Reproducible Reports with R Markdown

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

2019/07/18

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

Load files for today's workshop

-

```
# install.packages("knitr")
library(knitr)
```

Allison Horst

Learning objectives

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)

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."

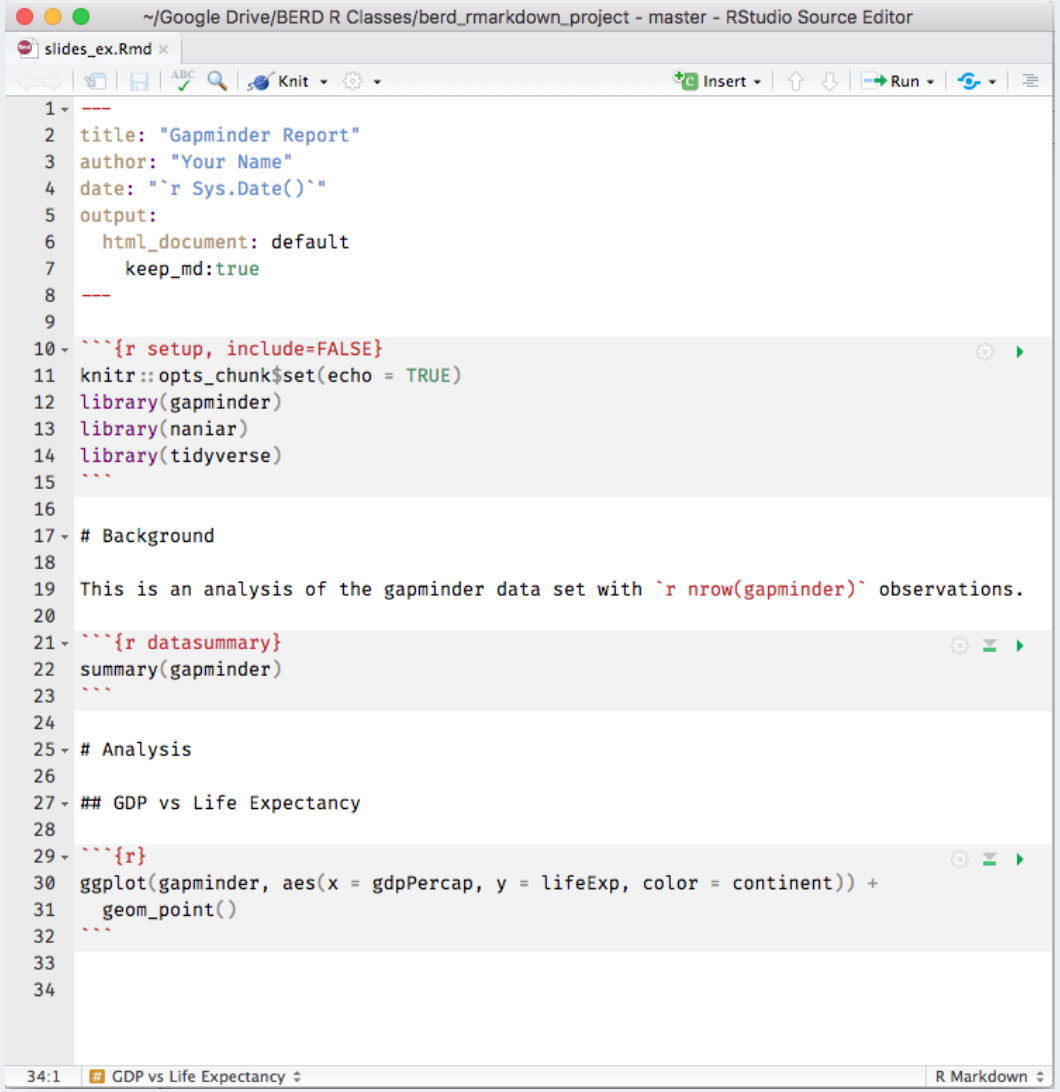
R Opensci Reproducibility Guide

R Markdown = `.Rmd` file

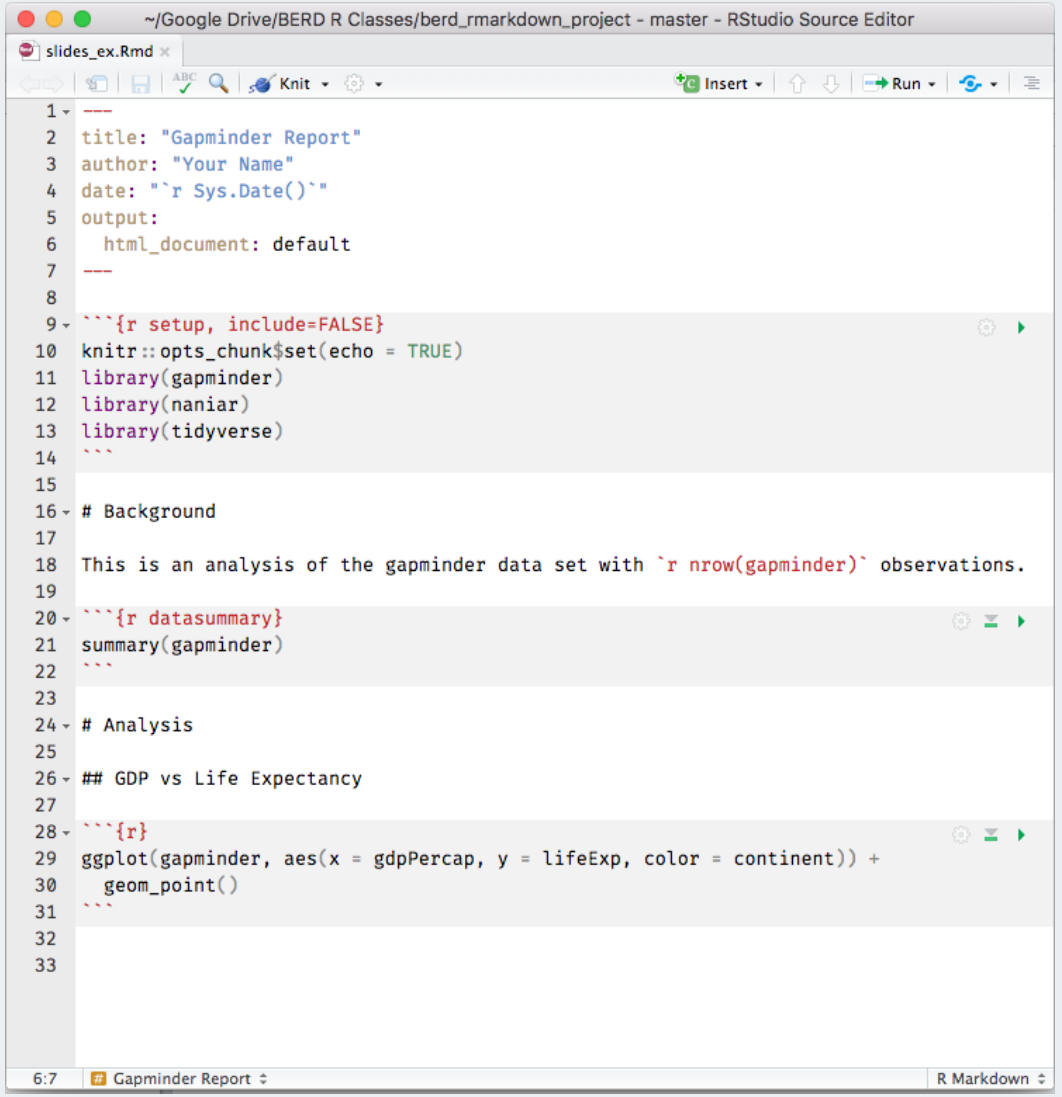
Code + text (in markdown syntax)

