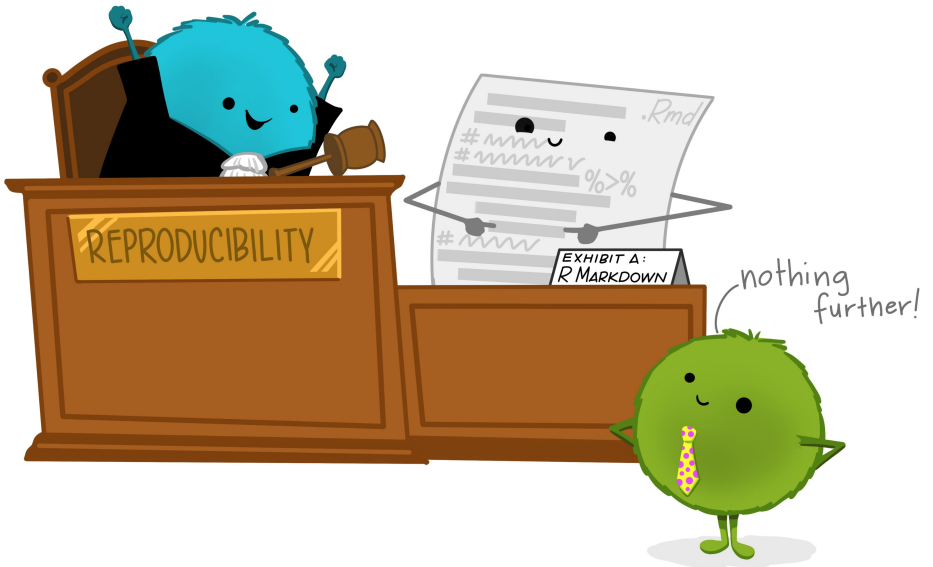
RMarkdown Basics

Recap

- Reproducibility across all sciences is a **huge** issue!
- We need to be able to reproduce scientific findings
    - This means the data & code from a given paper must be easy to access and **human readable**
- When it comes to programming in R, using **RMarkdown** can help us make sure that other humans can understand our analyses



Artwork by Allison Horst

Anatomy of RNotebook

The anatomy of all `.Rmd` files (RNotebook or RMarkdown):

1. **Document Information**
2. **Formatted Text**
3. **Code Chunks**

Anatomy of RNotebook

The anatomy of all `.Rmd` files (RNotebook or RMarkdown):

1. Document Information
2. Formatted Text
3. Code Chunks

Document Information

- Title
- Subtitle
- Date
- Author
- Output type

```
---
title: "Making Pretty Code"
subtitle: "with RMarkdown"
author: "Shelly Cooper"
output: html_document
---
```

This is sometimes called a YAML header.

Document Information

- Title
- Subtitle
- Date
- Author
- Output type

```
---
title: "Making Pretty Code"
author: "Shelly Cooper"
subtitle: with RMarkdown
output:
  pdf_document: default
  html_document: default
---
```

IMPORTANT: Do NOT change any of the tabs/indents in the YAML header. If you copy something from the internet, pay attention to this!

Anatomy of RNotebook

The anatomy of all `.Rmd` files (RNotebook or RMarkdown):

1. Document Information

2. Formatted Text

3. Code Chunks

Formatted Text

You need to tell R how you want your text to be formatted:

- Headers
- Bolded text
- Italicized text
- Hyperlinks
- Bullet/numbered lists

Headers

The number of # indicates what size and level your header should be.

```
# Header 1
```

Header 1

```
## Header 2
```

Header 2

```
### Header 3
```

Header 3

```
#### Header 4
```

Header 4

```
##### Header 5
```

Header 5

Headers

```
# Introduction
Background information goes here.

# Methods

### Participant Demographics
The current study looked at a sample of healthy adults (ages 18-80) across the
lifespan.

### Statistical Analyses
We first looked at descriptive statistics. Then we ran a multiple regression to look
at how three independent variables impacted a dependent variable.

# Results

### Descriptive Results
Go here

### Multiple Regression Results
Go here

# Discussion
Our study was better than yours. This paper has been accepted without needing any
revisions!
```

Headers

Introduction

Background information goes here.

Methods

Participant Demographics

The current study looked at a sample of healthy adults (ages 18-80) across the lifespan.

Statistical Analyses

We first looked at descriptive statistics. Then we ran a multiple regression to look at how three independent variables impacted a dependent variable.

Results

Descriptive Results

Go here

Multiple Regression Results

Go here

Discussion

Our study was better than yours. This paper has been accepted without needing any revisions!

Bold & Italics

Bold text

- **bold text **
- __bold text __ (2 underscores)

Italicized text

- * italicized text *
- _ italicized text _

To ***combine***, pick * for one and _ for the other

- **_combine_**
- _**combine**_
- *__combine__*
- __*combine*__

Hyperlinks

- The word(s) you want to be the link go inside square brackets `[ ]`
- Immediately after, it's a pair of parentheses `( )` that contains the actual link.

`[Google](www.google.com) is my friend!`

[Google](www.google.com) is my friend!

