a large title "Environment/History".
- Files/Plots/Packages/Help**: The bottom-right pane displays a file tree with files like intro-to-r-slides.html, intro-to-r-slides.Rmd, libs, setup.Rmd, and style.css, along with a large title "Files/Plots/Packages/Help".
- Console/Terminal**: The bottom-left pane shows the R startup message and a command prompt >.

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)

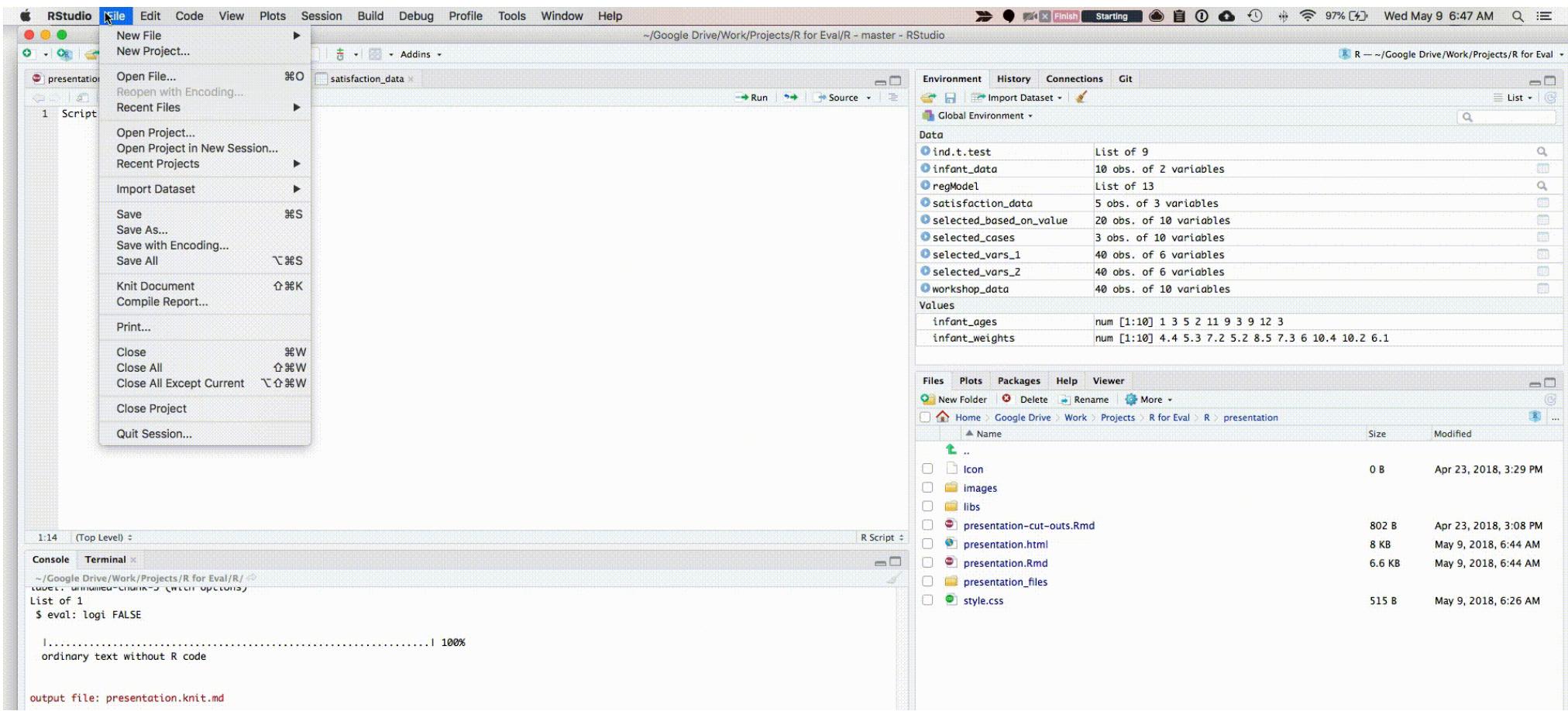
```
# This is a comment  
  
data <- read_csv("data.csv")
```

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

The screenshot shows the RStudio interface with the following details:

- Source View:** Displays R code. Lines 246-259 show code to install packages: `## Install Packages`, `install.packages("tidyverse")`, and `install.packages("skimr")`. Line 255 notes that package names must be in quotes.
- Console View:** Shows the output of the package installation command.
- File Browser:** Shows a list of packages in the User Library, including abind, afex, arm, askpass, assertthat, backports, base64enc, BayesFactor, bayestestR, binom, bit, bit64, blob, bookdown, brio, broom, and bslib.
- Environment View:** Shows the current environment variables.
- Plots View:** Shows a small image of a data analysis pipeline.
- Help View:** Shows help documentation for the selected package.
- Connections View:** Shows network connections.

```
239
240 head(data, n = 5)
241
242 ...
243
244
245 ...
246 ## Install Packages
247
248 The syntax to install packages is as follows.
249
250 ```{r eval=F}
251 install.packages("tidyverse")
252 install.packages("skimr")
253 ...
254
255 The package name must be in quotes.
256
257 --
258
259 .dk-highlight-box[
260 Packages should be installed **once per computer**.
261 ]
262
263:48 [1] .center[!](images/data_analysis_pipeline.png) :
R Markdown ::
```

R 4.3.1 - ~/Library/CloudStorage/Box-Box/Georgetown/Workshop/BIP-Intro-to-R-Summer-2023/

```
> | The downloaded binary packages are in
    /var/folders/bj/k48c5pg95p3f66x8j20kgs2sf68j1h/T//RtmpUZX5pw/downloaded_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 syntax together...

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

function
(does stuff)

```
data_frame <- read_csv(file_name)
```

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.

function
(does stuff)

argument
(input)

```
data_frame <- read_csv(file_name)
```

- 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.

object
(stores output)

function
(does stuff)

argument
(input)

```
data_frame <- read_csv(file_name)
```

- 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

object
(stores output)

function
(does stuff)

argument
(input)

```
data_frame <- read_csv(file_name)
```

assignment operator

- 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?

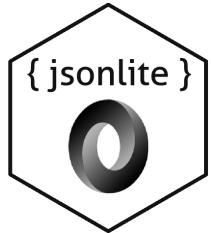
00:10



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

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

SQLite Databases

```
# 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

snake_case

camelCase

periods.in.names

Example

student_data

studentData

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.

The image shows a screenshot of the RStudio interface with several panes visible:

- Scripts**: Shows an R script with code for installing packages, loading tidyverse and skimr, importing faketucky data, and examining data.
- Environment/History**: Shows the Global Environment pane with the title "Environment/History".
- Console/Terminal**: Shows the R console output, including the R license and a welcome message about natural language support.
- Files/Plots/Packages/Help**: Shows the Files pane displaying files and folders related to the "intro-to-r-slides" project.

Name	Type	Size	Modified
Icon	Image	0 B	Feb 12, 2019, 7:20 AM
images	Folder		
intro-to-r-slides.html	HTML	15.1 KB	Feb 13, 2019, 9:39 AM
intro-to-r-slides.Rmd	Markdown	9.5 KB	Feb 12, 2019, 9:39 AM
libs	Folder		
setup.Rmd	Markdown	2.3 KB	Feb 12, 2019, 2:57 PM
style.css	Style Sheet		Feb 12, 2019, 3:07 PM

?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:

The screenshot displays a vertical stack of RMarkdown document components:

- YAML**: Configuration at the top of the file.
- Code Chunk**: An R code chunk starting with ````{r setup, include=FALSE}`.
- Text**: A descriptive text block about R Markdown.
- Code Chunk**: An R code chunk starting with ````{r cars}`.
- Text**: A descriptive text block about including plots.
- Code Chunk**: An R code chunk starting with ````{r pressure, echo=FALSE}`.

```
---
```

```
title: "My Super Fancy Report"
author: "David Keyes"
output: html_document
```

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

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

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

## Including Plots

You can also embed plots, for example:
```

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

Or much easier put:

Anatomy of an R Markdown Document

