Research Workflow in R

Rocky Aikens

Chen Lab Summer 2020 Group Meeting

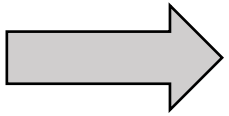
Main Questions I had as a youth

- How do I “save” my work so I don’t cause disasters accidentally when I change things?
- How do I record what I did during an analysis?
- How do I communicate my analyses with other people?
- How do I write my pipelines so that other people (and myself) can read and reuse?
- How do I organize my code/data/notes?
- How do I write my pipelines so that writing papers is easy?
- Later: How do I develop code for public use?

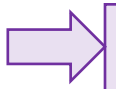
Main Questions I had as a youth

Version Control	• How do I “save” my work so I don’t cause disasters accidentally when I change things?
Documentation	• How do I record what I did during an analysis? • How do I communicate my analyses with other people?
	• How do I write my pipelines so that other people (and myself) can read and reuse?
Workflow	• How do I organize my code/data/notes? • How do I write my pipelines so that writing papers is easy?
Publication	• Later: How do I develop code for public use?

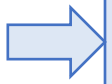
Topics



Arrow indicates exercises for you to do on your computer



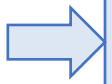
- Prologue: Git and Github



- R projects
- Directory structure

- Writing R code

- Writing R markdowns



- Completing an analysis
- Preview: A Package-based workflow

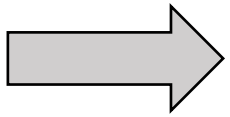
Caveats

- This is what I do and it works well for me. This doesn't have to be what you do.

- My work generally involves shallow codebases with few code-contributing collaborators.
  - I want my code to be reusable, but the primary product of my work is seldom the code itself.
  - I want my code to be easy to share with another R-proficient programmer, but I seldom work actively on a team with other programmers.

Prerequisites

- R and RStudio installed
- Tidyverse installed in R
- Basic working knowledge of R or another programming language



Arrow indicates places where you should follow along on your computer

- Raise your hand on Zoom to indicate when you're done
- Ask for help whenever you need it!

Git and Github

Git and Github – Why?

- Git and Github are used for "version control." Using github lets you save a folder with your current work (and all previous versions) online.
- This is good for when:
  - You can't access the computer with your original work
  - You realize you made a mistake and need to go back in time
  - You want to share your code with collaborators and the public

Github in RStudio

- My workflow:
  - Whenever I start a new project, I set it up as a git repository:
    - see https://happygitwithr.com/new-github-first.html
  - Multiple times a day, I will save my work on github via the Rstudio interface
  - All my code (and previous versions) are available to me and others online!
    - When sharing to collaborators, I'll say:
      "All my work is on github here: https://github.com/raikens1/PilotMatch."
    - When writing papers, I'll write in the methods section:
      "All code and data are publicly available at https://github.com/raikens1/PilotMatch."

Github in RStudio

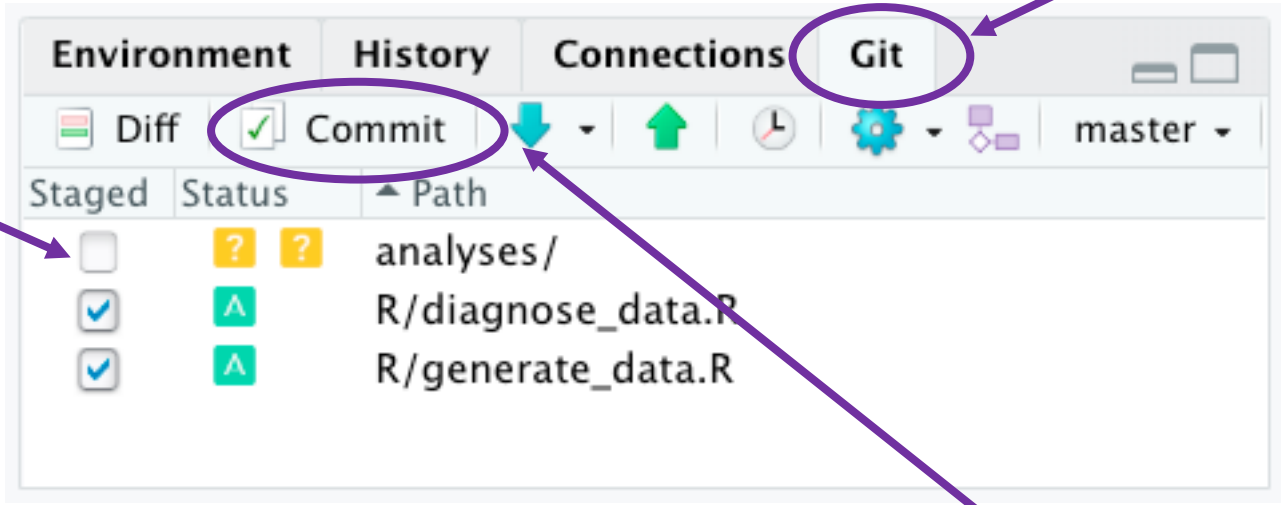
- Find Github intimidating? RStudio has a quite accessible interface, although it takes some setting up.
  - Using Git and Github in RStudio: https://happygitwithr.com/