`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



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)
```

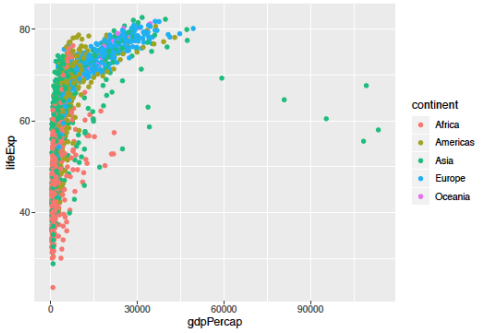
```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   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

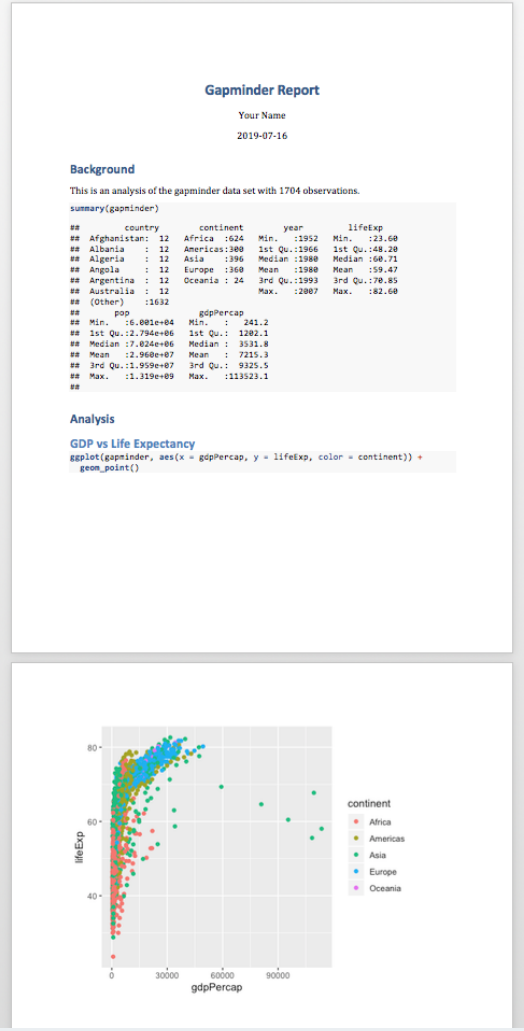
knitr converts .Rmd -> .md -> .doc

~/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

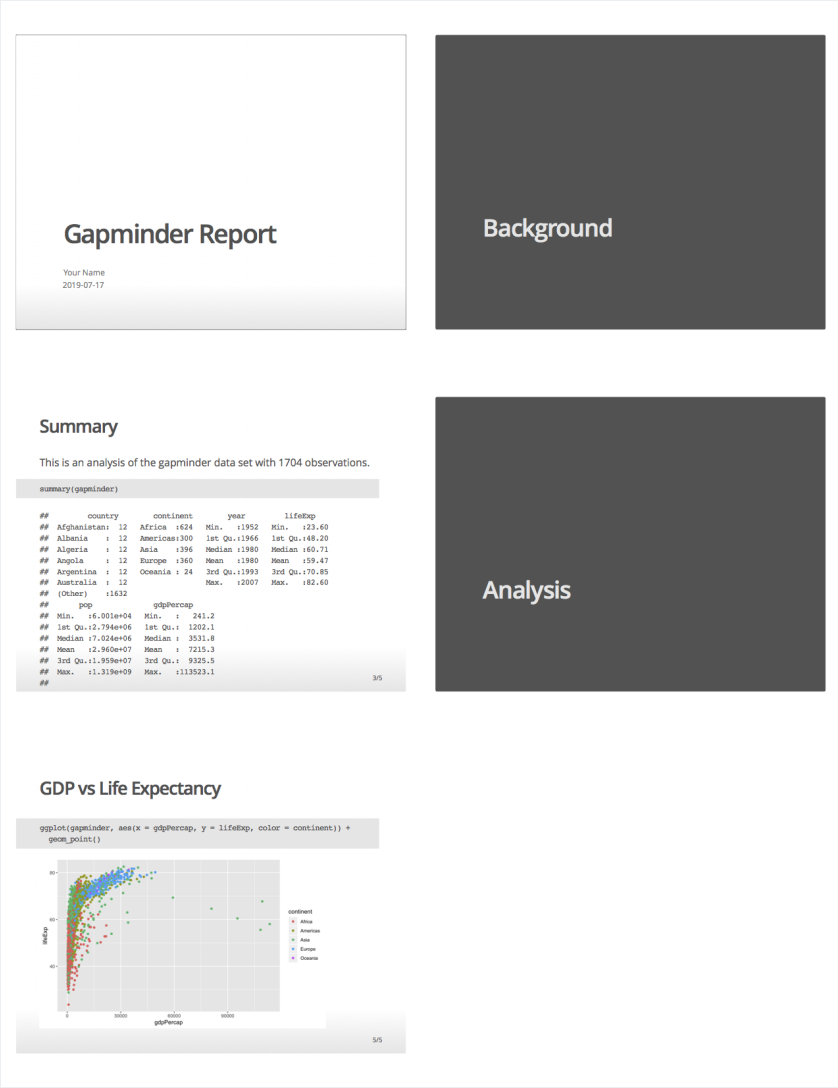
~/Google Drive/BERD R Classes/berd_rmarkdown_project - master - RStudio Source Editor