```
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  
15  
16 `r`  
17 x <- rnorm(100)  
18 summary(x)  
19  
20
```

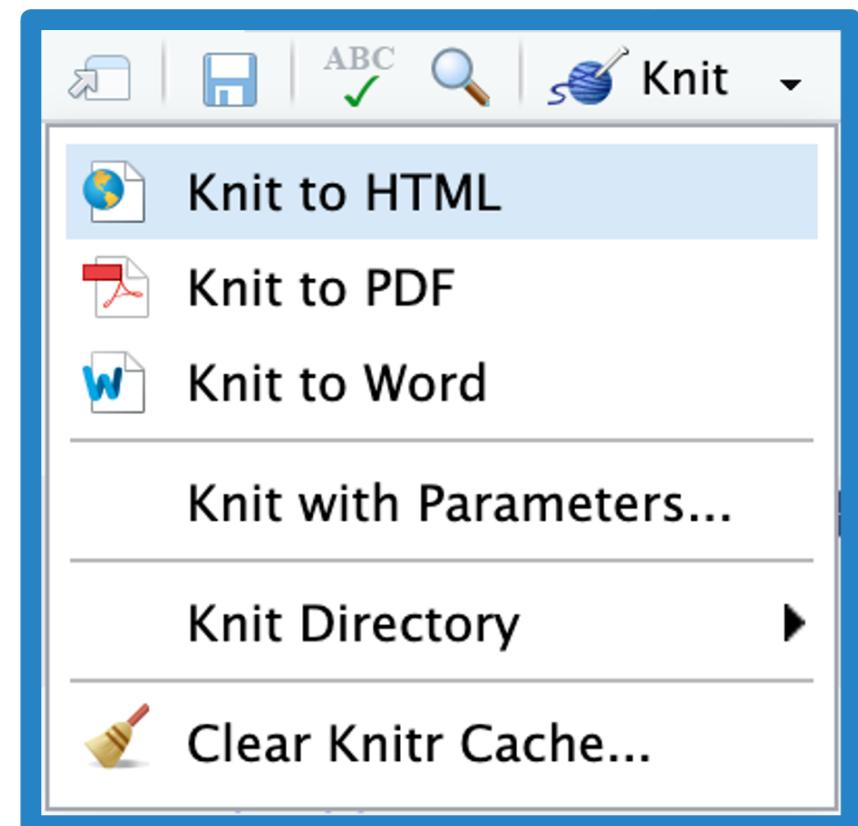
Header

Text
(with marks)

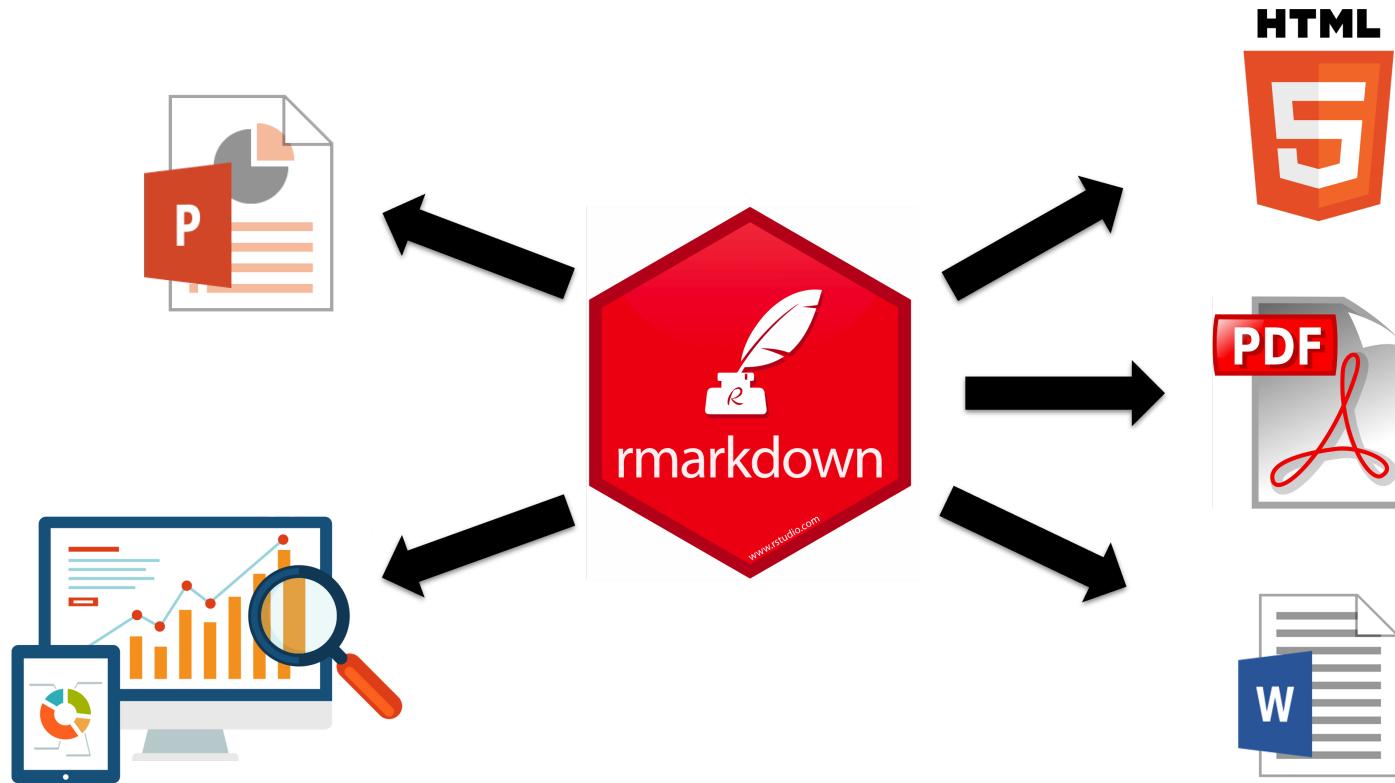
Code chunk

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.