Github in RStudio

1. Once your git repo is configured, the Git tab in your upper right window will track the changes you make in your folder

2. Choose which files you want to save on github by clicking the checkboxes

(This runs `git add`)

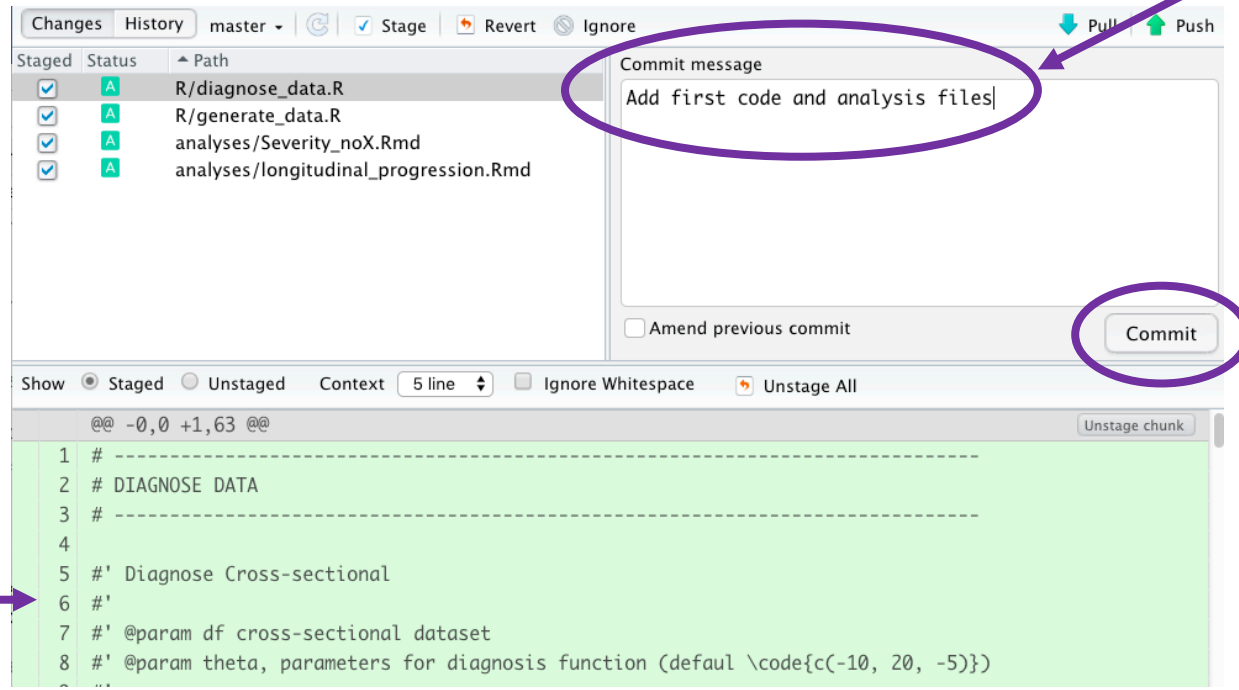


3. Click “Commit” when you want to save your work

Github in RStudio

Hitting “commit” opens a new window:

2. Write a note in the upper right panel to remind you what changes you’re saving



1. The lower pane shows the changes you’ve made to the files you checked

(This is like `git diff`)