Bullet Lists - Unordered

- First line must end with a `:` (colon)
- Must have an empty line
- Must have a space after the bullet

Unformatted

```
Brazilian States:

- Rio Grande do Sul
- Parana
- Rio de Janeiro
```

Formatted

Brazilian States:

- Rio Grande do Sul
- Parana
- Rio de Janeiro

Bullets can be `-` (dashes), `+` (plus), or `*` (asterisk), but all come out looking like what you see here.

Bullet Lists - Ordered

Same thing, but now with numbers

Unformatted

```
Brazilian States:

1. Rio Grande do Sul
2. Parana
3. Rio de Janeiro
```

Formatted

Brazilian States:

1. Rio Grande do Sul
2. Parana
3. Rio de Janeiro

Bullet Lists - Nested

You can have organized, nested lists. Go to the next line, and press **2 spaces**. Then put your new bullet symbol.

- Do NOT press `tab`. For whatever reason, R doesn't like it for Markdown.
- If you still are stuck, try 4 spaces -- that should work

Unformatted

```
Brazilian States & Capitals:

1. Rio Grande do Sul
    * Porto Alegre
        - This is the best!
2. Parana
    * Curitiba
3. Rio de Janeiro
    * Rio de Janeiro
```

Formatted

Brazilian States & Capitals:

1. Rio Grande do Sul
    - Porto Alegre
        - This is the best!
2. Parana
    - Curitiba
3. Rio de Janeiro
    - Rio de Janeiro

Anatomy of RNotebook

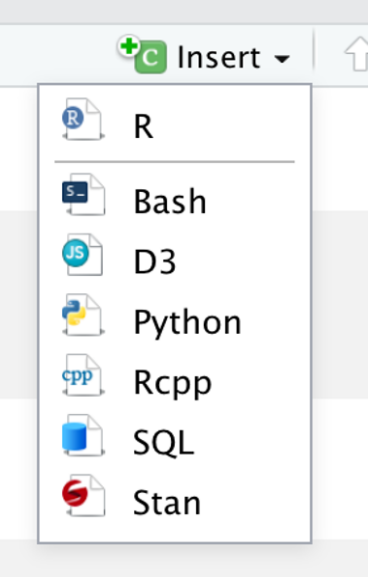
The anatomy of all `.Rmd` files (RNotebook or RMarkdown):

1. **Document Information**
2. **Formatted Text**
3. **Code Chunks**

Code Chunks

- This is what makes RMarkdown so cool!
- Type your code directly into a code chunk and work with it just like you would a `.R` script file
- When you're done, click `knit` at the top to generate your pretty report
    - All code chunks will be executed (unless you say otherwise...see next lecture)

Code Chunks

```
```{r}
# code goes here
answer <- 1 + 2
```
```

Each code chunk:

- **Starts and ends** with three backticks
  - if you don't have the ending 3, you're gonna have a bad time
- Has `{r}` at top next to the first 3 backticks
- Has gray background
- Looks like normal code
- Runs like normal code

Making Code Chunks

To make a code chunk:

- Use the `insert` button
- Manually type the backticks & `{r}`
- Keyboard shortcuts
  - PCs: `ctrl` + `alt` + `i`
  - Macs: `cmd` + `opt` + `i`



A finished product

```
# Step 1: Data Preparation
Let's first get our data ready for an analysis
```{r}
# load the packages
library(knitr)
library(psych)
library(ggplot2)

# set the working directory
setwd("~/Desktop/rSkillLab/")

# import data
midus <- read.csv("midus.csv")
```

Step 2: Analysis
Now let's get the mean of the age variable.
```{r}
mean_age <- mean(midus$age, na.rm = TRUE)
```
```

Step 1: Data Preparation

Let's first get our data ready for an analysis

```r
# load the packages
library(knitr)
library(psych)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```r
# set the working directory
setwd("~/Desktop/rSkillLab/")

# import data
midus <- read.csv("midus.csv")
```

Step 2: Analysis

Now let's get the mean of the age variable.

```r
mean_age <- mean(midus$age, na.rm = TRUE)
```

Code Chunks

All code will run in a code chunk

- It's very literal!
- We can't see `midus` or `mean_age`
  - These are stored as objects
  - If you want to see them, you need to tell R to show them to you
- We needed to import `midus`
  - If it was in your Environment *before* knitting the file AND you did *not* import your data, it would fail
  - Whatever data you use, you need to import it!

Viewing data.frames

To help make data.frames readable for humans, use the `kable()` function

- comes from the `knitr` package, although surprisingly, you don't need to manually load this one

1. Data Preparation

Let's first get our data ready for an analysis

```
# load the packages
library(knitr)
library(psych)
library(ggplot2)

# set working directory
setwd("~/Box Sync/Brazil 2019/")

# import data
midus <- read.csv("midus.csv")

# view the first 6 rows of the midus dataset
# head(midus) gets the first 6 rows
# kable() makes the output pretty
kable(x = head(midus), caption = "Midus Dataset")
```

Midus Dataset

ID	sex	age	BMI	physical_health_self	mental_health_self	self_esteem	life_satisfaction	hostility	heart_self	heart_father
10001	Male	61	26.263	2	4	42	7.750	5.5	No	No
10002	Male	69	24.077	5	5	34	8.250	6.0	No	Yes
10005	Female	80	NA	4	4	49	9.333	4.0	No	No
10006	Female	60	NA	3	3	NA	NA	NA	No	Yes
10010	Male	55	NA	4	3	28	8.250	8.0	No	Yes
10011	Female	52	25.991	5	4	41	7.000	5.5	No	No

RMarkdown Flexibility

Anatomy of .Rmd

1. Document Information
2. Formatted Text
3. Code Chunks

Today

All the different parameters you can add to your code chunks and document information (aka yaml header) for added flexibility.