```
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
```

```
15  
16 ```{r}  
17 x <- rnorm(100)  
18 summary(x)  
19 ``
```

```
20  
21 ## Including Plots|  
22  
23 ```{r, echo=FALSE}  
24 hist(x)  
25 ``
```

My Markdown Document

One Hashtag = Large Header

Two Hashtags = Smaller Header

Here is some text.

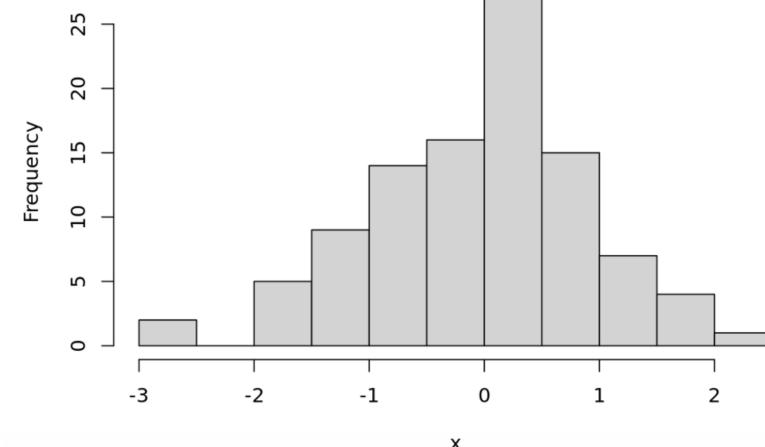
- It's easy to make a list
- Here is how you style text *cursive* or **bold**