3. Click “Commit” to save your message and changes. Then close the window.

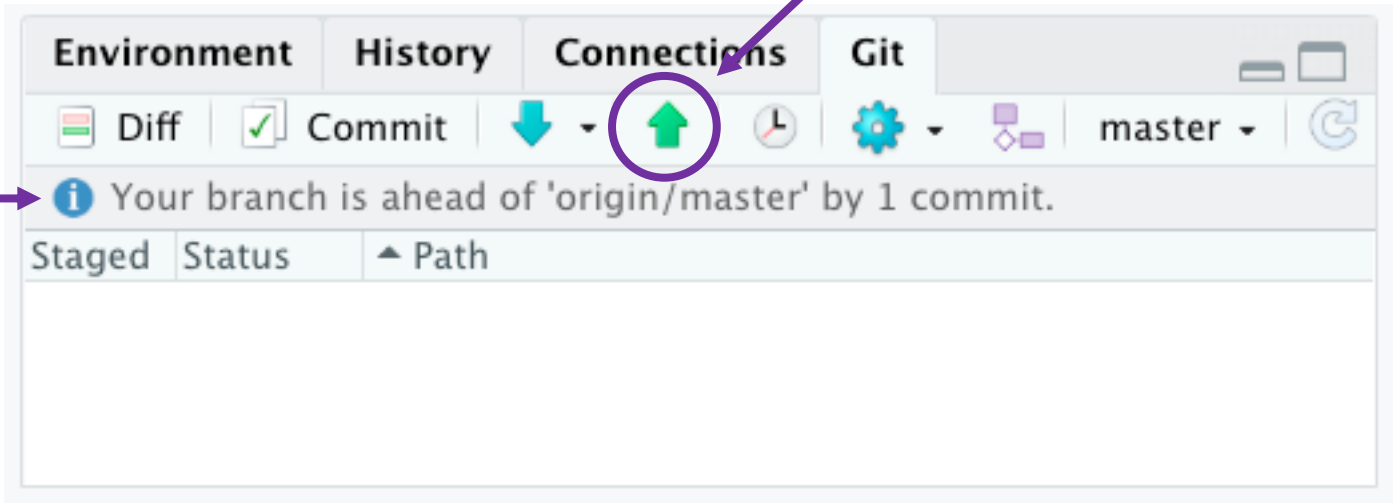
(This runs `git commit`)

Github in RStudio

Last step: Back in RStudio, click the green arrow to upload your changes to github.com

(This runs `git push`)

RStudio will show this banner to remind you when you have changes you haven't uploaded



That's 80% of all the git you need to know

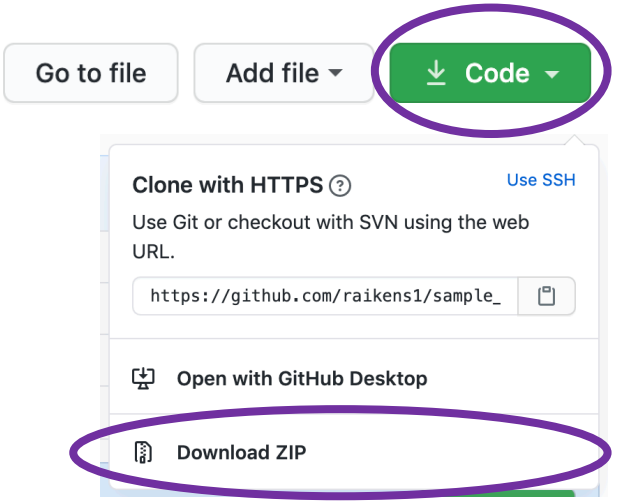
Start: Clone my Repo

In command line:

```
git clone https://github.com/raikens1/sample_analysis_project
```

Or just download it:

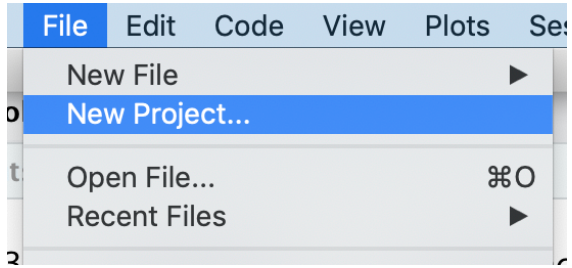
- Go to https://github.com/raikens1/sample_analysis_project
- Click “Code”
- Click “Download Zip”
- Unzip that folder and put it someplace reasonable