Absolutely, 100% do **NOT** memorize *any* of this!

Parameter

A parameter is the thing that goes within the curly brackets `{ }` at the top of a code chunk. Some useful ones we will cover:

- Programming languge (necessary)
- Name of chunk (strongly encouraged)
- Echo
- Include
- Eval
- Message/Warning
- Figure parameters

What programming language?

- Default is R (obviously)
- But it can be different...

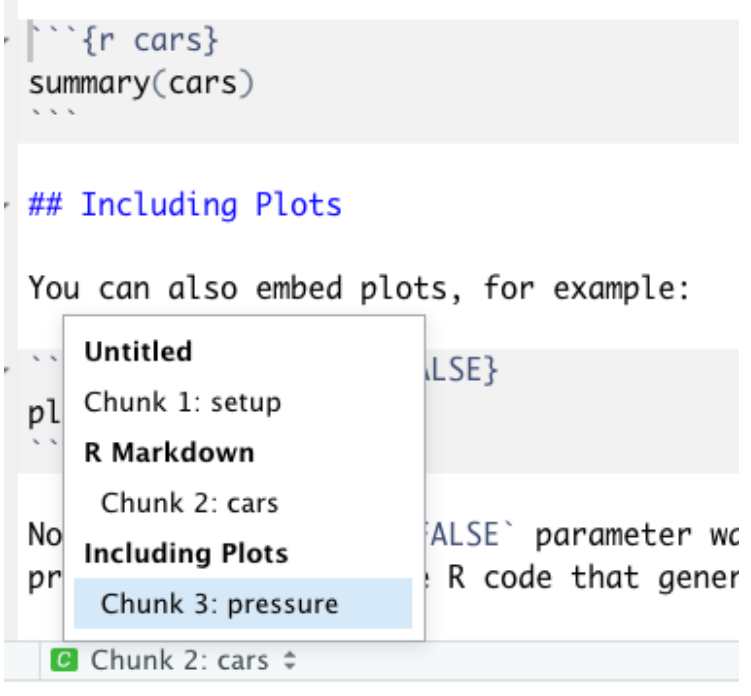
```{r}

```

```{python}

```

Name of chunk

- No comma (`,`) after the programming language parameter
- Name your chunk something that you will remember and makes sense!
  - Do **not** include spaces in the name
  - Do **not** include special characters like `#`, `$`, `%`, etc.
  - Dashes (`-`) and underscores (`_`) are OK
- Naming your chunk should help you navigate your document



The `echo` parameter

The `echo` parameter refers to if the code in the chunk should show up in your output document.

- If you want your code to appear, `echo = TRUE` (this is the default behavior)
- If you only the *OUTPUT* of the code to appear (not the code itself), set `echo = FALSE`

All the code will be run and executed no matter what!

echo = TRUE

```
# 1. Data Preparation
Let's first get our data ready for an analysis by having a setup code chunk. This
will include things like loading packages, setting the correct working directory,
reading in the data, and sometimes even just checking out the first couple of rows to
make sure everything worked.

```{r setup, echo=TRUE}
library(knitr)
library(psych)
library(ggplot2)

# set working directory
setwd("~/Box Sync/Brazil 2019/")

# import data
midus <- read.csv("midus.csv")

# view the first 6 rows of the midus dataset
# head(midus) gets the first 6 rows
# kable() makes the output pretty
kable(x = head(midus), caption = "Midus Dataset")
```
```