```
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

Histogram of x



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

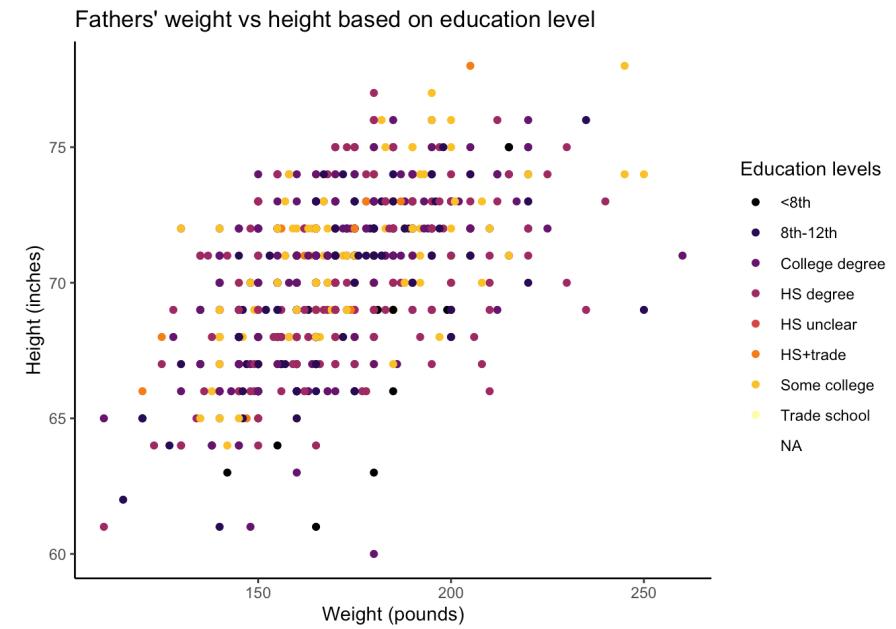
Output

- Bulleted list item #1
 - Bulleted list item #2
-
1. Numbered list item #1
 2. Numbered list item #2

