Reproducible Reports with R Markdown

Jessica Minnier, PhD & Meike Niederhausen, PhD
OCTRI Biostatistics, Epidemiology, Research & Design (BERD) Workshop

2019/07/18 & 2019/09/25

slides: bit.ly/berd_rmd
pdf: bit.ly/berd_rmd_pdf

Load files for today's workshop

1. Open slides bit.ly/berd_rmd
2. Get project folder
    - Download zip folder at bit.ly/berd_rmd_zip
    - UNZIP completely (right click-> "extract all")
    - Open unzipped folder
    - Open (double click) `berd_rmarkdown_project.Rproj`
    - Inside RStudio 'Files' tab: click on file `00-install.R` and click "Run" to run all lines of code.

Allison Horst

Learning objectives

- Understand how to use literate programming for reproducible research
- Basics of Markdown language
- Learn how to create R Markdown files with code and markdown text
- Turn R Markdown files into html, pdf, Word, or presentation files
- Learn about reproducible project workflows
- (If time allows) Learn some additional R Markdown tips

Why Reproducibility?

- Evidence your results are correct.
- Allow others to use our methods and results.

> "An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result."

-- (Claerbout and Karrenbach 1992)

> Your closest collaborator is you six months ago, but you don't reply to emails.

-- @gonuke, quoting @mtholder

Types of Reproducibility

- **Computational reproducibility:** detailed information is provided about
  - *code, software, hardware and implementation details.*
- **Empirical reproducibility:** detailed information is provided about
  - non-computational empirical *scientific experiments and observations [data].*
- **Statistical reproducibility:** detailed information is provided about
  - the choice of *statistical tests, model parameters, threshold values, etc.*

R Opensci Reproducibility Guide

Software tool for reproducibility: *Literate Programming*

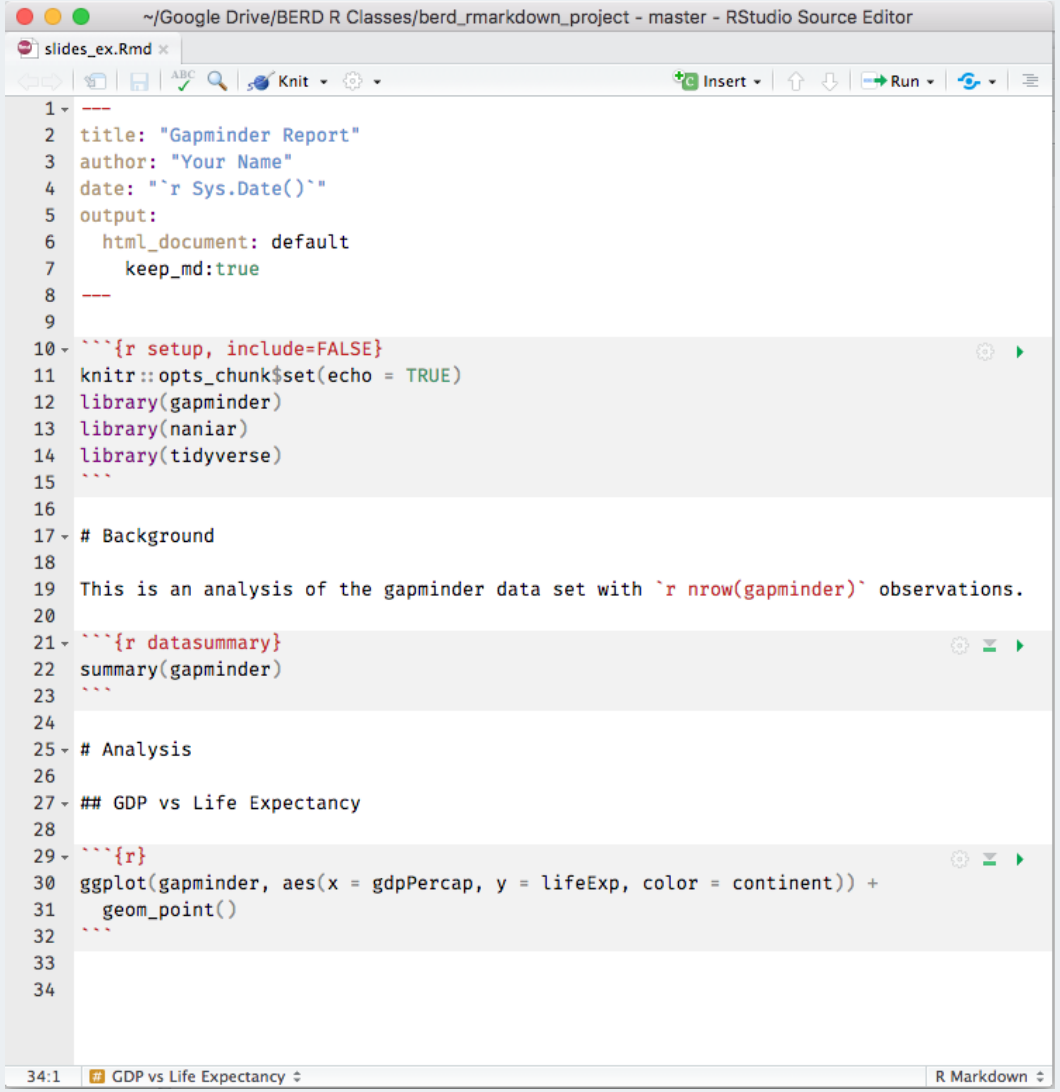
> "These tools enable writing and publishing **self-contained documents that include narrative and code used to generate both text and graphical results**.
>
> In the R ecosystem, knitr [R markdown] and its ancestor Sweave used with RStudio are the main tools for literate computing. Markdown or LaTeX are used for writing the narrative, with chunks of R code sprinkled throughout the narrative. IPython is a popular related system for the Python language, providing an interactive notebook for browser-based literate computing."

R Opensci Reproducibility Guide

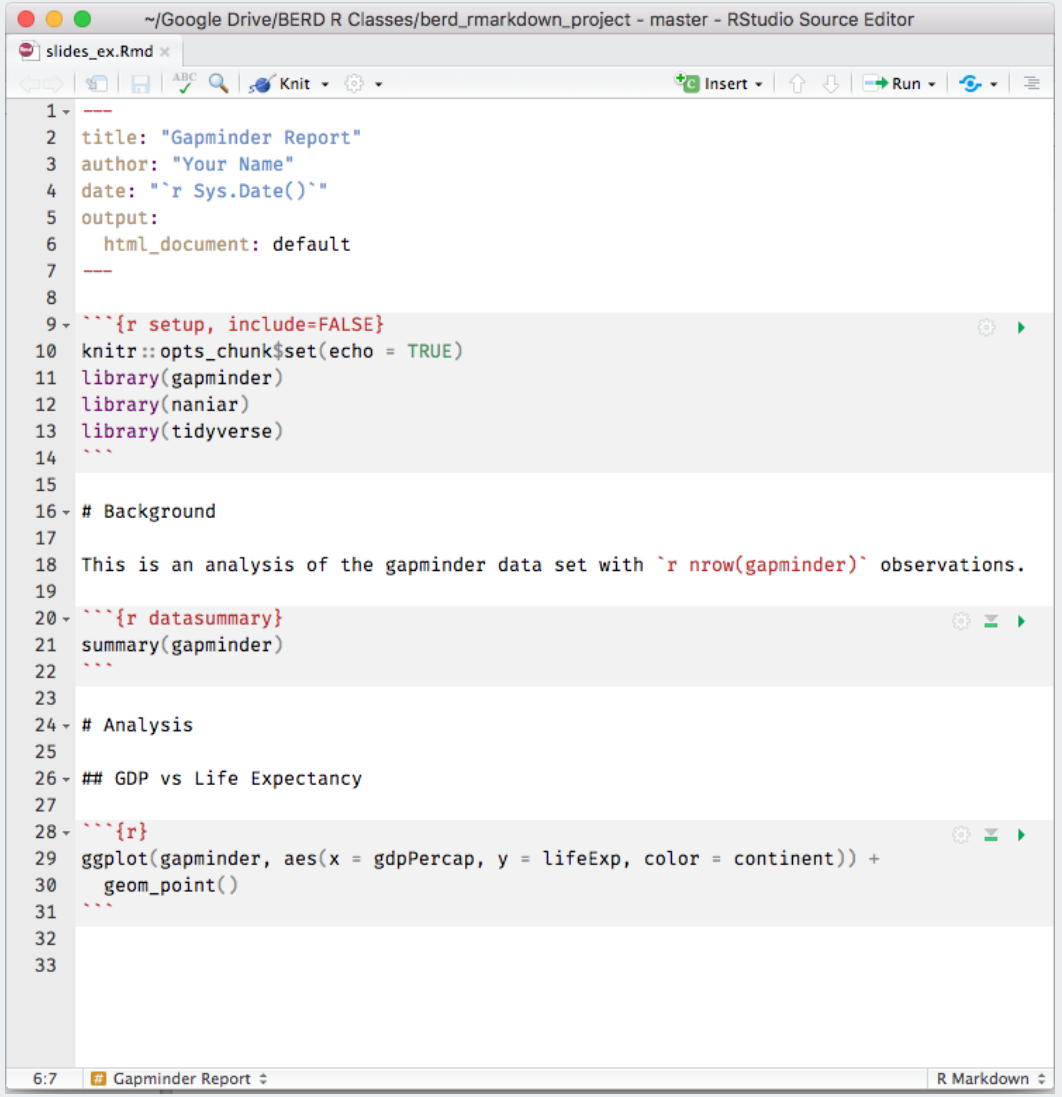
R Markdown = .Rmd file = Code + text

`knitr` is a package that converts `.Rmd` files containing code + markdown syntax to a plain text `.md` markdown file, and then to other formats (html, pdf, Word, etc)

knitr converts .Rmd -> .md (behind the scenes)



knitr converts .Rmd -> .md -> .html



knitr converts .Rmd -> .md -> .pdf

~/Google Drive/BERD R Classes/berd_rmarkdown_project - master - RStudio Source Editor

slides_ex.Rmd

```
---
title: "Gapminder Report"
author: "Your Name"
date: "`r Sys.Date()`"
output:
  pdf_document: default
---

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

Background

This is an analysis of the gapminder data set with `r nrow(gapminder)` observations.

```{r datasummary}
summary(gapminder)
```

Analysis

GDP vs Life Expectancy

```{r}
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point()
```
```

6:24 Gapminder Report R Markdown

Gapminder Report

Your Name

2019-07-17

Background

Summary

This is an analysis of the gapminder data set with 1704 observations.

```
summary(gapminder)
```

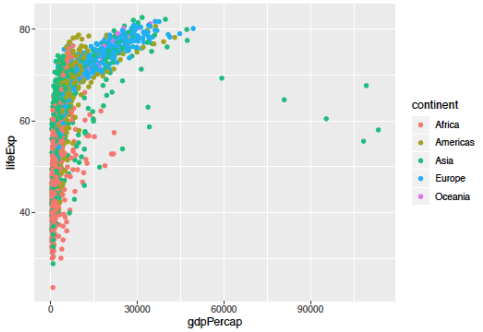
```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04   Min.   :   241.2
##  1st Qu.:2.794e+06   1st Qu.:  1202.1
##  Median :7.024e+06   Median :  3531.8
##  Mean   :2.960e+07   Mean   :  7215.3
##  3rd Qu.:1.959e+07   3rd Qu.:  9325.5
##  Max.   :1.319e+09   Max.   :113523.1
##
```

Analysis

GDP vs Life Expectancy

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point()
```

1



2

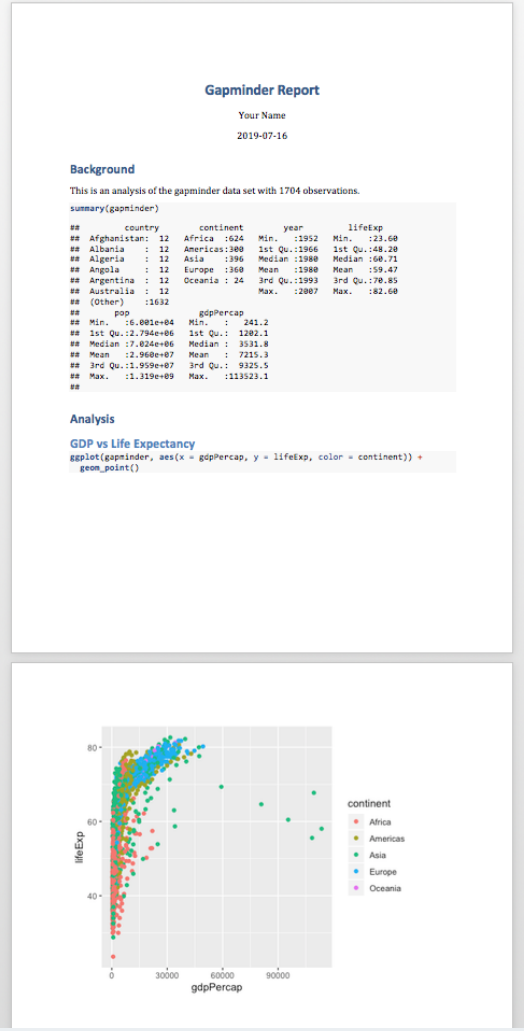
knitr converts .Rmd -> .md -> .doc

~/Google Drive/BERD R Classes/berd_rmarkdown_project - master - RStudio Source Editor

slides_ex.Rmd

```
1  ---
2  title: "Gapminder Report"
3  author: "Your Name"
4  date: "`r Sys.Date()`"
5  output:
6    word_document: default
7  ---
8
9  ```{r setup, include=FALSE}
10 knitr::opts_chunk$set(echo = TRUE)
11 library(gapminder)
12 library(naniar)
13 library(tidyverse)
14 ```
15
16 # Background
17
18 This is an analysis of the gapminder data set with `r nrow(gapminder)` observations.
19
20 ```{r datasummary}
21 summary(gapminder)
22 ```
23
24 # Analysis
25
26 ## GDP vs Life Expectancy
27
28 ```{r}
29 ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
30   geom_point()
31 ```
32
33
```

6:25 Gapminder Report R Markdown



Gapminder Report

Your Name

2019-07-16

Background

This is an analysis of the gapminder data set with 1704 observations.