1. Data Preparation

Let's first get our data ready for an analysis by having a setup code chunk. This will include things like loading packages, setting the correct working directory, reading in the data, and sometimes even just checking out the first couple of rows to make sure everything worked.

```
library(knitr)
library(psych)
library(ggplot2)

# set working directory
setwd("~/Box Sync/Brazil 2019/")

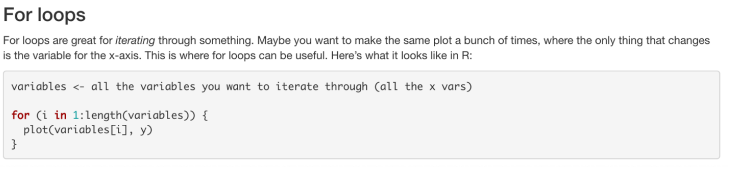
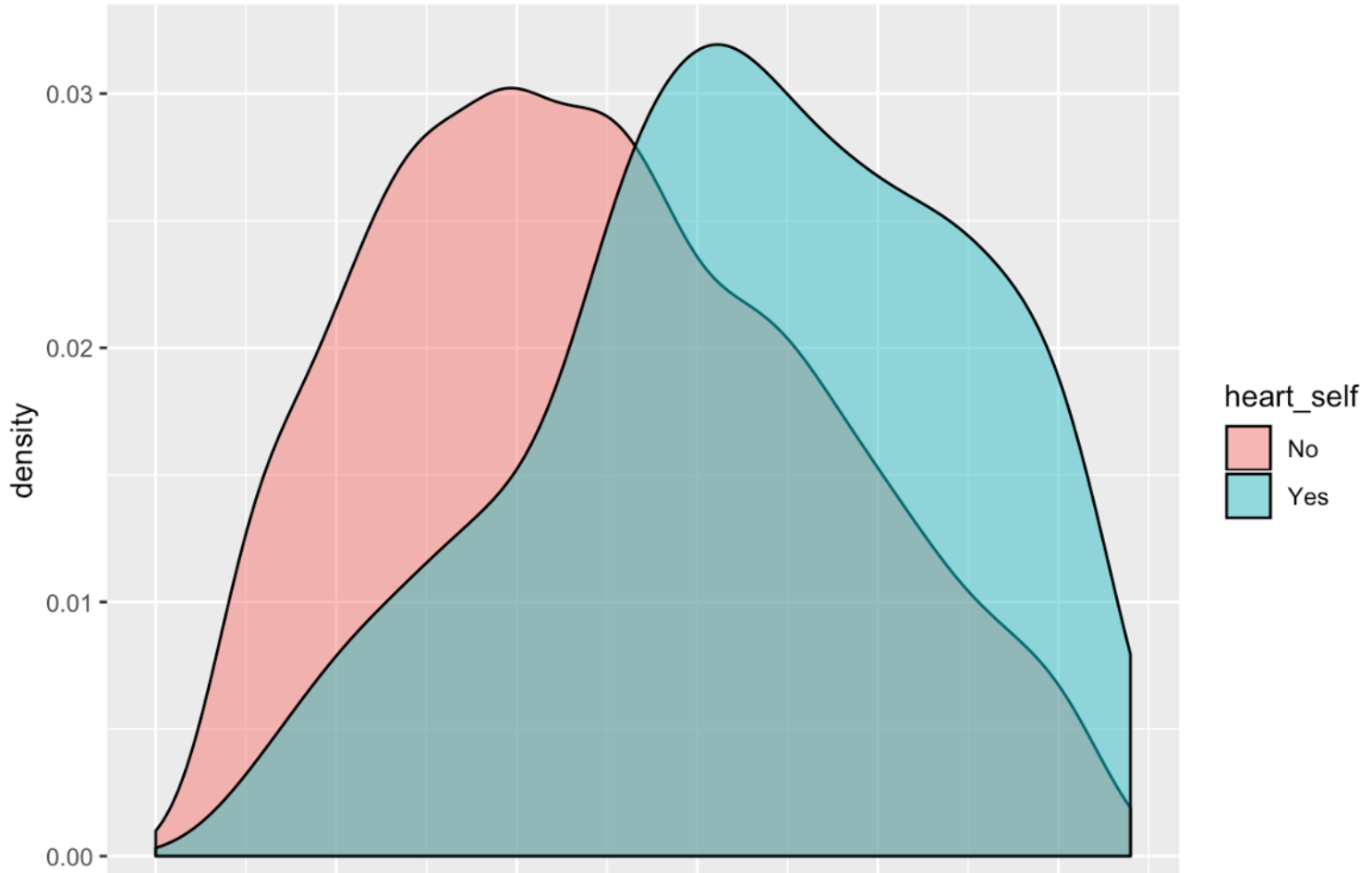
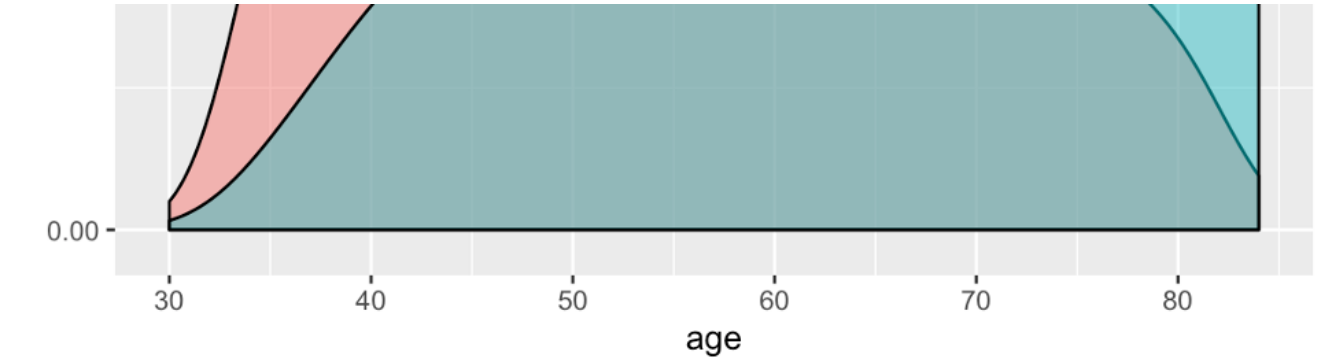
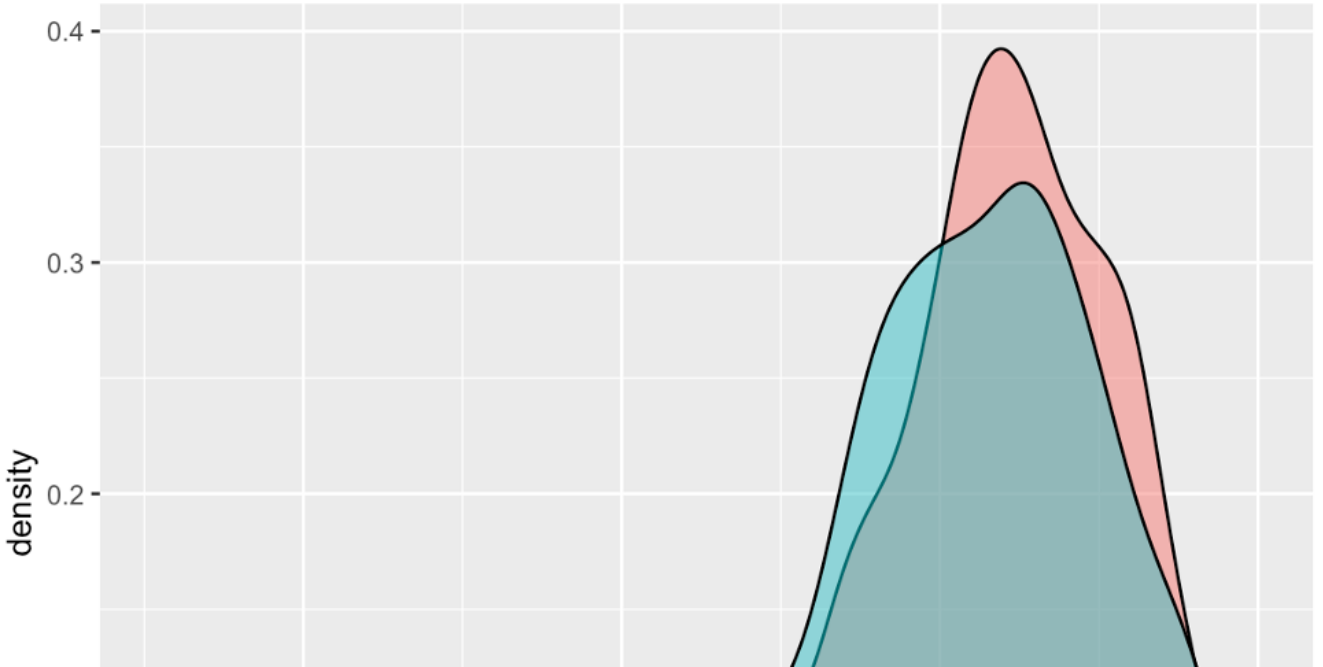
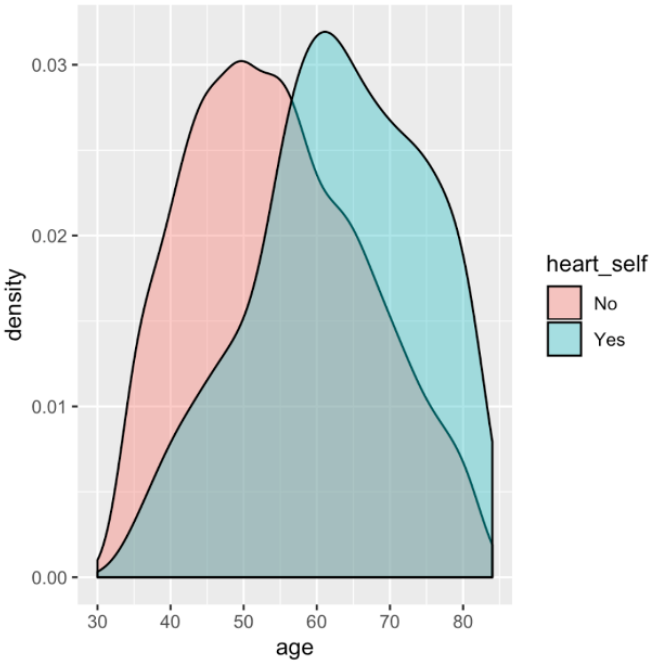
# import data
midus <- read.csv("midus.csv")

# view the first 6 rows of the midus dataset
# head(midus) gets the first 6 rows
# kable() makes the output pretty
kable(x = head(midus), caption = "Midus Dataset")
```