slides_ex.Rmd

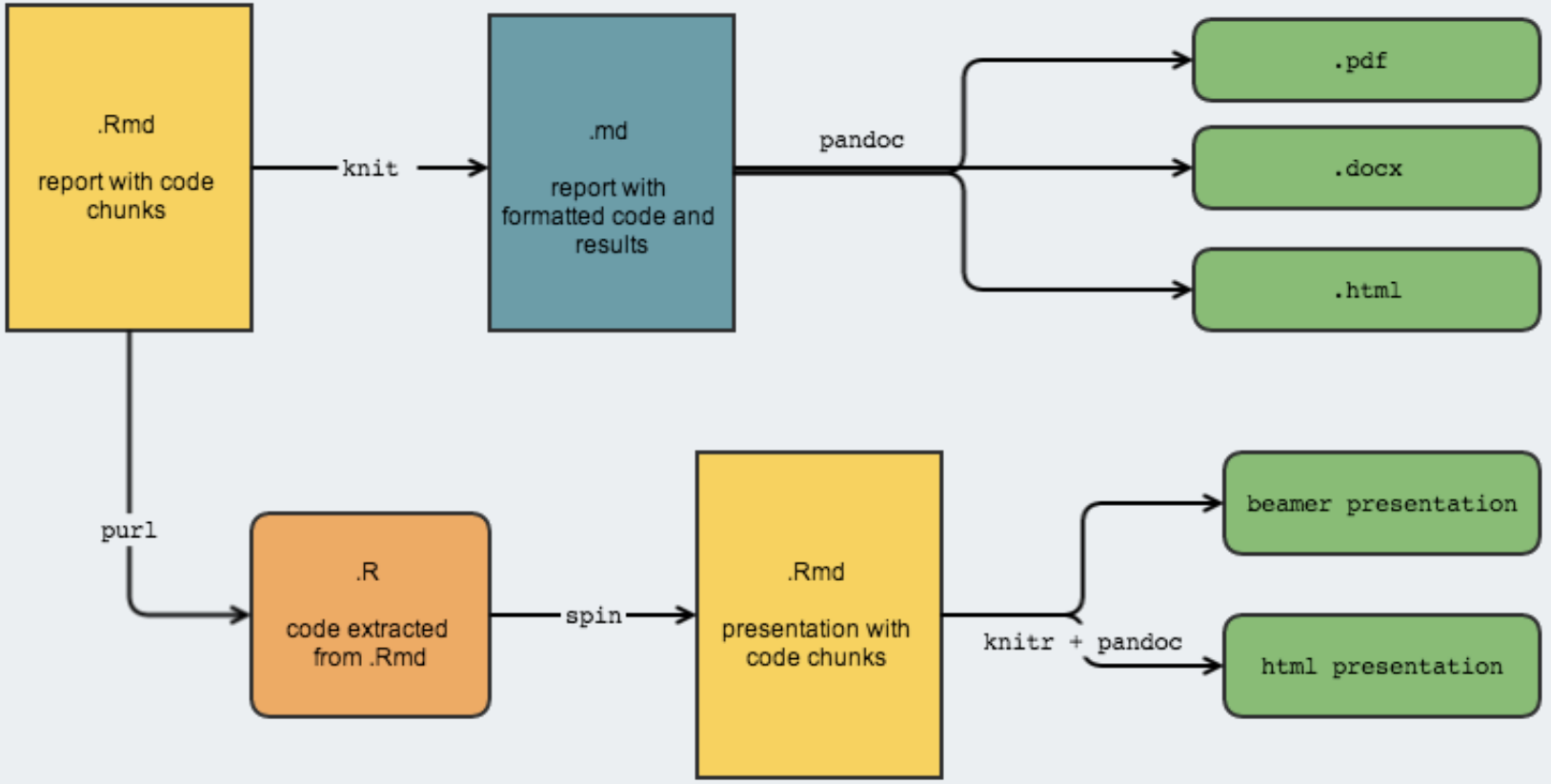
Knit | Insert | Run

```
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
- Sets *working directory*
- A project is associated with a directory folder
  - 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