```
summary(gapminder)

##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04   Min.   :   241.2
##  1st Qu.:2.794e+06   1st Qu.:  1202.1
##  Median :7.024e+06   Median :  3531.8
##  Mean   :2.960e+07   Mean   :  7215.3
##  3rd Qu.:1.959e+07   3rd Qu.:  9325.5
##  Max.   :1.319e+09   Max.   :113523.1
##
```

Analysis

GDP vs Life Expectancy

```
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point()
```

knitr converts .Rmd -> .md -> slides

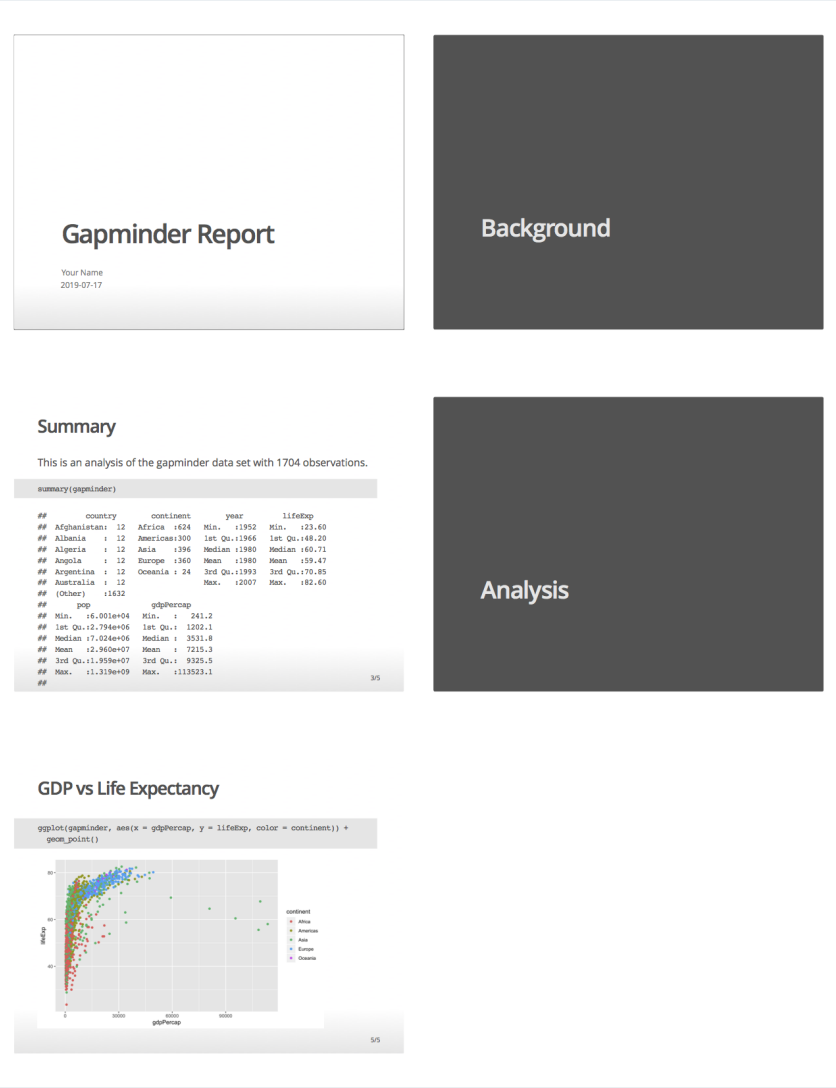
~/Google Drive/BERD R Classes/berd_rmarkdown_project - master - RStudio Source Editor

slides_ex.Rmd

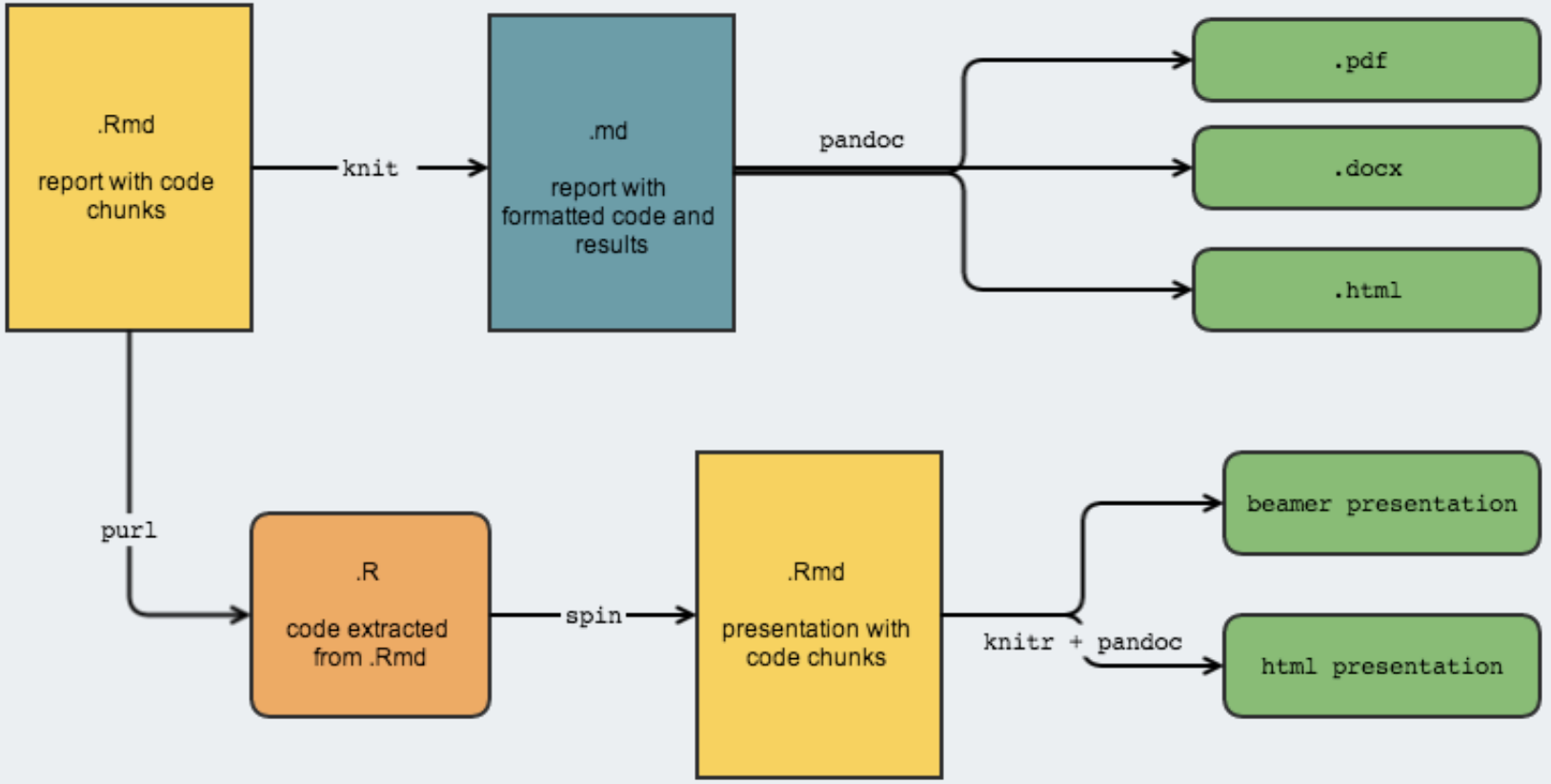
Knit | Insert | Run

```
1  ---
2  title: "Gapminder Report"
3  author: "Your Name"
4  date: "`r Sys.Date()`"
5  output:
6    ioslides_presentation
7  ---
8
9  ```{r setup, include=FALSE}
10 knitr::opts_chunk$set(echo = TRUE)
11 library(gapminder)
12 library(naniar)
13 library(tidyverse)
14 ```
15
16 # Background
17
18 This is an analysis of the gapminder data set with `r nrow(gapminder)` observations.
19
20 ```{r datasummary}
21 summary(gapminder)
22 ```
23
24 # Analysis
25
26 ## GDP vs Life Expectancy
27
28 ```{r}
29 ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
30   geom_point()
31 ```
32
33
```

6:24 | Gapminder Report | R Markdown



R Markdown vs. knitr::knit()



Michael Sachs

Good practices in RStudio

Use projects (read this)

- Create an RStudio project for each data analysis project
- A project is associated with a directory folder
  - Sets *working directory*
  - Keep data files there
  - Keep scripts there; edit them, run them in bits or as a whole
  - Save your outputs (plots and cleaned data) there
- Only use relative paths, never absolute paths
  - relative (good): `read_csv("data/mydata.csv")`
  - absolute (bad): `read_csv("/home/yourname/Documents/stuff/mydata.csv")`

Advantages of using projects

- standardize file paths
- keep everything together
- a whole folder can be shared and run on another computer

Basic R Markdown example

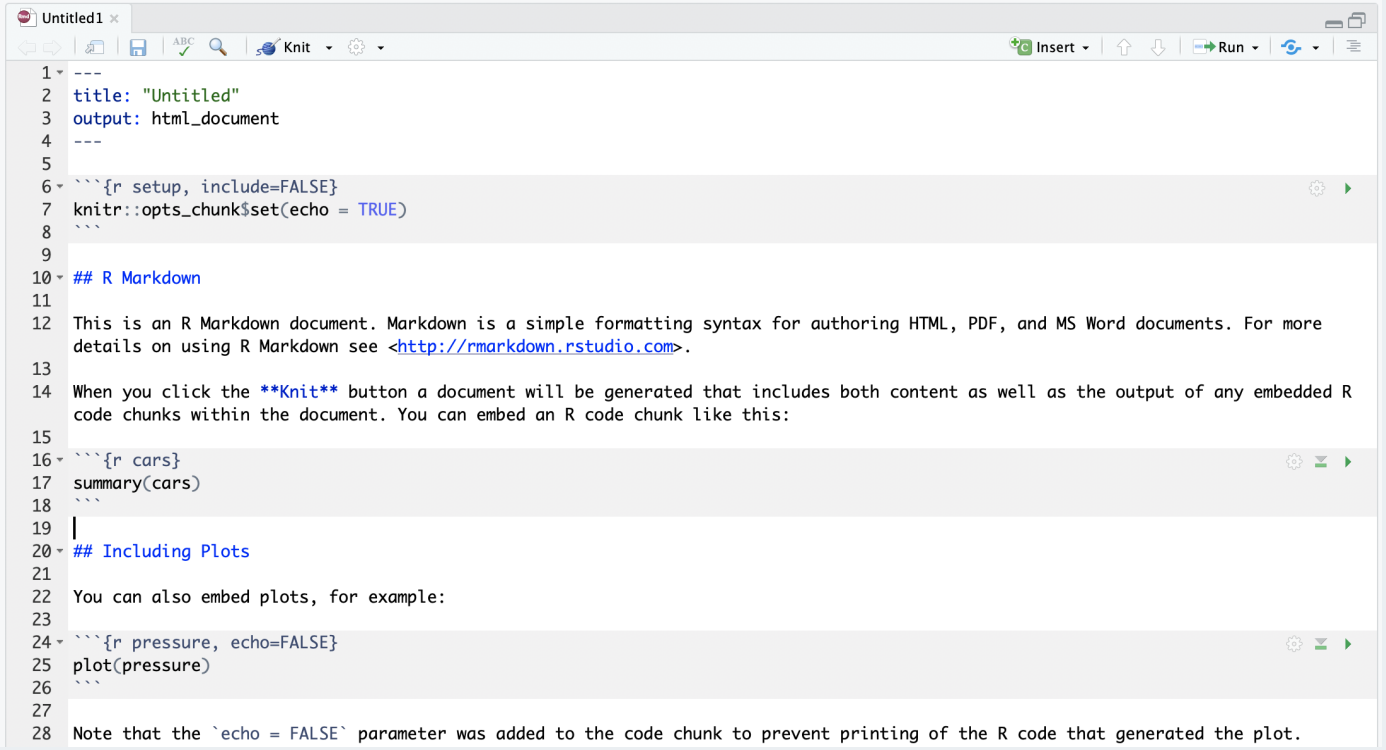
https://www.rstudio.com/products/rpackages/

Create an R Markdown file (.Rmd)

Two options:

1. click on File → New File → R Markdown... , or
2. in upper left corner of RStudio click on → R Markdown...

You should see the following text in your editor window:



Knit the .Rmd file

*Before knitting the .Rmd file, you must first **save it**.*

To **knit** the .Rmd file, either

1. click on the knit icon  at the top of the editor window
2. or use keyboard shortcuts
    - Mac: *Command+Shift+K*
    - PC: *Ctrl+Shift+K*
3. or use the `render()` command in Console - *See Extensions section for details*

A new window will open with the html output.

Remark:

- The template .Rmd file that RStudio creates will knit to an html file by default
- Later we will go over knitting to other file types

Compare the .Rmd file with its html output

.Rmd file

default_html.Rmd

Knit | Insert | Run