Midus Dataset

ID	sex	age	BMI	physical_health_self	mental_health_self	self_esteem	life_satisfaction	hostility	heart_self	heart_father
10001	Male	61	26.263	2	4	42	7.750	5.5	No	No
10002	Male	69	24.077	5	5	34	8.250	6.0	No	Yes
10005	Female	80	NA	4	4	49	9.333	4.0	No	No
10006	Female	60	NA	3	3	NA	NA	NA	No	Yes
10010	Male	55	NA	4	3	28	8.250	8.0	No	Yes
10011	Female	52	25.991	5	4	41	7.000	5.5	No	No

echo = FALSE

```
# 1. Data Preparation
Let's first get our data ready for an analysis by having a setup code chunk. This
will include things like loading packages, setting the correct working directory,
reading in the data, and sometimes even just checking out the first couple of rows to
make sure everything worked.

```{r setup, echo=FALSE}
library(knitr)
library(psych)
library(ggplot2)

# set working directory
setwd("~/Box Sync/Brazil 2019/")

# import data
midus <- read.csv("midus.csv")

# view the first 6 rows of the midus dataset
# head(midus) gets the first 6 rows
# kable() makes the output pretty
kable(x = head(midus), caption = "Midus Dataset")
```
```

1. Data Preparation

Let's first get our data ready for an analysis by having a setup code chunk. This will include things like loading packages, setting the correct working directory, reading in the data, and sometimes even just checking out the first couple of rows to make sure everything worked.