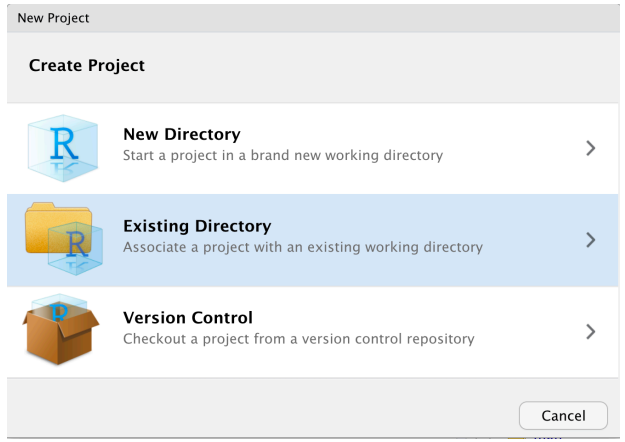
R Projects and Structure

Make it a project

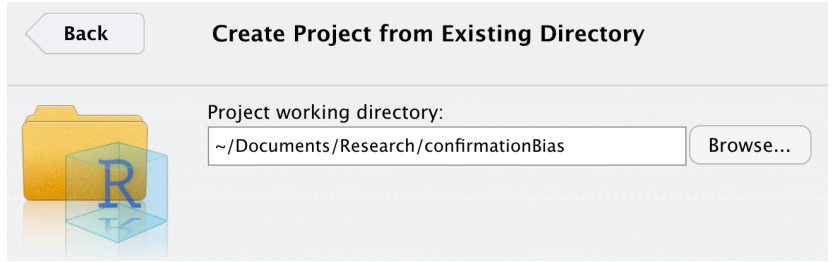
1. In Rstudio:
   Click File >> New Project



2. Select “Existing Directory”



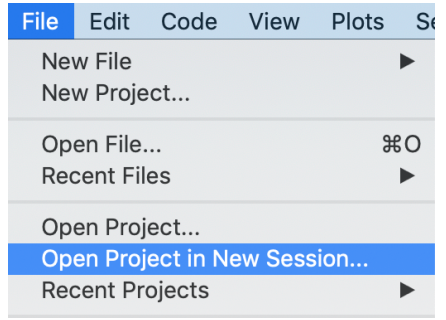
3. Select the folder for the repo you just downloaded