```
---
title: "Untitled"
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)
```

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

html output

~/Google Drive/BERD R Classes/berd_rmarkdown_project/default/default_html.html

default_html.html | Open in Browser | Find | Publish

Untitled

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.

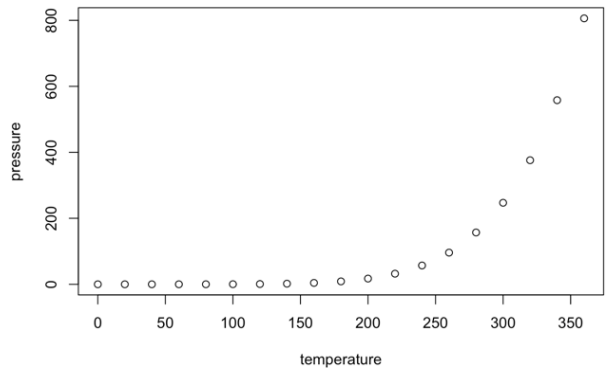
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



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

Compare the .Rmd file with its html output

.Rmd file

default_html.Rmd

Knit | Insert | Run

```
---
title: "Untitled"
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)
```

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

YAML metadata

Text

Code chunk

html output

~/Google Drive/BERD R Classes/berd_rmarkdown_project/default/default_html.html

default_html.html | Open in Browser | Find | Publish

Untitled

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:

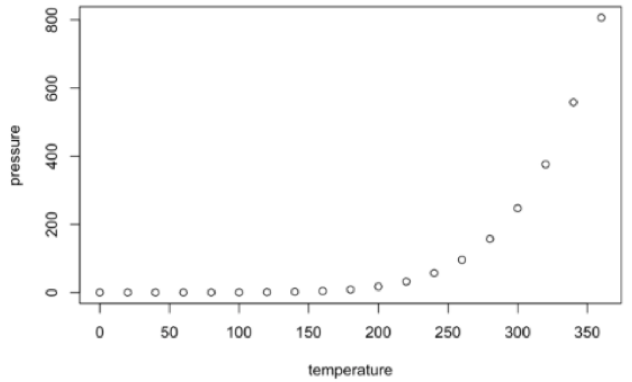
```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Code output

Including Plots

You can also embed plots, for example:



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

3 types of R Markdown content

1. *Text*
2. Code chunks
3. YAML metadata

Formatting text

- Markdown is a markup language similar to html or LaTeX
- All text formatting is specified via code

Text in editor:

```
Time to learn how to format text using R Markdown!

If I put two spaces
at the end of a line it will force a line break and start a new line.

*This text is in italics*, but _so is this text_.

**Bold** also has __2 options__

~~Should this be deleted?~~

`Sometimes text needs to be verbatim`

>or even a block quote.

Need^superscripts^ or~subscripts~?
```

Output:

Time to learn how to format text using R Markdown!

If I put two spaces
at the end of a line it will force a line break and start a new line.

This text is in italics, but *so is this text*.

Bold also has **2 options**

~~Should this be deleted?~~

`Sometimes text needs to be verbatim`

> or even a block quote.

Need$^{\text{superscripts}}$ or$_{\text{subscripts}}$?

Headers

- Organize your documents using headers to create sections and subsections
- Later in the workshop we will cover
  - automatically numbering headers in output file for easy reference
  - easily creating a TOC based on the header names

Text in editor:

```
# Header 1

## Header 2

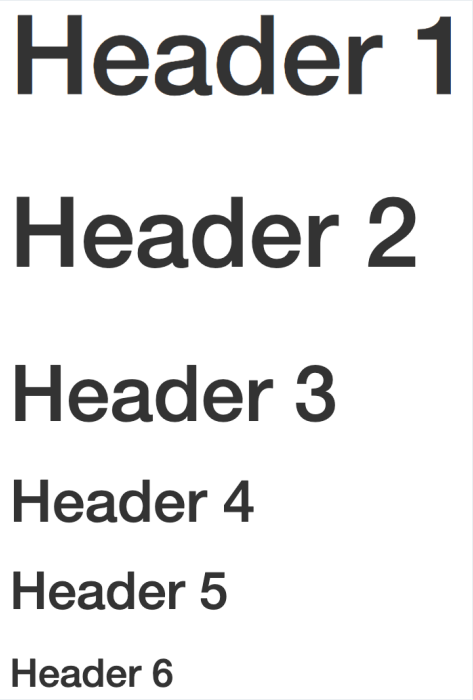
### Header 3

#### Header 4

##### Header 5

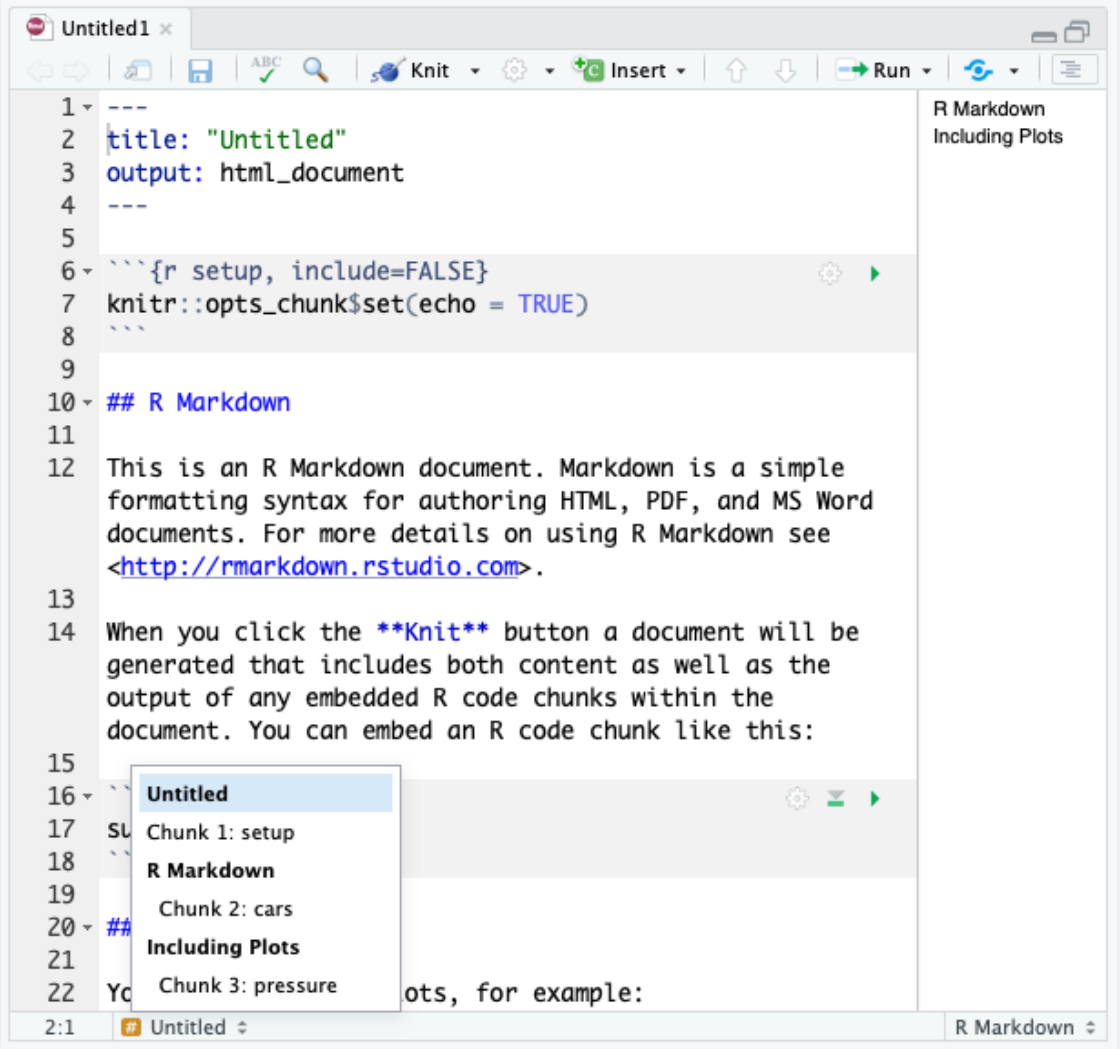
###### Header 6
```

Output:



RStudio tip

You can easily navigate through your .Rmd file if you use headers to outline your text



Unnumbered lists

Text in editor:

```
* This is an **unnumbered list**
  + with *sub-items*
      - and *sub-sub-items*,
          - or even deeper.
* You can use characters *, +, and - to create lists.
  * The order of the
      * characters is not important
          + and characters can be repeated.

What *is* important is the *spacing*!
+ indent each
  * sub-level with a tab and make sure
  * there is a space between the character starting
the list and the first bit of text,
  *otherwise the text won't be a new bullet in the
list
```

- This is an **unnumbered list**
  - with *sub-items*
    - and *sub-sub-items*,
      - or even deeper.
- You can use characters *, +, and - to create lists.
  - The order of the
    - characters is not important
      - and characters can be repeated.

What *is* important is the *spacing*!

- indent each
  - sub-level with a tab and make sure
  - there is a space between the character starting the list and the first bit of text, *otherwise the text won't be a new bullet in the list

Numbered lists

Text in editor:

```
1. This is a **Numbered list**,
1. which can have
  i. sub-items
      A. and sub-sub-items
1. Each bullet
  i. can start with `1.` or `i.`,
  i. in theory.
(@) This doesn't always work

when lists get interrupted though.

(@) Using `(@)` instead keeps
  * the numbering continuous.
    + Note that you can also
(@) nest unnumbered lists
  * within numbered lists
```

Output:

1. This is a **Numbered list**,
2. which can have

i. sub-items
   A. and sub-sub-items

1. Each bullet

i. can start with `1.` or `i.`,
ii. in theory.

1. This doesn't always work

when lists get interrupted though.