Midus Dataset

ID	sex	age	BMI	physical_health_self	mental_health_self	self_esteem	life_satisfaction	hostility	heart_self	heart_father
10001	Male	61	26.263	2	4	42	7.750	5.5	No	No
10002	Male	69	24.077	5	5	34	8.250	6.0	No	Yes
10005	Female	80	NA	4	4	49	9.333	4.0	No	No
10006	Female	60	NA	3	3	NA	NA	NA	No	Yes
10010	Male	55	NA	4	3	28	8.250	8.0	No	Yes
10011	Female	52	25.991	5	4	41	7.000	5.5	No	No

The `include` parameter

`include` is very similar to `echo` except *both* the code AND the output will not be shown.

- All code will be run and executed!

At the top of each .Rmd, it is helpful to have a code chunk, usually called `setup`. It's a great place to:

- import your dataset (`read.csv`)
- load the necessary packages (`library()`)
- set options that will apply to the entire document (later slides)
- etc...

For the most part, the people you send this file to do not need to see any of the above 👆. So it's a good place to use `include = FALSE`.

The `include` parameter

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

library(psych)
library(tidyverse)
library(wesanderson)

setwd("~/Desktop/")
anxiety <- read.csv("Fake_Fellowship_Data.csv")
anxiety
```

## Anxiety Dataset

Scientists think that a new drug might help reduce anxiety. You or your
colleagues run a clinical trial in order to see if it helps. You give 20
people the drug (aka "treatment" group), and another 20 people a sugar
pill (aka "placebo" group). You collect the following pieces of
information:
````

Anxiety Dataset

Scientists think that a new drug might help reduce anxiety. You or your colleagues run a clinical trial in order to see if it helps. You give 20 people the drug (aka “treatment” group), and another 20 people a sugar pill (aka “placebo” group). You collect the following pieces of information:

The `eval` parameter

- Should the code run at all?
- Default is TRUE
- This is sometimes nice when you're trying to teach a concept or explain how something works.

````
### For loops
For loops are great for *iterating* through something. Maybe you want to make the
same plot a bunch of times, where the only thing that changes is the variable for the
x-axis. This is where for loops can be useful. Here's what it looks like in R:

```{r teachingExample, eval=FALSE, echo=TRUE}

variables <- all the variables you want to iterate through (all the x vars)

for (i in 1:length(variables)) {
  plot(variables[i], y)
}

```
````



Messages & Warnings

- Sometimes functions will result in code that shows you *non-error* messages or warnings in the console. Examples:
  - When you load a package `ggplot::alpha is masked by psych::alpha`
  - `removed any NA`
- These are fine for personal use, but if you're sharing your codde, they don't need to see these things
- The default behavior of RMarkdown is to show you the messages and warnings
- To change, add `message = FALSE` and `warnring = FALSE` parameters

Note, it's hard to tell the difference between messages and warnings, so I usually do both

message = TRUE

```
```{r loadingPackages, message = TRUE}

#load packages
library(psych)
library(tidyverse)
library(knitr)
```
```

```
#load packages
library(psych)
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.2.1     ✓ purrr   0.3.3
## ✓ tibble  2.1.3     ✓ dplyr   0.8.4
## ✓ tidyr   1.0.2     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.4.0
```

```
## ── Conflicts ────────────────────────── tidyverse_conflicts() ──
## x ggplot2::%+%()   masks psych::%+%()
## x ggplot2::alpha() masks psych::alpha()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(knitr)
```

message = FALSE

```
```{r loadingPackages, message = FALSE}
#load packages
library(psych)
library(tidyverse)
library(knitr)
```
```

```
#load packages
library(psych)
library(tidyverse)
library(knitr)
```

Figures

How big do you want your figures to be on the page?

- can modify with `fig.width` & `fig.height`
- default is 7x7 inches, but you might want to make that smaller if you want to see 2 next to each other

How do you want your figures to be aligned on the page?

- right? left? lenter?
- can modify with `fig.align`
- default is to not make any adjustments at all

Problem with Figures

Aligning Plots



Problem with Figures





Solution to Figures

Solution to Figures

```
### Aligning Plots
```{r plots, fig.height=4.5, fig.width=4.5, echo = FALSE, fig.align='center'}
```

Aligning Plots


```

Parameters

- If you want to modify each individual code chunk, put the parameter within the curly brackets `{ }` (like we've been talking about)
- If you want to apply that same parameter to the *entire* document:
  - Have a setup chunk (this is good practice anyways)
  - Inside the setup chunk, you'll write something like this:

````
```{r setup, include=FALSE, warning=FALSE, message=FALSE}

knitr::opts_chunk$set(echo = TRUE)
```
````

You'll want to look it up for the exact parameter, but it usually starts with `knitr::opts_chunk`

Other things

- Table of contents
- Code folding
- Inline code
- Equations
- Running code in RMarkdown
- Variables & environment
- Output file types