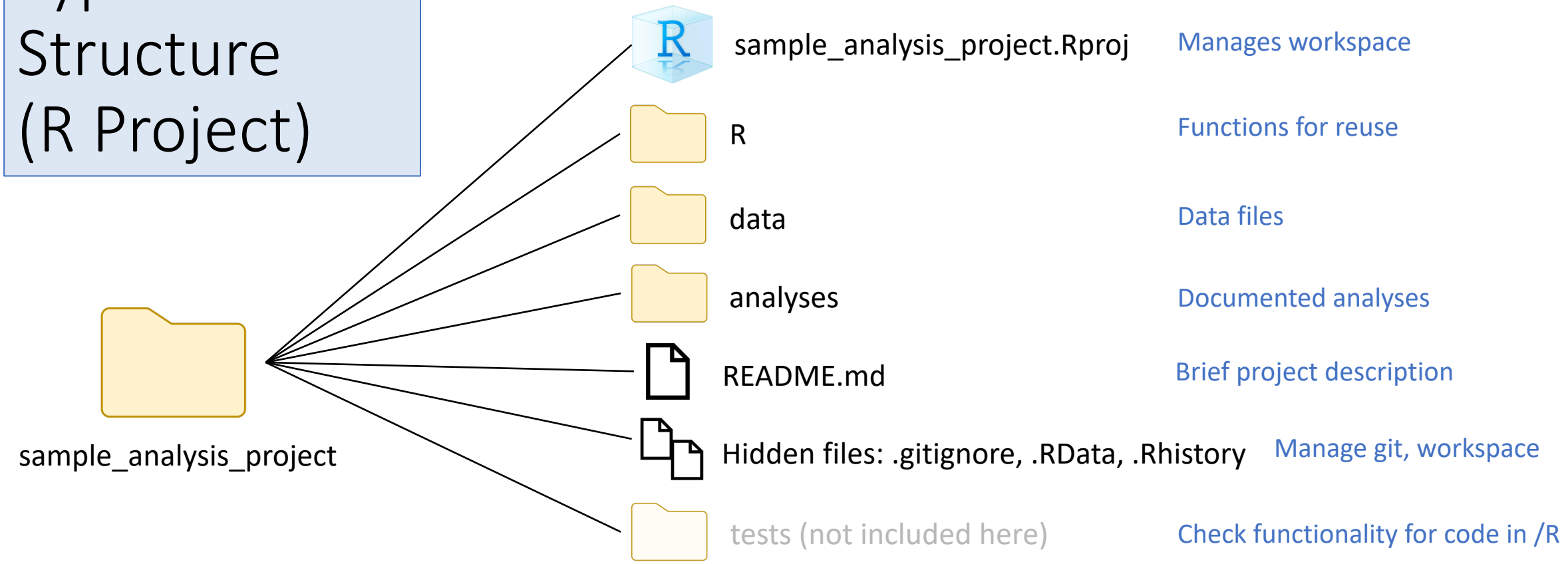
The R project (.Rproj) file

- The process on the previous slide created an R project (.Rproj) file.
- Whenever you open an R project, R starts a new session and reloads “what you were doing” when the project was last open.
  - data, history, loaded packages, open docs, etc.
- Closing one R project and opening another switches workspaces between research projects

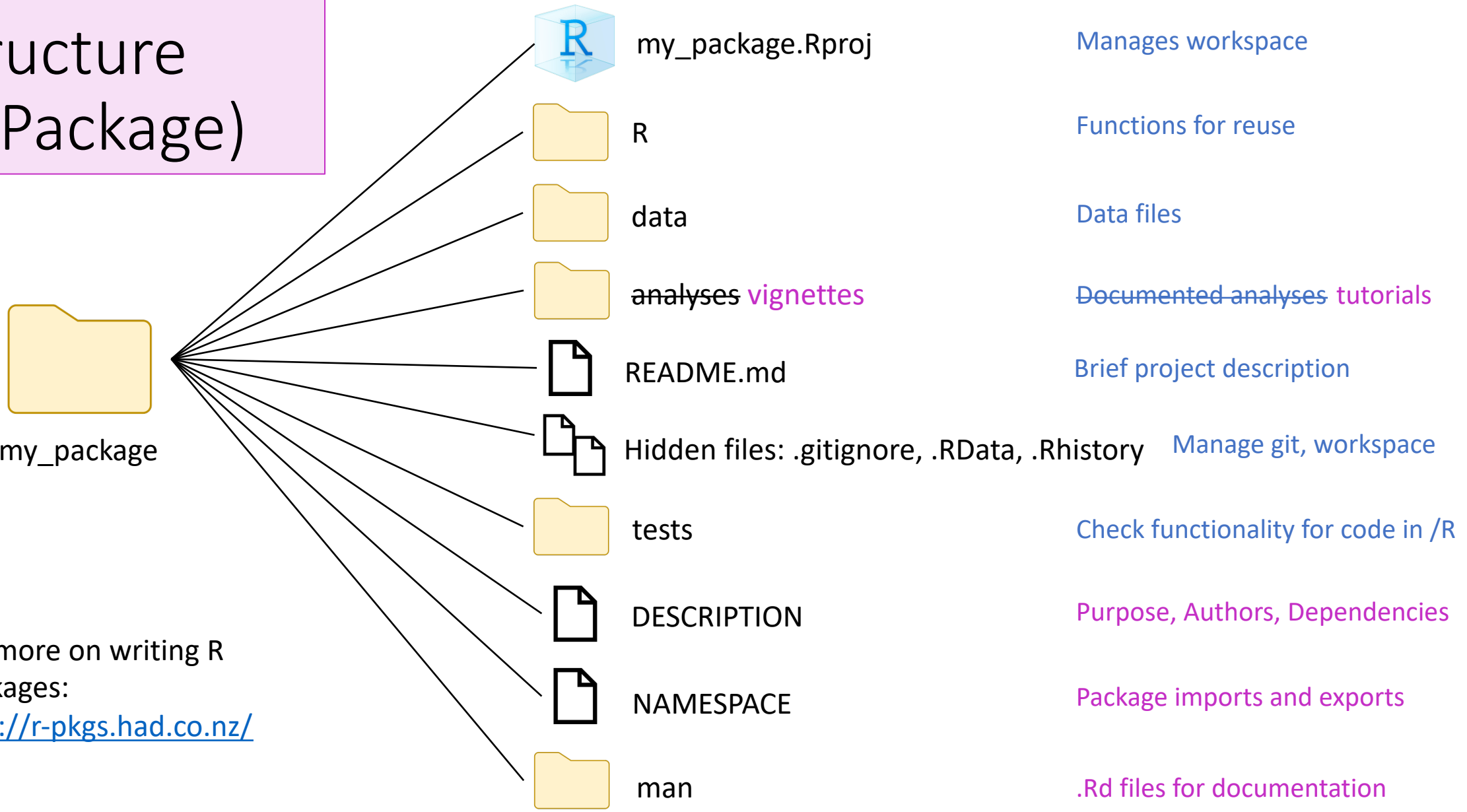
Click “File >> Open Project in New Session…” and open your new project