2. Using `(@)` instead keeps

- the numbering continuous.
    - Note that you can also

3. nest unnumbered lists

- within numbered lists

Math, horizontal rule, and hyperlinks

Text in editor:

```
* __Mathematical formulas and sybmols__ can be included
using LaTeX, both as *inline equations* or *formulas*:
  + Use single `$` for inline equations: $y=\beta_0 +
\beta_1x + \varepsilon$
  + Use double `$$` for centered formulas:

$$\hat{y}= \frac{3}{7} +  5 \mathrm{age} + 3^2 \cdot
\mathrm{height}$$
```

$$\hat{y} = \frac{3}{7} + 5\mathrm{age} + 3^2 \cdot \mathrm{height}$$

```
* __Horizontal rule__

***
* **Hyperlinks**
  + Learn more about LaTeX at this
[link](http://www.highpoint.edu/physics/files/2014/08/s
hort-math-guide.pdf).
]
```

Output:

- **Mathematical formulas and symbols** can be included using LaTeX, both as *inline equations* or *formulas*:
  - Use single $ for inline equations: $y = \beta_0 + \beta_1 x + \varepsilon$
  - Use double $$ for centered formulas:

$$\hat{y} = \frac{3}{7} + 5\mathrm{age} + 3^2 \cdot \mathrm{height}$$

- **Horizontal rule**

- **Hyperlinks**
  - Learn more about LaTeX at this link.

Insert images

Text in editor:

```
Gauss and the normal distribution were
featured on the 10 Deutsch Mark (DM) bill.
![alternate text: 10 DM bill](DM_10_Gauss.jpeg)
```


```
<!-- The alternate text only appears if the image
fails to load. -->
<!-- By the way, this is how you write comments
in markdown!! -->

You can also source an image on the internet
instead:
![10 DM bill](https://history.info/wp-content/uploads/2015/06/DEU-10m-anv.jpg)
```

Output:
Gauss and the normal distribution were featured on the 10 Deutsch Mark (DM) bill.

You can also source an image on the internet instead:

Tables created manually

Later we will use R code to create tables from data.

We can create tables using Markdown as well:

Text in editor:

```
Variable | n   | Mean $\pm$ SE
---------|-----|--------------
Age      | 198 | 42.3 $\pm$ 3.1 years
Height   | 194 | 68.1 $\pm$ 2.6 in
```

Output:

Variable	n	Mean ± SE
Age	198	42.3 ± 3.1 years
Height	194	68.1 ± 2.6 in

- We **do not recommend** creating tables where the numbers are hard-coded
  - since they are **not reproducible**!

Spell check

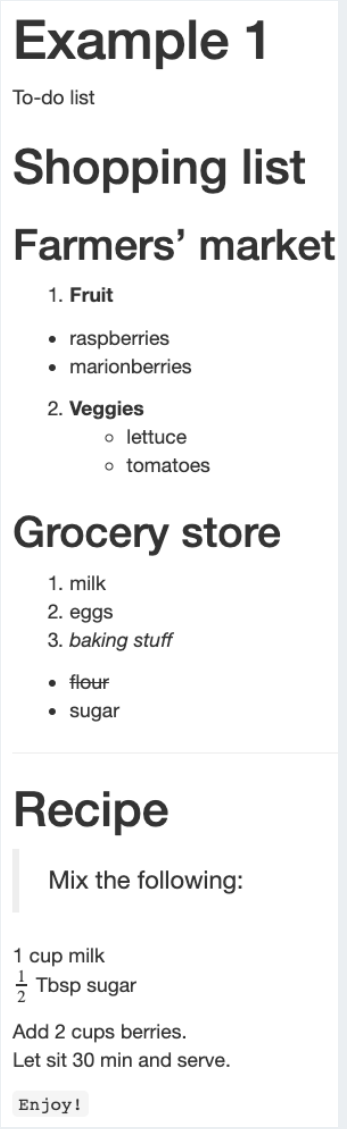
Alas, there are no autmatik sepll chekc to katch you're tipos and grammR.

- You can manually do a spell check by clicking on the  icon above the editor window.
- There is no built-in grammar check in RStudio.
  - The `gramr` package is an available RStudio Addin.

Practice!

Create an .Rmd file with file name `example1.Rmd` that creates the html output to the right.

- Hint: The first line is not a header.



Example 1

To-do list

Shopping list

Farmers' market

1. **Fruit**
   - raspberries
   - marionberries
2. **Veggies**
   - lettuce
   - tomatoes

Grocery store

1. milk
2. eggs
3. *baking stuff*

- ~~flour~~
- sugar

Recipe

> Mix the following:

1 cup milk
$\frac{1}{2}$ Tbsp sugar

Add 2 cups berries.
Let sit 30 min and serve.

```
Enjoy!
```

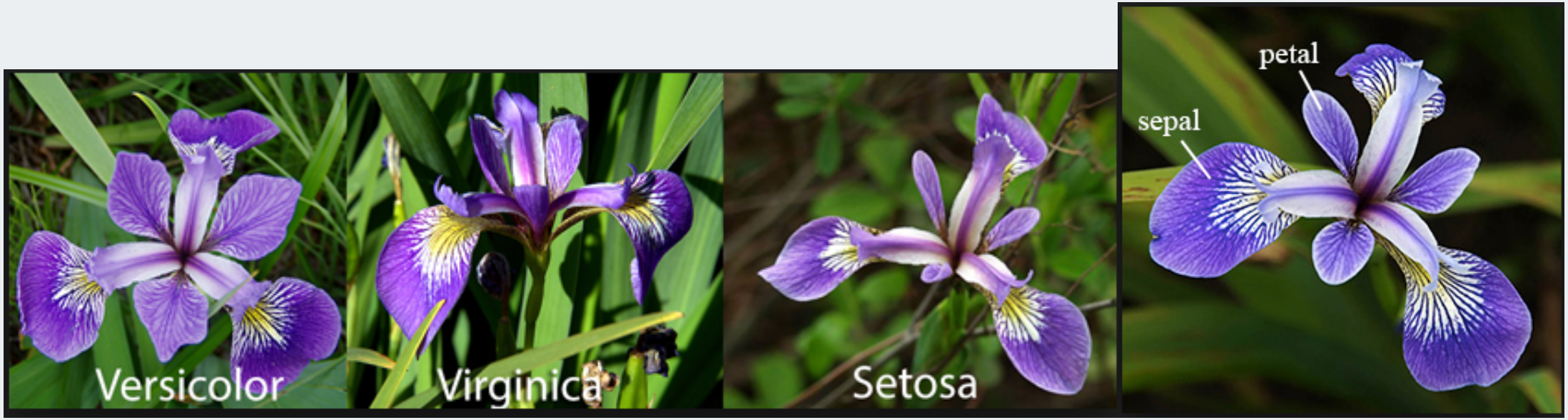
3 types of R Markdown content

1. Text
2. *Code chunks*
3. YAML metadata

Data description: Fisher's (or Anderson's) Iris data set

- n = 150
- 3 species of Iris flowers (Setosa, Virginica, and Versicolour)
  - 50 measurements of each type of Iris
- variables:
  - sepal length, sepal width, petal length, petal width, and species

Can the flower species be determined by these variables?



Gareth Duffy

Code chunks

Chunks of R code start with ```` ```{r} ```` and end with ```` ``` ````.

For example, the chunk

````
```{r}
summary(iris)
```
````

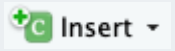
produces the output

```
summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

Create a code chunk

Code chunks can be created by either

1. Clicking on  →  at top right of editor window, or
2. **Keyboard shortcut**
    - Mac: *Command + Option + I*
    - PC: *Ctrl + Alt + I*

Chunk options- most common

Text in editor:

````
No options specified: see both code and output
```{r}
mean(iris$Sepal.Length)
```

__`echo`__ determines whether the R code is __displayed__ or not. The default is `TRUE`. When set to `FALSE`, the code is not displayed in the output:
```{r echo=FALSE}
mean(iris$Sepal.Length)
```

__`eval`__ determines whether the R code is __run__ or not. The default is `TRUE`. When set to `FALSE`, the code is not run but is displayed in the output:
```{r eval=FALSE}
mean(iris$Sepal.Length)
```
````

No options specified: see both code and output

```
mean(iris$Sepal.Length)
```

```
[1] 5.843333
```

echo determines whether the R code is **displayed** or not. The default is TRUE. When set to FALSE, the code is not displayed in the output:

```
[1] 5.843333
```

eval determines whether the R code is **run** or not. The default is TRUE. When set to FALSE, the code is not run but is displayed in the output:

```
mean(iris$Sepal.Length)
```

More chunk options

Text in editor:

````
__`include`__ determines whether to include
the R chunk in the output or not. The default
is `TRUE`. Below the chunk is run, but we do
not see the code or its output:
```{r include=FALSE}
mean(iris$Sepal.Length)
```
````

Output:

`include` determines whether to include the R chunk in the output or not. The default is `TRUE`. When set to `FALSE`, the chunk is run but we do not see the code or its output (note that nothing is displayed below):

- Setting **`include=FALSE`** is useful when you have R code that you want to run, but do not want to display either the code or its output.
- See the R Markdown cheatsheet for more chunk options.

Inline code

- You can also report R code output inline with the text
  - *R code is not shown in this case*

Text in editor:

````
```{r include=FALSE}
mean_SepalLength <- mean(iris$Sepal.Length)
```

The mean sepal length for all 3 species combined is
`r round(mean_SepalLength,1)`
(SD = `r round(sd(iris$Sepal.Length),1)`) cm.
````

Output:

The mean sepal length for all 3 species combined is 5.8 (SD = 0.8) cm.

- The code above is an example of where `include=FALSE` is used a chunk option to evaluate the code but not show the code or its output.
  - It saves the mean as `mean_SepalLength`, which can then be used later on.
- For the standard deviation, the inline code did the calculation.
- Thus it was not necessary to first save the mean as a variable.

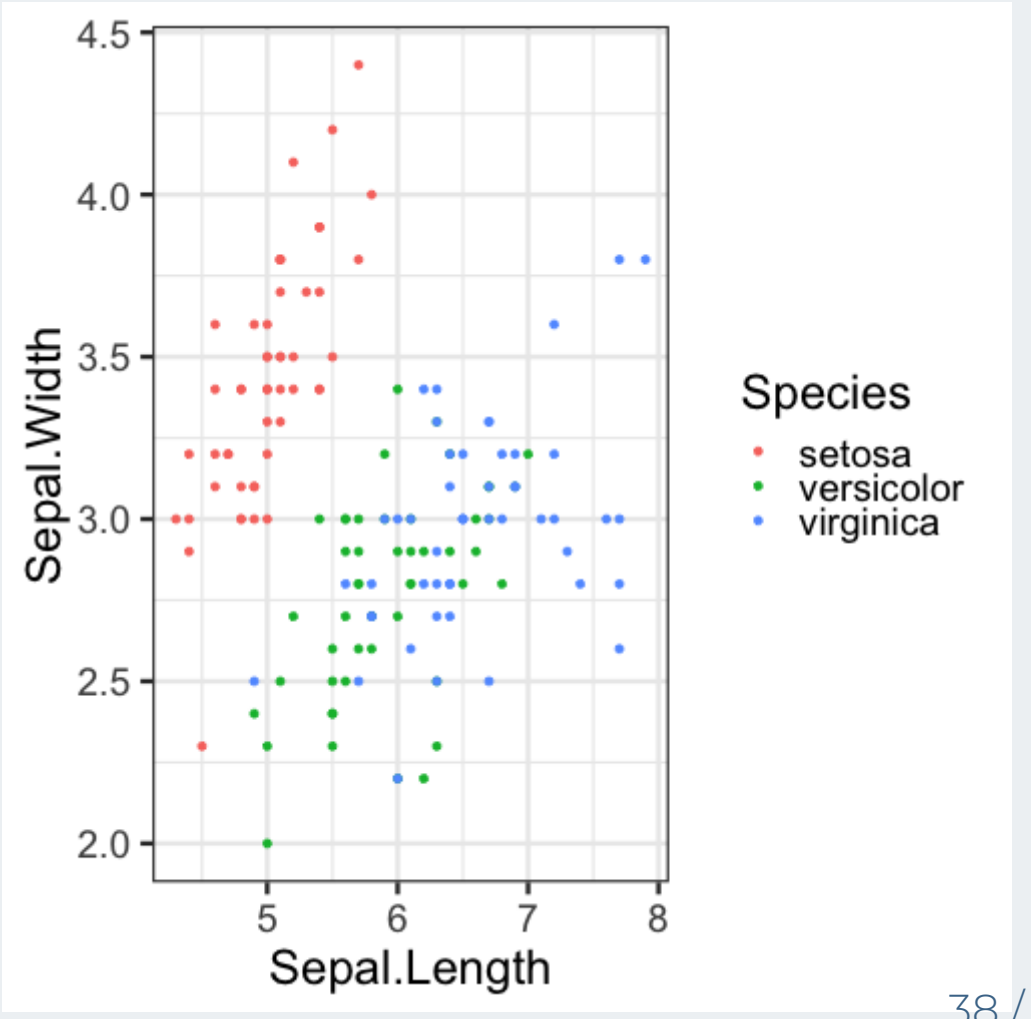
Figures

Text in editor:

````
```{r Sepal_WidthVsHeight, echo=FALSE, fig.width=7, fig.height=7}
library(ggplot2) # loads ggplot2 package
# Don't need to load ggplot2 if already loaded tidyverse package
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width,
                 color = Species)) +
  geom_point()
```
````

- Figure dimensions specified with `fig.width` and `fig.height`
- Figure name specified by the chunk label
  - The figure created by the chunk above is called `Sepal_WidthVsHeight-1.png`
  - *Chunk names must be unique!*
- `echo=FALSE` was used to hide the code and only display the figure

Output:



Tables - with no formatting

- Below we create a summary table with the mean and SD of sepal lengths
- The table is displayed with no special formatting

```
table_sepal_length <- iris %>%
  group_by(Species) %>%
  summarize(mean = mean(Sepal.Length),
            SD = sd(Sepal.Length))

table_sepal_length
```

```
# A tibble: 3 x 3
  Species     mean    SD
  <fct>      <dbl> <dbl>
1 setosa      5.01 0.352
2 versicolor  5.94 0.516
3 virginica   6.59 0.636
```

Tables - with kable

- The `kable` command from the `knitr` package has some basic formatting options
  - **html** tables: harder to read due to squished spacing; can include caption
  - **markdown** tables: nicer formatting; width = page width

Text in editor:

````
```{r echo=FALSE}
library(knitr)
# Only need to load package once in a document.
# Recommend doing this in 1st chunk of the .Rmd

kable(table_sepal_length,
      format = "html", digits = 2,
      caption = "Iris Sepal Lengths")

kable(table_sepal_length,
      format = "markdown", digits = 2,
      caption = "Iris Sepal Lengths")
# Note that the caption isn't shown!!
```
````

Output:

Iris Sepal Lengths

Species	mean	SD
setosa	5.01	0.35
versicolor	5.94	0.52
virginica	6.59	0.64

Species	mean	SD
setosa	5.01	0.35
versicolor	5.94	0.52
virginica	6.59	0.64

Tables - use kableExtra for more formatting options

Text in editor:

```
```{r echo=FALSE, message=FALSE}
library(kableExtra)

kable(table_sepal_length, digits = 2) %>%
  kable_styling(bootstrap_options = c("striped"),
                full_width = F) %>%
  add_header_above(c(" ", "Sepal Length^1^" = 2)) %>%
    # first column no header, next 2 columns have header
  add_indent(c(1, 2, 3)) %>%
    # specifying rows 1-3 of table; column names aren't a row
  footnote(general = "Fisher's Iris dataset",
           number = c("n = 150", "Data collected by Anderson"),
           alphabet = c("Lengths measured in cm")
           )
```
```

Output:

	Sepal Length[1]	
Species	**mean**	**SD**
setosa	5.01	0.35
versicolor	5.94	0.52
virginica	6.59	0.64

Note:
Fisher's Iris dataset
[1] n = 150
[2] Data collected by Anderson
[a] Lengths measured in cm

See Hao Zhu's webpage for many, many more `kableExtra` options.

Global chunk options

- You can set **global chunk options** that are **applied to all chunks** in the .Rmd file
  - Set global options in a chunk at the beginning of the .Rmd file
  - The template .Rmd file already includes a chunk labeled `setup`
  - Add more options as desired to this chunk
- Options are added within the `knitr::opts_chunk$set(...)` command
- Any of the many chunk options can be set in the `setup` chunk

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(
  fig.height=3, fig.width=7, fig.path='figs', fig.align = "center",
  echo = TRUE,
  warning=FALSE, message=FALSE,
  options(knitr.tables.format = `markdown`)
  )
```
````

- `fig.path` sets the folder name where figures generated by the .Rmd file will be saved
- See the R Markdown cheatsheet for more chunk options.

Practice! (part 1)

Edit the file `example2/example2.Rmd` to create html output that matches `example2/example2_output.html` shown below.

Data summary

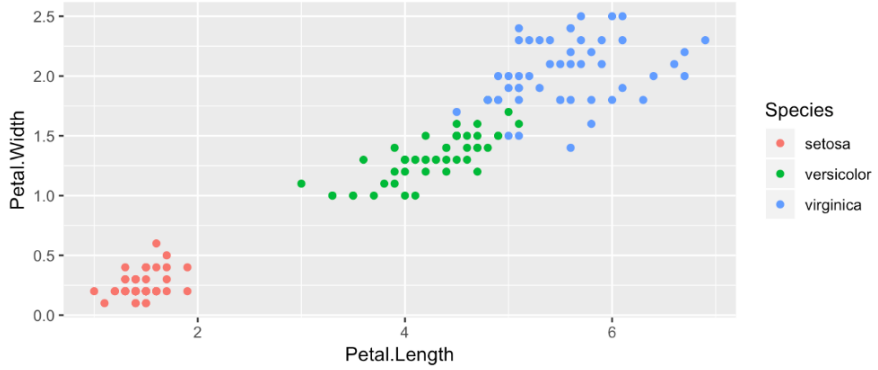
```
library(tidyverse)
library(knitr)
library(kableExtra)
```

Data: Fisher's (or Anderson's) Iris data set.

Petal Widths

Figure

Scatterplot of petal widths vs. length by species type



Summary statistics

The following table summarizes petal widths by species type:

Raw table

```
## # A tibble: 3 x 4
##   Species     mean    SD median
##   <fct>      <dbl> <dbl>  <dbl>
## 1 setosa     0.246 0.105    0.2
## 2 versicolor 1.33  0.198    1.3
## 3 virginica  2.03  0.275    2
```

Simple table

Species	mean	SD	median
setosa	0.2	0.1	0.2
versicolor	1.3	0.2	1.3
virginica	2.0	0.3	2.0

Somewhat better

Species	mean	SD	median
setosa	0.2	0.1	0.2
versicolor	1.3	0.2	1.3
virginica	2.0	0.3	2.0

Even better

	Petal Width[1]		
Species	mean	SD	median
setosa	0.2	0.1	0.2
versicolor	1.3	0.2	1.3
virginica	2.0	0.3	2.0

[1] n = 50 for each species

Practice! (part 2)

Create the table output shown below and at the end of `example2/example2_output.html` (code link)

If you're already done...

Use code from
https://haozhu233.github.io/kableExtra/awesome_table_in_html.html
to figure out how to make the table to the right.

	Petal Width[1]		
Species	**mean**	**SD**	**median**
setosa	0.2	0.1	0.2
versicolor	1.3	0.2	1.3
virginica	2.0	0.3	2.0

[1] n = 50 for each species

3 types of R Markdown content

1. Text
2. Code chunks
3. *YAML metadata*

YAML metadata

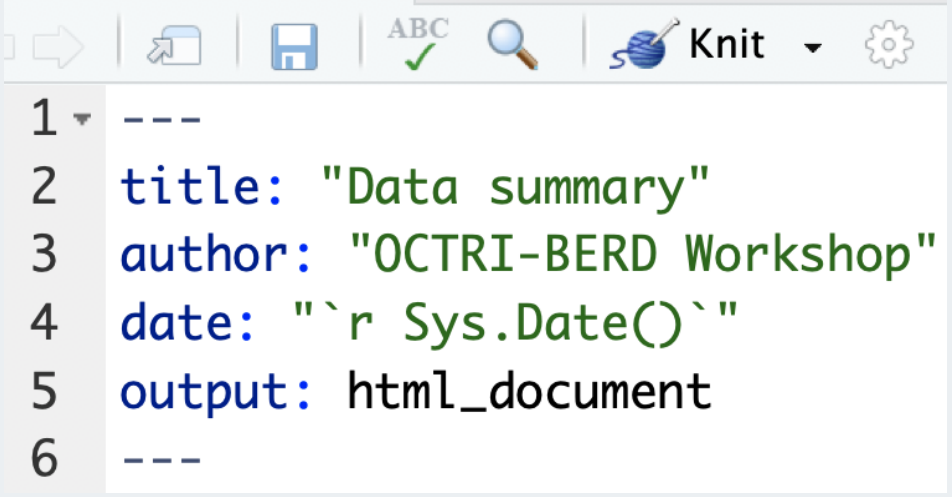
Many output options can be set in the **YAML metadata**, which is the *first set of code in the file starting and ending with ---*.

- YAML is an acronym for *YAML Ain't Markup Language*
- It sets the configuration specifications for the output file
- For more details about YAML in general, see the YAML Wikipedia page

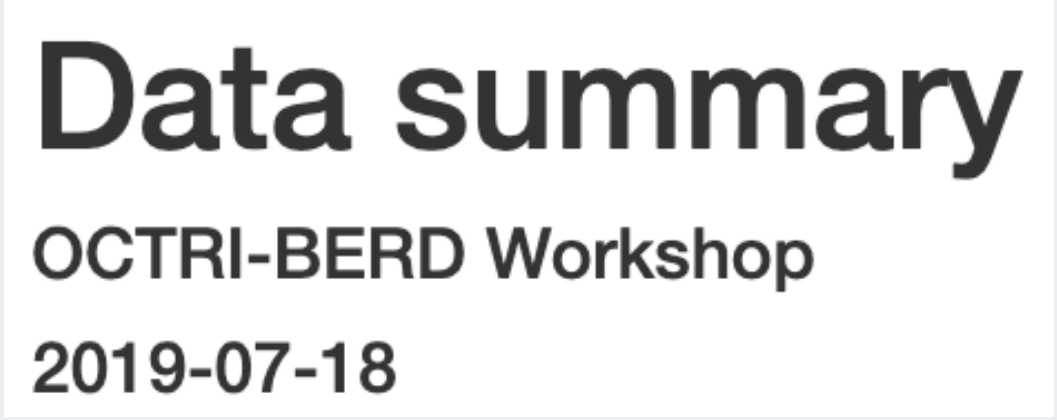
---

Set the **title**, **author**, and **date** that appear at the top of the output file

Text in editor:



Output:



Numbered sections & clickable table of contents

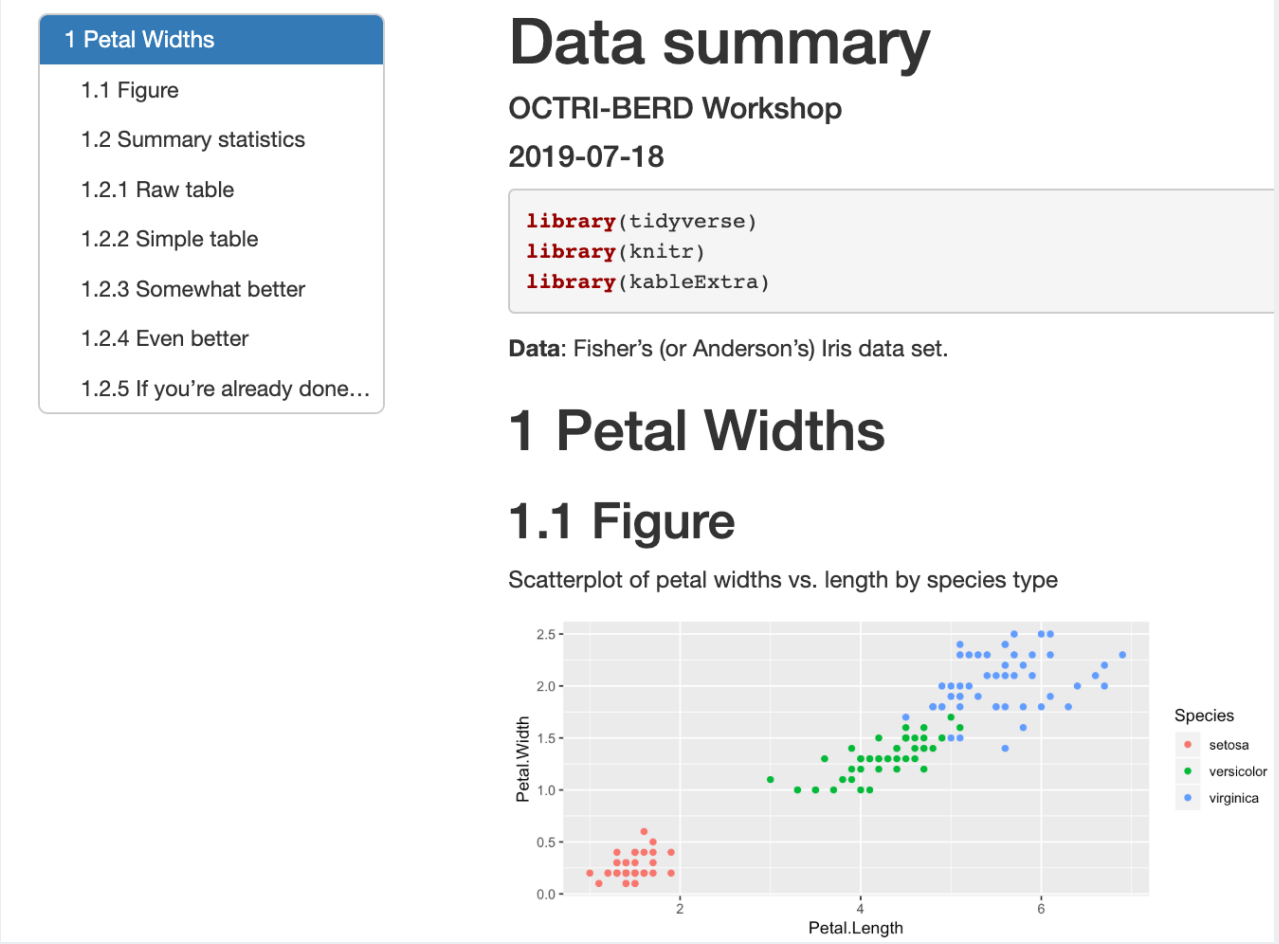
Text in editor: (`example3a.Rmd`)

Knit