Table of Contents

- The Table of Conents is based on Markdown headers (`#`)
- You specify if you want one at all in the YAML header
- It will show headers 1 (`#`) through 3 (`###`) by default
- If you want to modify this, you can do so with `toc_depth`

Table of Contents - default

```
---
title: "Table of Contents Examples"
author: "Shelly Cooper"
output:
  html_document:
    df_print: paged
    toc: yes
---

# Introduction/Background
blah blah blah

### Hypotheses
blah blah blah

# Methods

# Results

# Discussion

##### Acknowledgements
My students are amazing, and your other profs suck (jk, not really, they're
great!)
```

Table of Contents Examples

Shelly Cooper

- Introduction/Background
  - Hypotheses
- Methods
- Results
- Discussion

Introduction/Background

blah blah blah

Hypotheses

blah blah blah

Methods

Results

Discussion

Acknowledgements

My students are amazing, and your other profs suck (jk, not really, they're great!)

Table of Contents - `toc_depth: 5`

```
---
title: "Table of Contents Examples"
author: "Shelly Cooper"
output:
  html_document:
    df_print: paged
    toc: yes
    toc_depth: 5
---

# Introduction/Background
blah blah blah

### Hypotheses
blah blah blah

# Methods

# Results

# Discussion

##### Acknowledgements
My students are amazing, and your other profs suck (jk, not really, they're
great!)
```

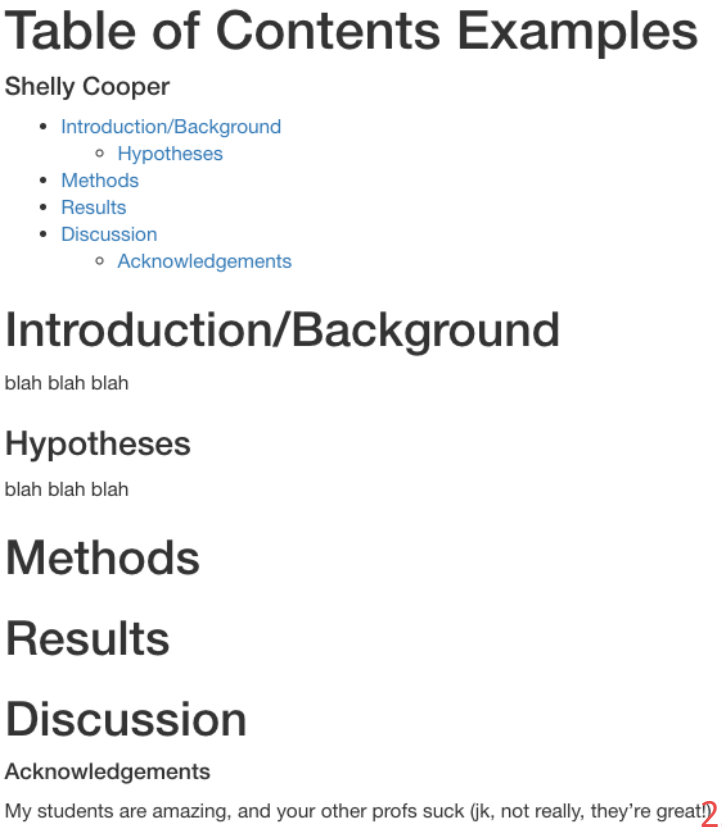


Table of Contents - HTML Only!

- If your output file is an HTML file, you can have the Table of Contents float down the side of your screen while you scroll!
- Use `toc_float` in the YAML header

```yaml
---
title: "Table of Contents Examples"
author: "Shelly Cooper"
output:
  html_document:
    df_print: paged
    toc: yes
    toc_depth: 5
    toc_float: yes
---

# Introduction/Background
blah blah blah

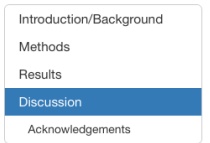
### Hypotheses
blah blah blah

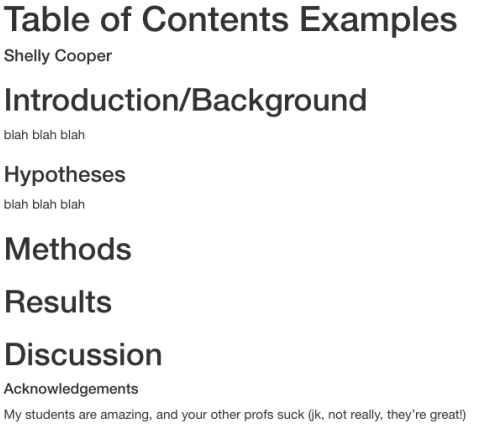
# Methods

# Results

# Discussion

##### Acknowledgements
My students are amazing, and your other profs suck (jk, not really, they're
great!)
```





Code Folding - HTML Only!