INSERT MEIKE'S SLIDES HERE

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

- .Rproj sets your working directory (**USE PROJECTS**)

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

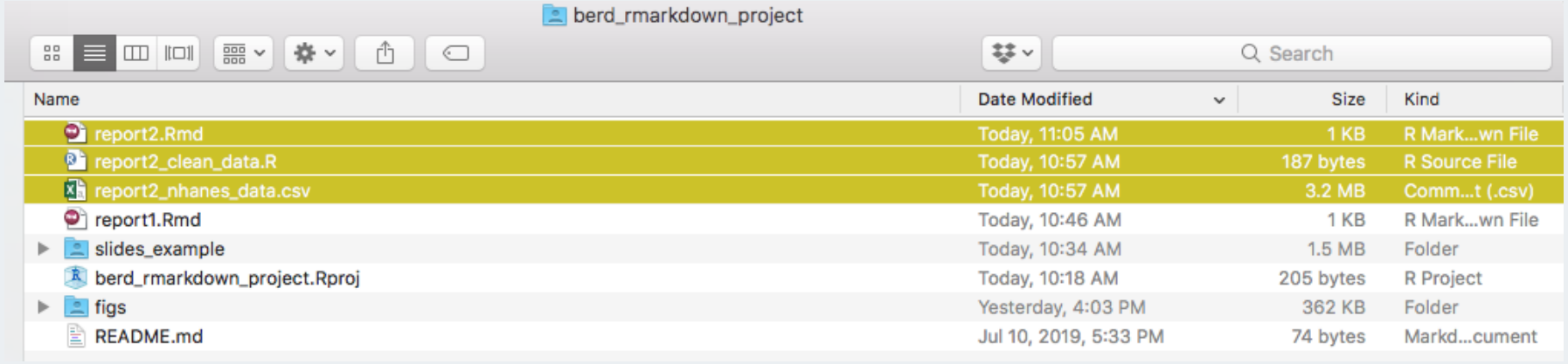