```yaml
---
title: "Data summary"
author: "OCTRI-BERD Workshop"
date: "`r Sys.Date()`"
output:
  html_document:
    number_sections: yes
    toc: yes
    toc_float:
      collapsed: no
      smooth_scroll: yes
---
```

Try out `collapsed: yes` and `smooth_scroll: no`

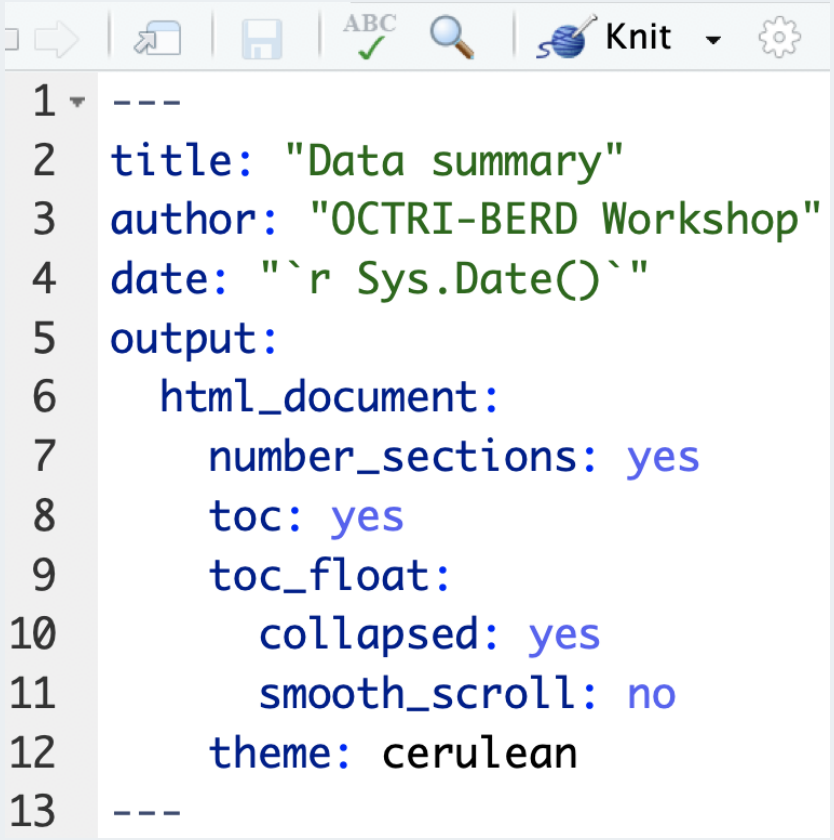
Output: (`example3a.html`)



Themes

- There are 12 themes to choose from without installing additional packages
- See http://www.datadreaming.org/post/r-markdown-theme-gallery/ for examples

Text in editor: (`example3b.Rmd`)



Code folding

- Code folding creates buttons in the output html file that lets users choose whether they want to see the R code or not
  - This only applies to R code from chunks with `echo = TRUE`
- `code_folding: hide` all R code hidden by default; user must click Code button to see R
- `code_folding: show` all R code shown by default; user must click Code button to hide R
- See https://bookdown.org/yihui/rmarkdown/html-document.html#code-folding for more info

Word documents

- Not many YAML options
- Cannot include html code or html-specific commands

Text in editor: (`Word_example3.Rmd`)

Knit

```
---
title: "Data summary"
author: "OCTRI-BERD Workshop"
date: "`r Sys.Date()`"
output:
  word_document:
    toc: yes
---
```

Output: (`Word_example3.docx`)

Data summary

OCTRI-BERD Workshop

2019-07-18

Table of Contents

Petal Widths 1

Figure 1

Summary statistics 1

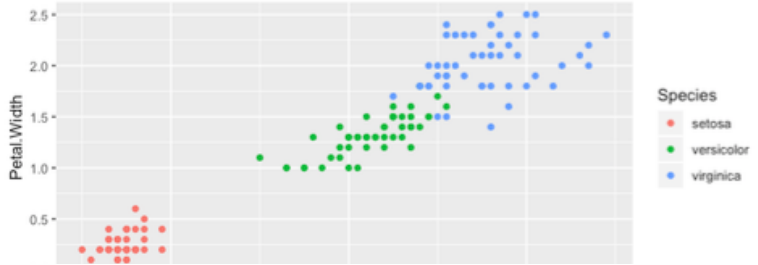

```
library(tidyverse)
library(knitr)
library(kableExtra)
```

Data: Fisher's (or Anderson's) Iris data set.

Petal Widths

Figure

Scatterplot of petal widths vs. length by species type



Word documents - tables options limited

- Cannot use `kableExtra` package options
- `kable` can be used

Summary statistics

The following table summarizes petal widths by species type:

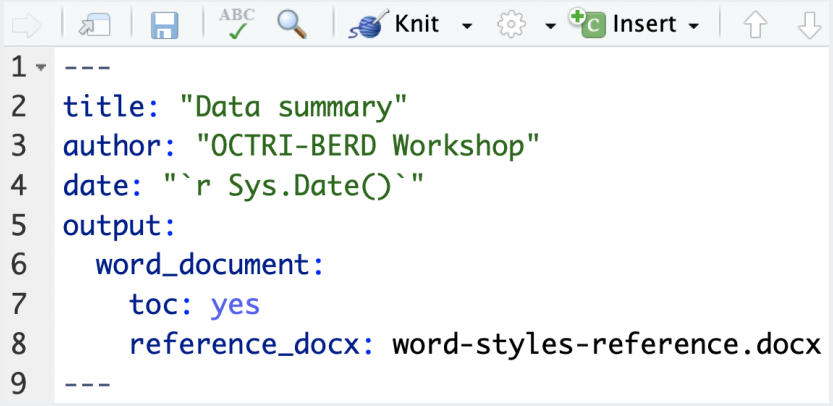
```
kable(table_petal_width, digits=1,
      format="markdown")
```

Species	mean	SD	median
setosa	0.2	0.1	0.2
versicolor	1.3	0.2	1.3
virginica	2.0	0.3	2.0

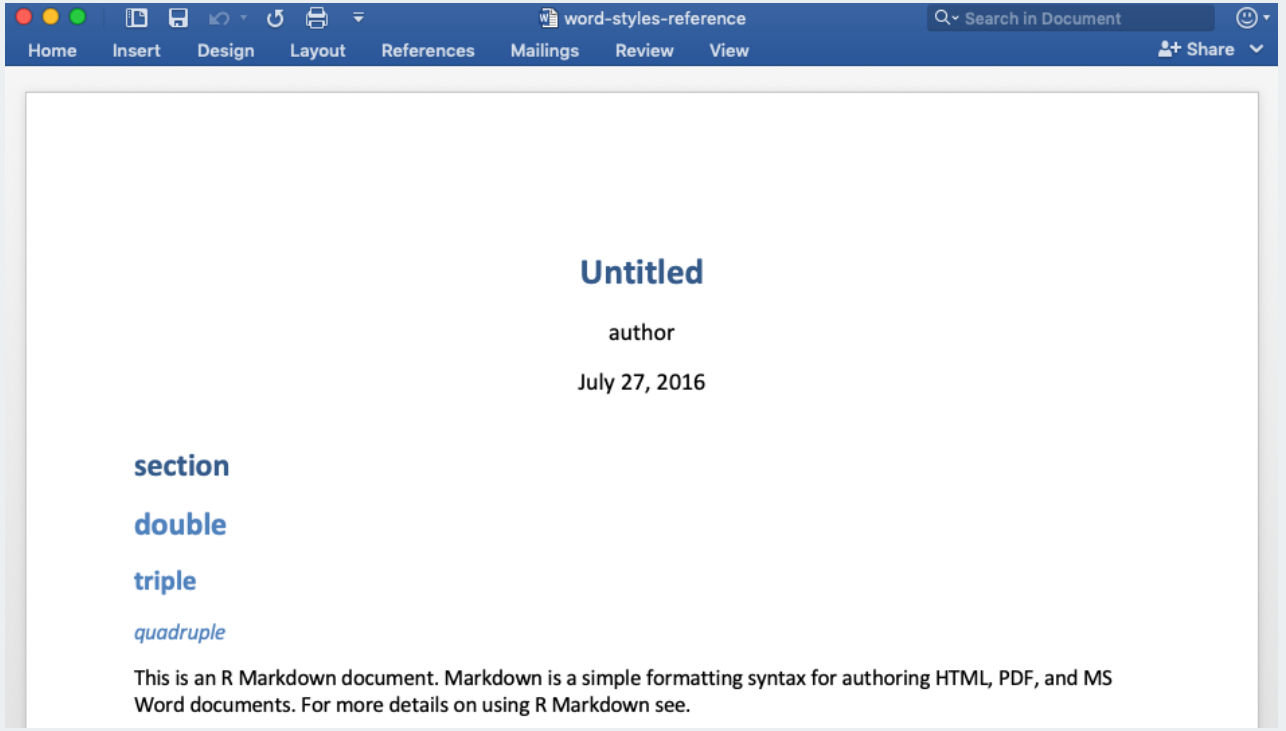
Word documents - using a style file

- Create a Word doc with preferred formatting
  - font types and sizes, margins, header colors, etc.

YAML with code to include style file:



Sample style file: (`word-styles-reference.docx`)

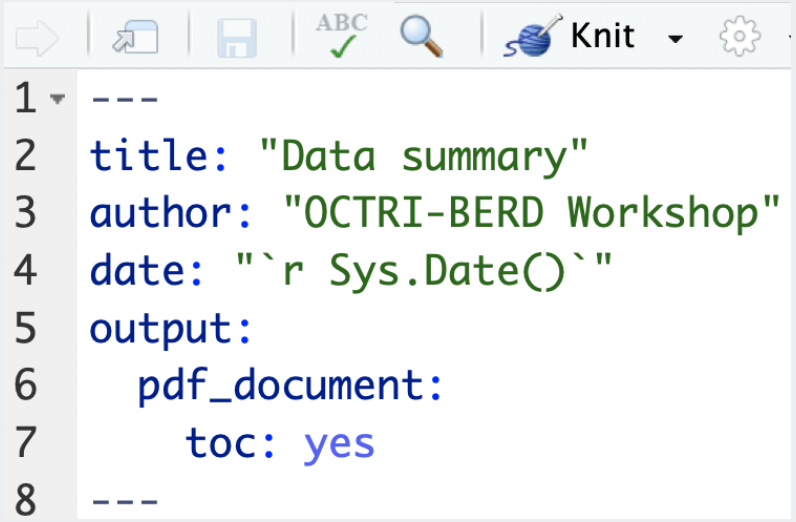


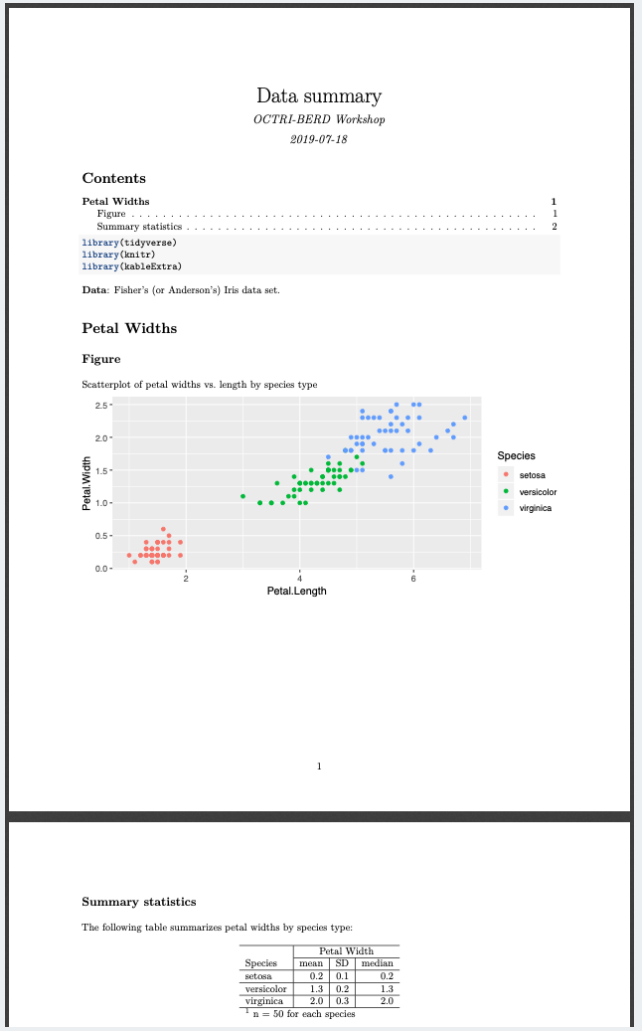
The Word doc created by RStudio will have the same formatting as the specified style file.

pdf documents

Producing pdf documents requires that LaTeX be installed on your computer

- Few YAML options
- Lots of table options, including kableExtra
- Can use LaTeX code for formatting





See `pdf_example3.Rmd` for code and `pdf_example3.pdf` for output.

Practice!

Change the YAML of `example2/example2.Rmd` to

1. Add your name as author
2. Produce a Word document or a pdf document

Extensions and Tips

Real time knitting: `xaringan::inf_mr()`

Instead of clicking "Knit" every time to see your updated document output, try this:

After installing the `xaringan` package,

`.Rmd` files can be run and rendered "live" as you type/save when you either run