- Give the person you are sharing the document with the option of looking at your code
- Add `code_folding: "hide"` to the YAML header

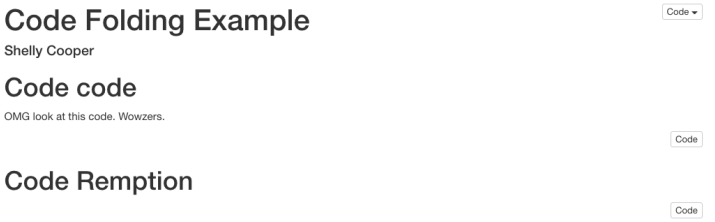
```
---
title: "Code Folding Example"
author: "Shelly Cooper"
output:
  html_document:
    df_print: paged
    code_folding: "hide"
---

# Code code
OMG look at this code. Wowzers.

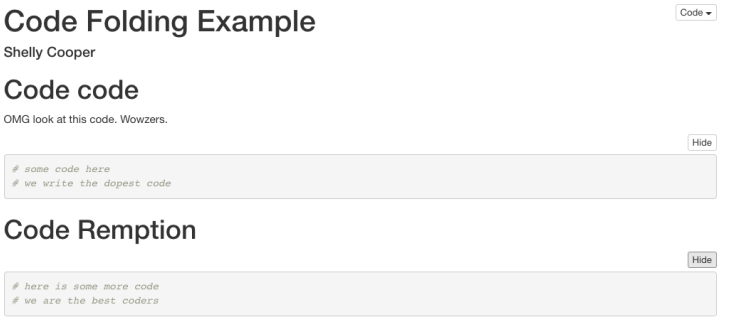
```{r}
# some code here
# we write the dopest code
```

Code Remption

```{r}
# here is some more code
# we are the best coders
```
```



Code Folding - HTML Only!



Inline Code

- Sometimes you want to acknowledge that something is code in formatted text. Ex: I am formatted text, but `this` is `code`.
- This is called "inline code"
- Wrap the text within single backtics

```
### Statistical Analyses
We used the `midus` data set for this analysis. The line of code we ran to
load in our data was `midus <- read.csv("midus.csv")`.
```

Statistical Analyses

We used the `midus` data set for this analysis. The line of code we ran to load in our data was `midus <- read.csv("midus.csv")`.

Inline Code

- You can also run some R code the same way
- This is helpful when:
  - You stored a value that you'd like to share
  - You want to show someting in a way your audience will like (e.g., rounding a number)
- This is NOT great for lots of code! Meant to be a quick thing.

```
# Methods

```{r}
iris <- iris

meanPetalLength <- mean(iris$Petal.Length)
meanPetalLength
```

Results
We saw that the average petal length was `r round(x = meanPetalLength,
digits = 1)`. We have no idea if this is reasonable or not because we don't
know anything about flowers.
```

Methods

```
iris <- iris

meanPetalLength <- mean(iris$Petal.Length)
meanPetalLength
```

```
## [1] 3.758
```

Results

We saw that the average petal length was 3.8. We have no idea if this is reasonable or not because we don't know anything about flowers.

Equations

- LaTeX works well in RMarkdown. Really great for equations/matrices
- For an equation, surround with $

```
- Equation of a line is $y = mx+b$
- Greek letters are fun $\alpha$, $\beta$, $\omega$,
- If you want your equation to be centered on the page, use double `$$`

$$ c^2 = a^2 + b^2$$
```

- Equation of a line is $y = mx + b$
- Greek letters are fun α, β, ω,
- If you want your equation to be centered on the page, use double $$

$$c^2 = a^2 + b^2$$

Running Code

- You can run code within a code chunk just like an .R script (go line by line or highlight lots of lines and hit `Run`)
- OR you can use the green arrow at the top right corner of the code chunk to run all lines within that chunk.

```
# Methods

```{r}
iris <- iris

meanPetalLength <- mean(iris$Petal.Length)
meanPetalLength
```

Results
We saw that the average petal length was `r round(x = meanPetalLength,
digits = 1)`. We have no idea if this is reasonable or not because we don't
know anything about flowers.
```

Running Code

- You can also use the other button to run all code chunks *above* the current chunk!
- This is wonderful when you think you're working with the wrong variable, or you screw up your data.frame and want to get it back to how it was

```
# Methods

```{r}
iris <- iris

meanPetalLength <- mean(iris$Petal.Length)
meanPetalLength
```

Results
We saw that the average petal length was `r round(x = meanPetalLength,
digits = 1)`. We have no idea if this is reasonable or not because we don't
know anything about flowers.
```

Reminder!

You **need** to make sure that the variables/datasets you want to work with are imported or created within the .Rmd file!

- When a .Rmd file knits, it starts as though there is NOTHING in your Environment and NONE of the packages are loaded...like a brand new R session
- If you import your data through the GUI, and *don't* put that code into the .Rmd file, when you go to `knit`, it won't find your file and it won't work properly
- The same is true if you make a variable in your Console, but forget to put that code into your .Rmd file

Other file types

- We've mainly been working with HTML files. Other options include Word and PDF (and some more advanced things like slides -- all of the slides in this class were made with RMarkdown!)
- Word Documents
    - This works...OK-ish. The only weird thing is that it will say you can't edit the file after it knits. You need to save it as a new file in order to edit it. But it will usually prompt you to do so.
    - Also, formatting tables in Word is really annoying.
- PDF Documents
    - If you have a Mac, this should work seamlessly
    - If you have a PC, you need to have LaTeX on your computer, which is usually installed with MiKTeX. See the Resources tab for how to do this -- it's very annoying.

Don't forget about R's Cheatsheets!

`Help > Cheatsheets` or Google `RMarkdown cheatsheet`

Also, there are cheatsheets for *a lot* of things, including `dplyr` and `ggplot2`. Again, never memorize anything!