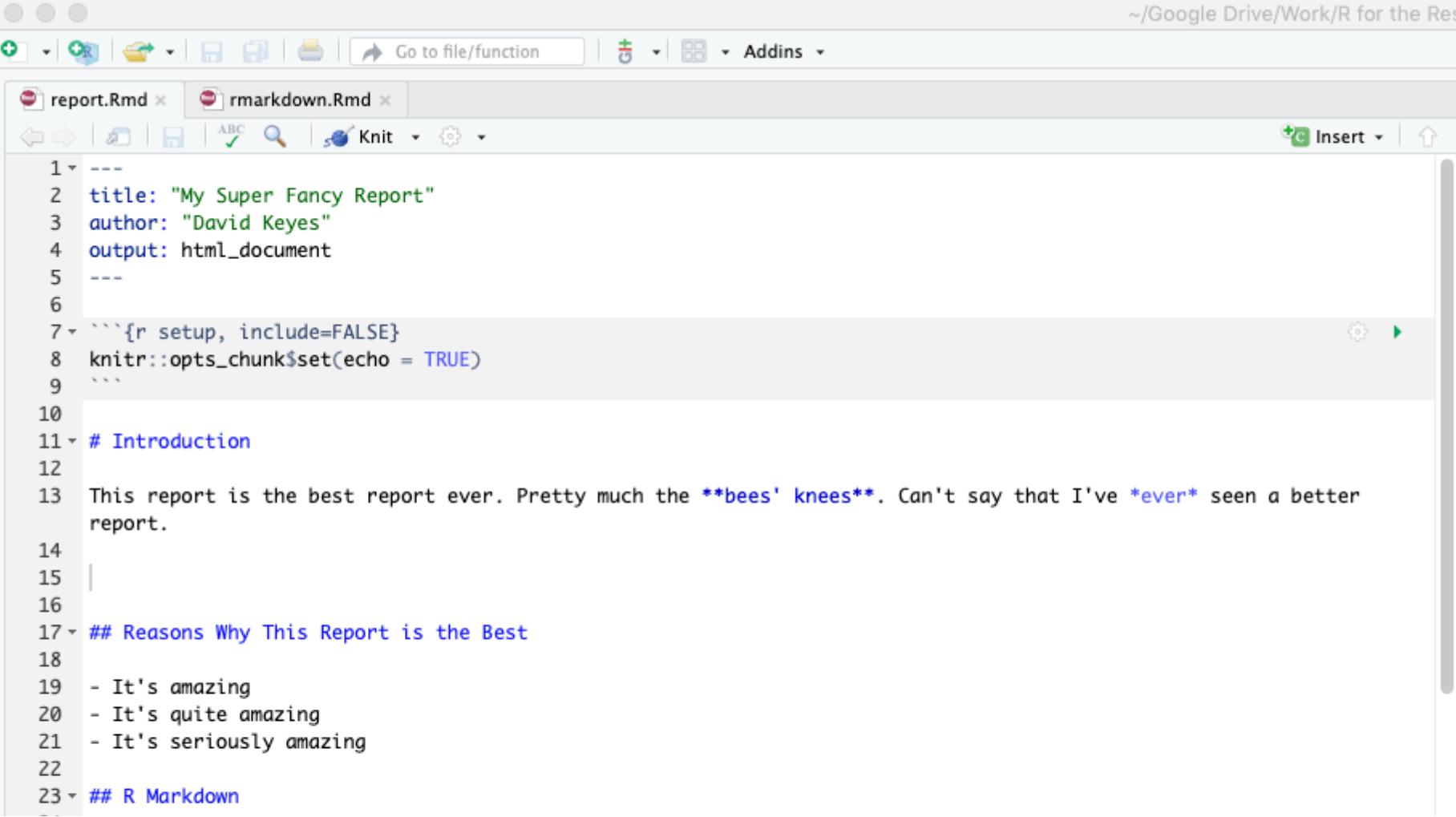
Code Chunk

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

```
154 ````{r}
155 ggplot(data,aes(dwt,dht, color = ded_lbls)) +
156   geom_point() + scale_color_viridis_d(option = "inferno") +
157   labs_(title = "Fathers' weight vs height based on education level",
158         x = "Weight (pounds)",
159         y = "Height (inches)",
160         color = "Education levels") +
161   theme_classic()
162 ``
163 ``
164 ````
```



Insert a Code Chunk: Button



The screenshot shows the RStudio interface with an R Markdown file open. The title bar indicates the path: ~/Google Drive/Work/R for the Rest of My Life. The top menu bar includes standard icons for file operations and a "Go to file/function" search bar. Below the menu is a toolbar with icons for file creation, saving, and navigation, along with a "Knit" button. The main workspace displays the following R Markdown code:

```
1 ---  
2 title: "My Super Fancy Report"  
3 author: "David Keyes"  
4 output: html_document  
5 ---  
6  
7 ```{r setup, include=FALSE}  
8 knitr::opts_chunk$set(echo = TRUE)  
9 ```  
10  
11 # Introduction  
12  
13 This report is the best report ever. Pretty much the bees' knees. Can't say that I've *ever* seen a better  
report.  
14  
15  
16  
17 ## Reasons Why This Report is the Best  
18  
19 - It's amazing  
20 - It's quite amazing  
21 - It's seriously amazing  
22  
23 ## R Markdown
```

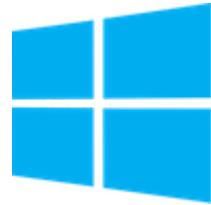
The code is color-coded: green for titles, blue for authors, red for the output type, and purple for the R chunk setup. The "## Reasons Why This Report is the Best" section is highlighted with a light gray background.

Insert a Code Chunk: Keyboard Shortcut



Mac

command+option+l



Windows

control+alt+l

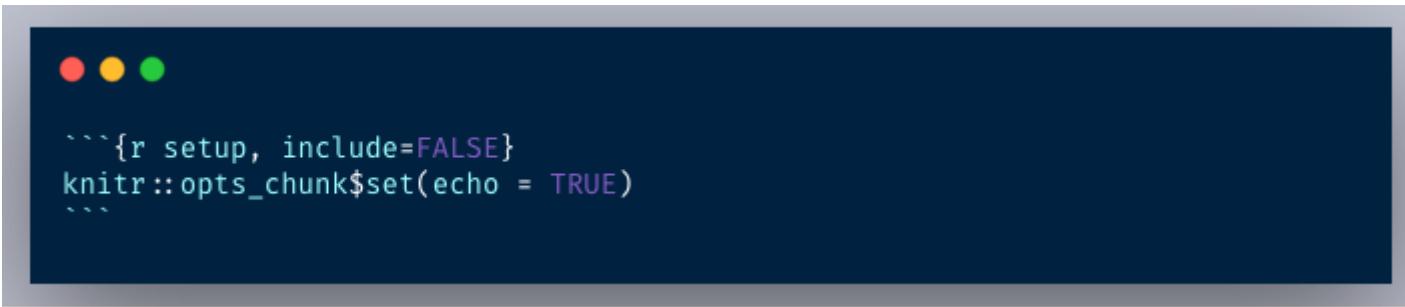
Chunk Options

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`.

A screenshot of an RStudio code editor window. The title bar shows 'RStudio' and 'Untitled.Rmd'. The main pane contains the following R code:```{r setup, include=FALSE}knitr::opts_chunk\$set(echo = TRUE)```The code is highlighted in white against a dark blue background. There are three small colored dots (red, yellow, green) in the top-left corner of the code area.

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 beginning
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?

If traveling was not an issue (you can teleport anywhere in a blink) |
| chosen_weekend_activity | 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



```
data %>%
  step1 %>%
  step2 %>%
  step3
```

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!