- .Rmd files when knit look for sourced files *in the folder they live in*

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

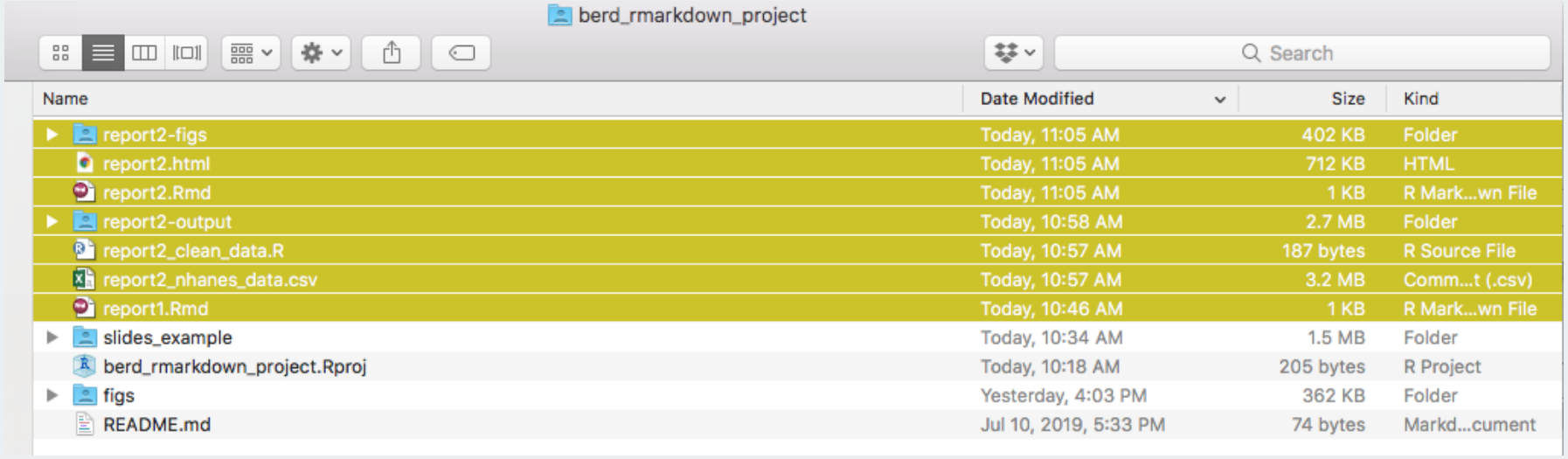
- It's good practice to organize all your code/data/output into separate folders

These three facts together can cause a headache. Enter `here::here()`!