```
xaringan::inf_mr()
```

in the console when your `.Rmd` file is open. *Or,* click on on Adddins (top of screen), scroll down to "Xaringan" and click on "Infinite Moon Reader"

This is a new feature, so you need the most recent version of `xaringan` and RStudio. It works well for `html_document` output.

Reproducible Workflow

Be Organized

Your files must make sense to yourself 6 months from now, and/or other collaborators.

Jenny Bryan's "What They Forgot to Teach you About R" RStudio::conf2018 training

No! Absolute! File! Paths! (don't `setwd()`)

Absolute paths ≠ reproducible

Relative paths = reproducible (if done correctly)

> *If the first line of your R script is*
>
> ```
> setwd("C:\Users\jenny\path\that\only\I\have")
> ```
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

Jenny Bryan's oft quoted opinion; see post on Project-oriented workflow

Project directory structure

- The .Rproj file sets your working/home directory (**USE PROJECTS**)

```
# Use a relative path, "relative to" the project folder
read_csv("mydata.csv") # looks in .Rproj folder
```

- When .Rmd files knit, they look for sourced files *in the folder they live in*

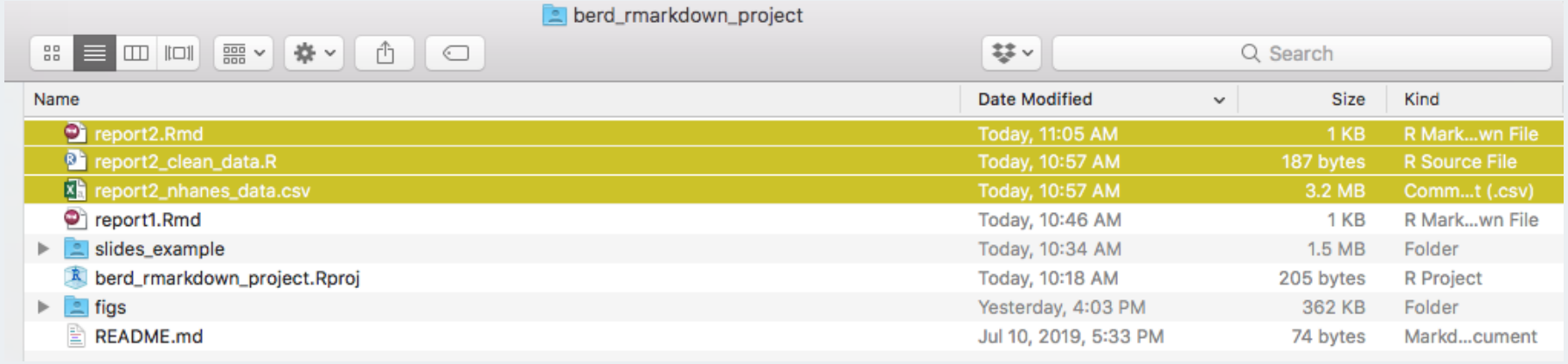
````
```{r data, eval=TRUE}
read_csv("mydata.csv") # looks in .Rmd's folder
```
````

- It's good practice to organize all your code/data/output into separate folders

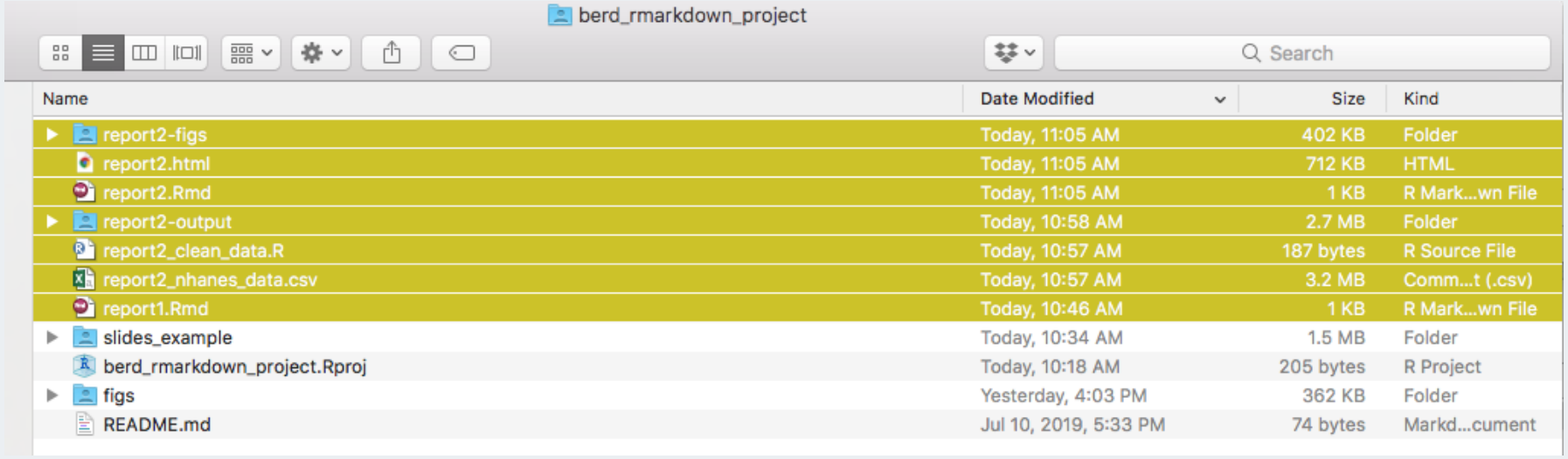
These three facts together can cause a headache.

- Enter `here::here()`!

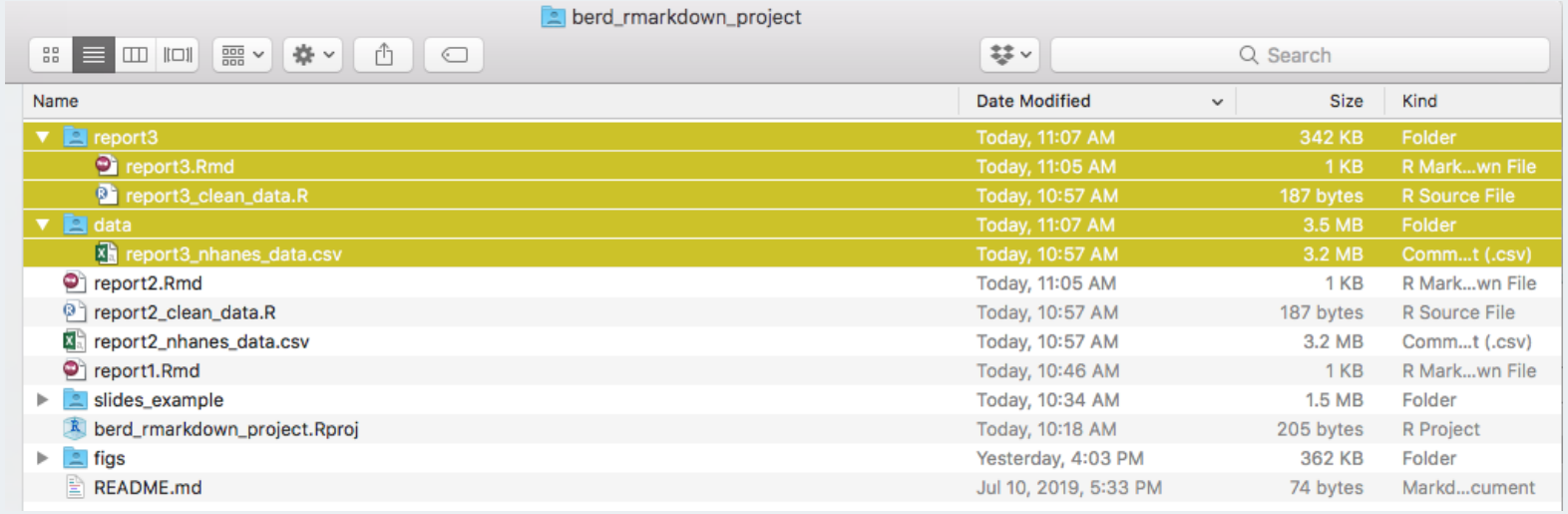
Everything in one folder



After knitting, this gives you (file)



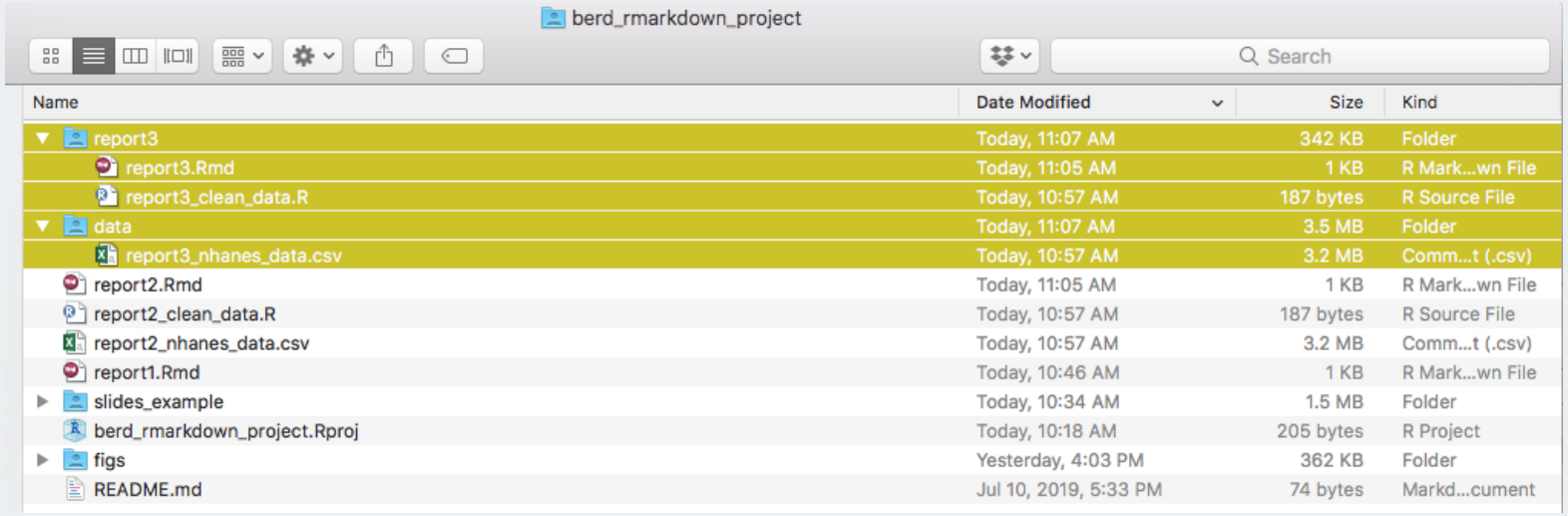
Slightly more organized



Dot dot: A tip about "moving up" a directory/folder

- In unix, to point to the folder one level up (it contains the folder you're in), use `..` or `../`
  - As in `cd ..` moves up one directory,
  - or `cp ../myfile.txt newfile.txt` copies a file one level up into the current folder (working directory)
- In `.Rmd` when you want to source the data in the `data/` folder, you could use `..` to move up a folder into the main directory, and then back down into the `data/` folder:

```
# From the .Rmd folder, move up one folder then down to the data folder
mydata <- read_csv("../data/report3_nhanes_data.csv")
```



Find the . . confusing? Use here::here()!

Allison Horst

here::here() → relative paths to the project directory

- The **here** package's **here()** function solves this issue of inconsistent working directories.
- The point of RStudio project workflow is to always have the same "home" working directory = where the **.Rproj** file is.
- **here::here()** returns the project directory as a string
- Fully reproducible if the whole folder is moved or shared or posted to github
- Portable to ALL systems (Mac, PC, unix), don't worry about / (Mac) or \ (PC) or spaces etc

```
here::here()
```

```
[1] "/Users/niederha/Google Drive/BERD R Classes/berd_r_courses_github"
```

here::here() with folders and filenames

- `here::here("folder","filename")` returns the entire file path as a string
- These file paths work when running a `.Rmd` file interactively like a notebook, when knitting it, when copying it to the console, wherever, whenever!!

```
here::here("data","mydatafile.csv")
```

```
[1] "/Users/niederha/Google Drive/BERD R Classes/berd_r_courses_github/data/mydatafile.csv"
```

```
here::here("data","raw-data","mydatafile.csv")
```

```
[1] "/Users/niederha/Google Drive/BERD R Classes/berd_r_courses_github/data/raw-data/mydataf
```

We will explore how and when to use this in the exercises.

Practice!

Within your project folder, open this file and follow the instructions:

- `example4/example4.Rmd`

More Extensions and Tips

Even more organized: child documents

If you want to have separate `.Rmd` files that are sourced in one large document, you can have "child document chunks":

A file called `report_prelim.Rmd` in the `analysis/` folder

(No YAML):

````
# Details about experiment

Here are some details.
I can make a plot, too.

```{r plotstuff}
plot(x,y)
```
````

In the main doc `main_doc.Rmd`

````
---
title: "Main Report:
output: html_document
---

# Preliminary Analysis

```{r child = here("analysis","report_prelim.Rmd")}

```

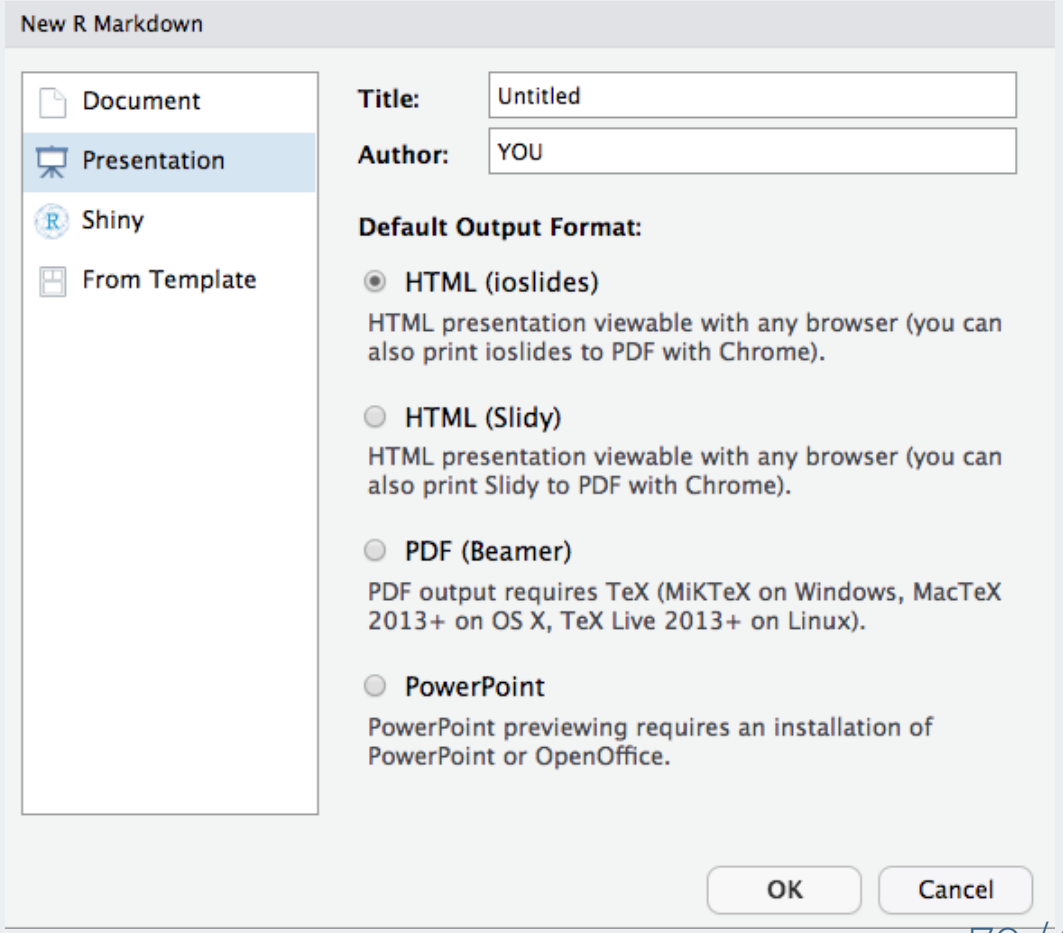
# Conclusion

```{r}
kable(summarytable)
```
````

Make presentation slides

- These slides were made using a `.Rmd` file with the `xaringan` package!
- Simple templates can be found in `File -> new File -> R Markdown -> Presentation`

- Each type of presentation uses different syntax to start a new slide, such as
  - `# Slide Header` , or
  - `---`
- ioslides and Slidy are html slides; simple options
- Beamer is from LaTeX
- Xaringan is html based on java script remark.js; has the most flexibility for customizing slides
- PowerPoint is in the newest RStudio release; can use custom templates



Presentations Practice!

Open `example4/example4_pres.Rmd` and follow instructions.

Bonus: Try using `xaringan::inf_mr()` to update the output in real time.

Tabsets

A nice feature for showing multiple images or sections is with tabbed sections:

````
## Results {.tabset}

### By Species

```{r}
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Wi
  geom_point()
```

### Panel Species

```{r}
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Wi
geom_point()+
  facet_wrap(~Species)
```
````

Using other programming languages

- RStudio can run multiple programming languages in the same `.Rmd` (if they are installed on the computer), including SAS, STATA, and python.
- For more on how to use STATA and SAS, for example, see the documentation for these packages:
  - StataMarkdown
  - SASMarkdown

```
names(knitr::knit_engines$get())
```

```
 [1] "awk"        "bash"       "coffee"     "gawk"       "groovy"
 [6] "haskell"    "lein"       "mysql"      "node"       "octave"
[11] "perl"       "psql"       "Rscript"    "ruby"       "sas"
[16] "scala"      "sed"        "sh"         "stata"      "zsh"
[21] "highlight"  "Rcpp"       "tikz"       "dot"        "c"
[26] "fortran"    "fortran95"  "asy"        "cat"        "asis"
[31] "stan"       "block"      "block2"     "js"         "css"
[36] "sql"        "go"         "python"     "julia"      "sass"
[41] "scss"
```

Other languages: Limitations

- Each code chunk is run separately as a batch job when using other languages, so it's tricky to pass on objects/data to later code chunks.
- Easy way:
  - Use one language to clean data & save the cleaned data as a file
  - source the file and continue in another language.
- Other packages can be loaded that help to link objects from various languages, i.e.
  - reticulate can store objects created by python code for use in R
  - StataMarkdown and SASMarkdown use chunk option `collectcode=TRUE` to save code output.

````
```{r setup}
library(SASmarkdown)
```

```{sas clean_data, collectcode=TRUE}
/* clean data with SAS code */
/* export to file */
```

```{sas analyze_data}
/* analyze data from above code */
```

```{r analyze_data}
# source clean data file and run code
```
````

Knit other types of output

- Journal articles, custom templates
  - File → New File → R Markdown → From template
- Dashboards: flexdashboard report output
- Interactive reports with shiny
- Interactive tutorials with learnr
- Websites: blogdown
- Books: bookdown
- Posters: posterdown
- Grad school theses: thesisdown
- It's really endless....

`rmarkdown::render()`

It can sometimes be easier to set options and change output files/locations when using the `render()` function in the `rmarkdown` package. This is also useful for rendering multiple documents in a batch, or using parameterized reports.

In a `.R` file, or in the console, run commands to knit the documents:

```
library(rmarkdown)
render("report1.Rmd")

# Render in a directory
render(here::here("report3","report3.Rmd"))

# Render a single format
render("report1.Rmd", output_format = "html_document")

# Render multiple formats
render("report1.Rmd", output_format = c("html_document", "pdf_document"))

# Render to a different file name or folder
render("report1.Rmd",
       output_format = "html_document",
       output_file = "output/report1_2019_07_18.html")
```

`knitr::purl()` → `.R` file

Run in the console or keep in a separate R file to extract all the R code into a `.R` file.

```
# makes an R file report1.R in same director
knitr::purl("report1.Rmd")

# Can be more specific with output
knitr::purl(here::here("report3","report3.Rmd"), # Rmd location
            out = here::here("report3","report3_code_only.R")) # R output location
```

`knitr::knit_exit()`: End document early

- Exit the document early.
- Place this in your `.Rmd` to end document there and ignore the rest.
- Run parts of the document at a time

````
```{r}
knitr::knit_exit()
```
````

Parameterized Reports

````
---
title: My Report
output: html_document
params:
  data: file.csv
  printcode: TRUE
  year: 2018

---

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

```{r}
mydata <- read_csv(params$data)
mydata <- mydata %>%
  filter(year==params$year)
```
````

- Use the Knit button and you will be prompted for values
- Use `rmarkdown::render` (default values are set in YAML)
- See chapter in R Markdown book for details

```
rmarkdown::render(
  "myreport.Rmd",
  params =
    list(data = "newfile.csv",
         year = "2019",
         printcode = FALSE),
  output_file = "report2019_newfile.html"
)
```

Many more bonus tips

- Use git and github for version control, and use output format github_document - see an example
- Quickly convert `.R` files to `.html` with the notebook/compile button or `knitr::spin()`
- Include HTML headers or Latex preambles and files for definitions in YAML
- Add references and a bibliography with BibTex `.bib` files
- Similar to `.Rmd` are RStudio "notebooks" -- like an `.Rmd` but all the output is saved as it is run in the notebook.
- Publish rendered html on Rpubs with Publish button, or through github + netlify.
- Look at these slides by Alison Hill and these by Yihui Xie for many, many more tips and examples

References

- RStudio's R Markdown lessons
- Xie Y. et al R Markdown: The Definitive Guide book online
- Explanation of difference between knitr/Rmd/pandoc
- Teach data science: Getting started with R Markdown
- Alison Hill & Yihui Xie's Advanced R Markdown Workshop Materials
- UCLA's Intro to R Markdown slides
- Software Carpentry Learning R Markdown Materials

Cheatsheets:

- R Markdown cheatsheet
- R Markdown reference guide

Possible Future Workshop Topics?

- tables
- ggplot2 visualization
- advanced tidyverse: functions, purrr (apply/map)
- statistical modeling in R

Contact info:

- Jessica Minnier: *minnier@ohsu.edu*
- Meike Niederhausen: *niederha@ohsu.edu*

This workshop info:

- Code for these slides are on github, with links to other course materials: jminnier/berd_r_courses
- The `.Rmd` file that generated the slides is on github and can be downloaded here, though you need to download the whole R project to knit the file.
- The project folder of examples can be downloaded at github.com/jminnier/berd_rmarkdown_project & the solutions are in the `solns/` folder.