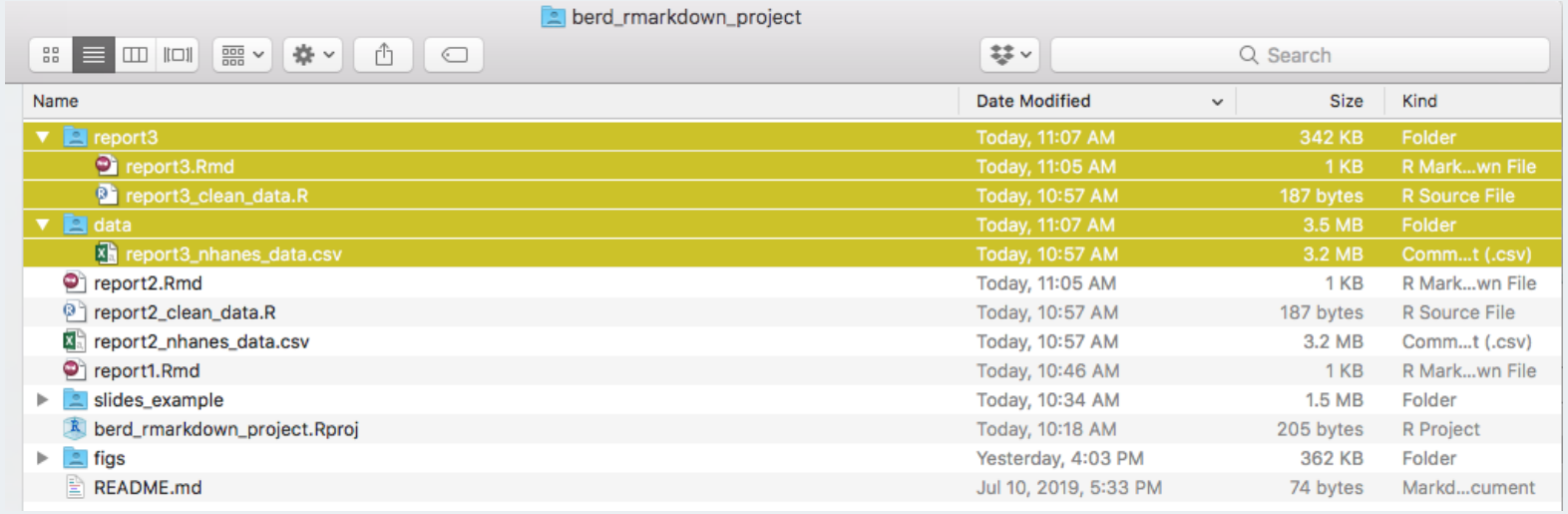
Everything in one folder



After knitting, this gives you (file)



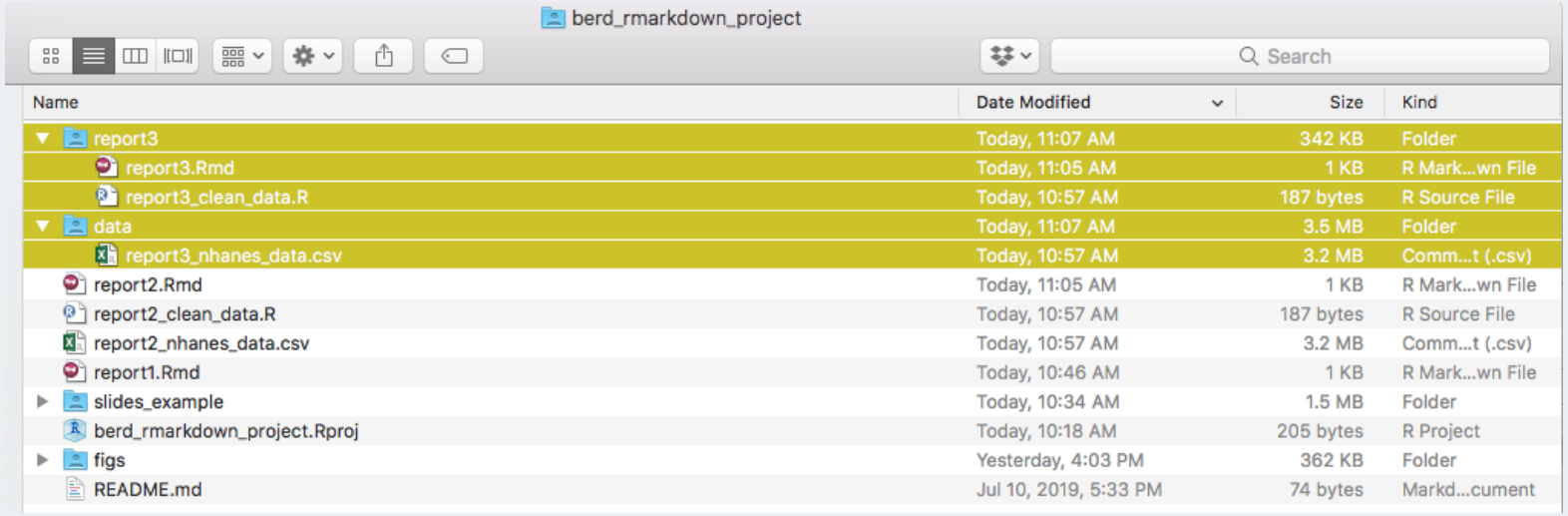
Slightly more organized



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")
```



Exercises:

Within your project folder, open these files and follow the instructions:

- `report2.Rmd`
- `report3/report3.Rmd`

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 `/` or `\` or spaces etc

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

```
[1] "/Users/minnier/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 an `.Rmd` file interactively like a notebook, when knitting it, when copying it to the console, wherever, whenever!!

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

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

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

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

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

Exercises:

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

- `report3-here/report3_here.Rmd`

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)
```
````

Extensions and Tips

Using other languages

- Rstudio can run multiple programming languages in the same `.Rmd` (if they are installed), 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 */
```

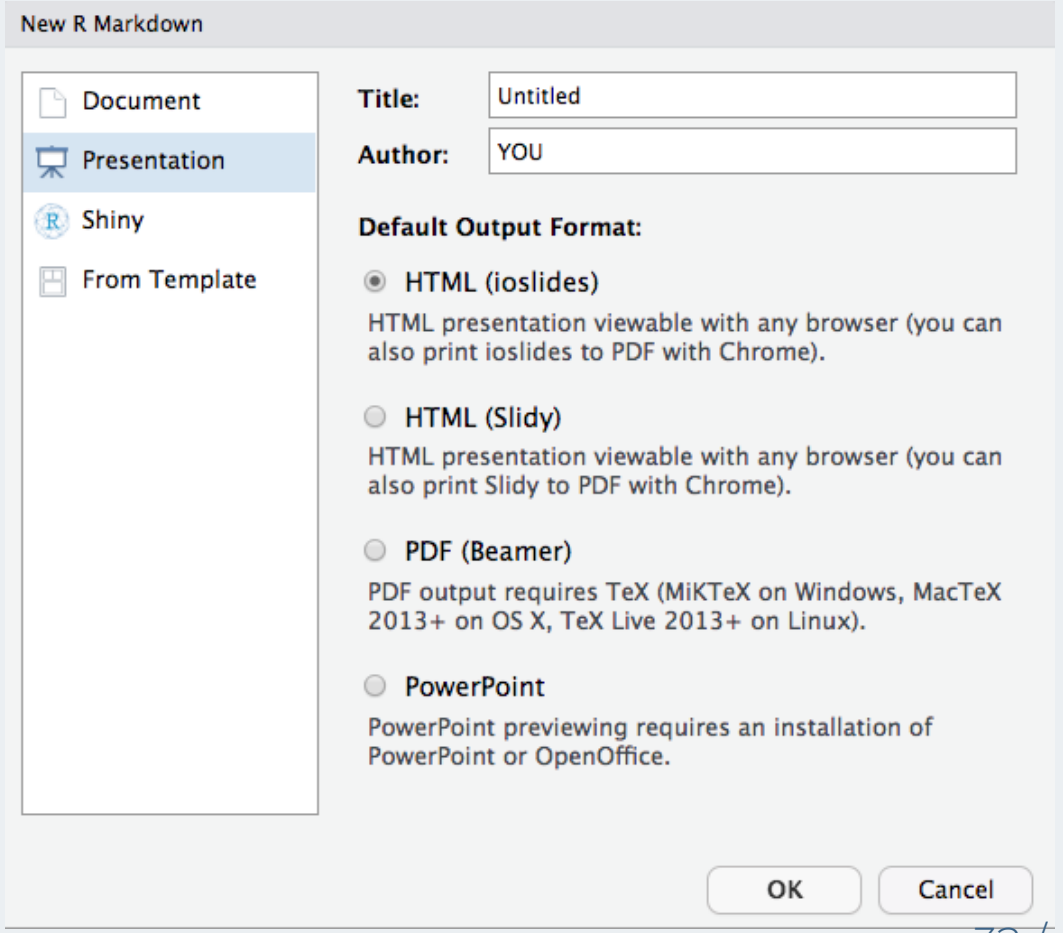
```{sas analyze_data}
/* analyze data from above code */
```

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

Make presentation slides

- These slides were made using `.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
  - `# Slide Header`
  - `---`
- ioslides and Slidy are html slides, simple options
- Beamer is from Latex
- Xaringan (html based on javascript remark.js) has the most flexibility for customizing slides
- PowerPoint is in the newest Rstudio release, can use custom templates



Presentations Exercise

Open `report2_pres.Rmd` and follow instructions.

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()

In an .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 = "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 an `.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

- Instead of Knit, run `xaringan::inf_mr()` in the console (or use the addin) to live-render to html as you change an `.Rmd`
- 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.
- 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 on github: jminnier/berd_r_courses
- all the R code in an R script
- answers to practice problems can be found here: html