Typical Structure (R Project)



Typical Structure (R Package)



For more on writing R packages:
http://r-pkgs.had.co.nz/

analyses

- What goes here?
  - R Markdown (.Rmd) files describing downstream analysis.

R Markdown Documentation

- What is R Markdown?
  - File format which embeds text documentation (markdown and LaTeX) with code and code outputs (usually R, but can be python, shell, etc.).
  - Goal: Produce a computational artifact that others (and yourself!) can view, scrutinize, test, and run, to convince themselves that your ideas are valid.
  - Code and text are separated by a specific sequence of characters:

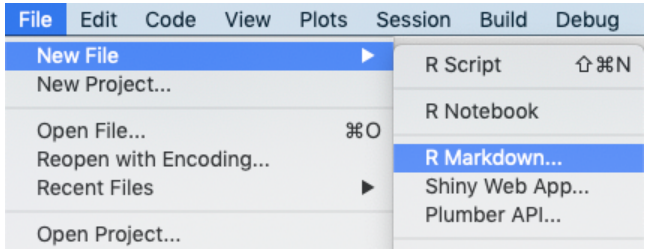
````
```{r}
# code goes here
```
````

R Markdown Documentation

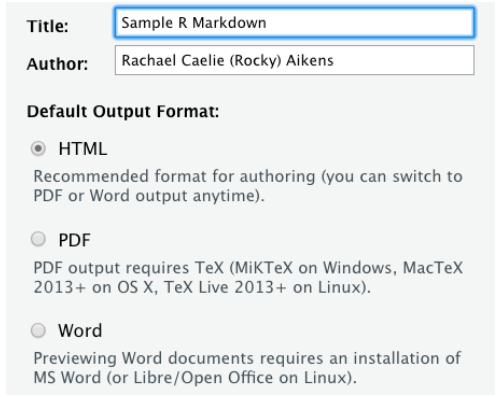
- I'll make an R markdown for
  - Lines of inquiry I'm still developing
  - Any document I want to pass to collaborators
  - Rendering all of my Main Figures for a paper in one place
  - Rendering my Supplementary Materials Document

Make a markdown

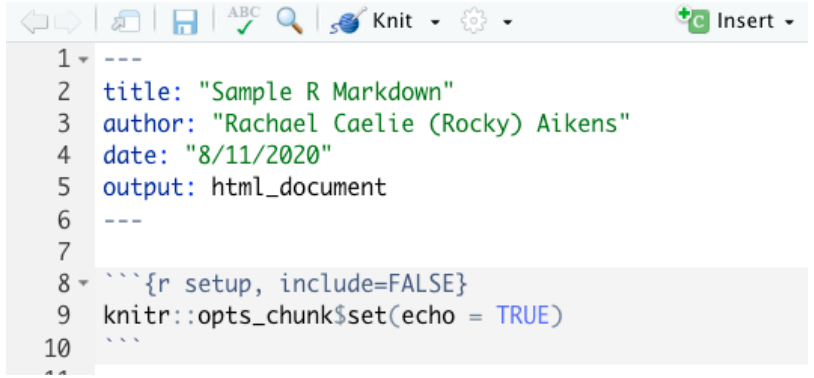
1. In Rstudio:
   Click “File >> New File >> R Markdown”



2. Give your document a title and select HTML for Default Output Format



3. RStudio will give you some starter code/text in your new R Markdown. Click “Knit” to see how that text compiles
   - (note: you’ll have to save your R markdown at this point)



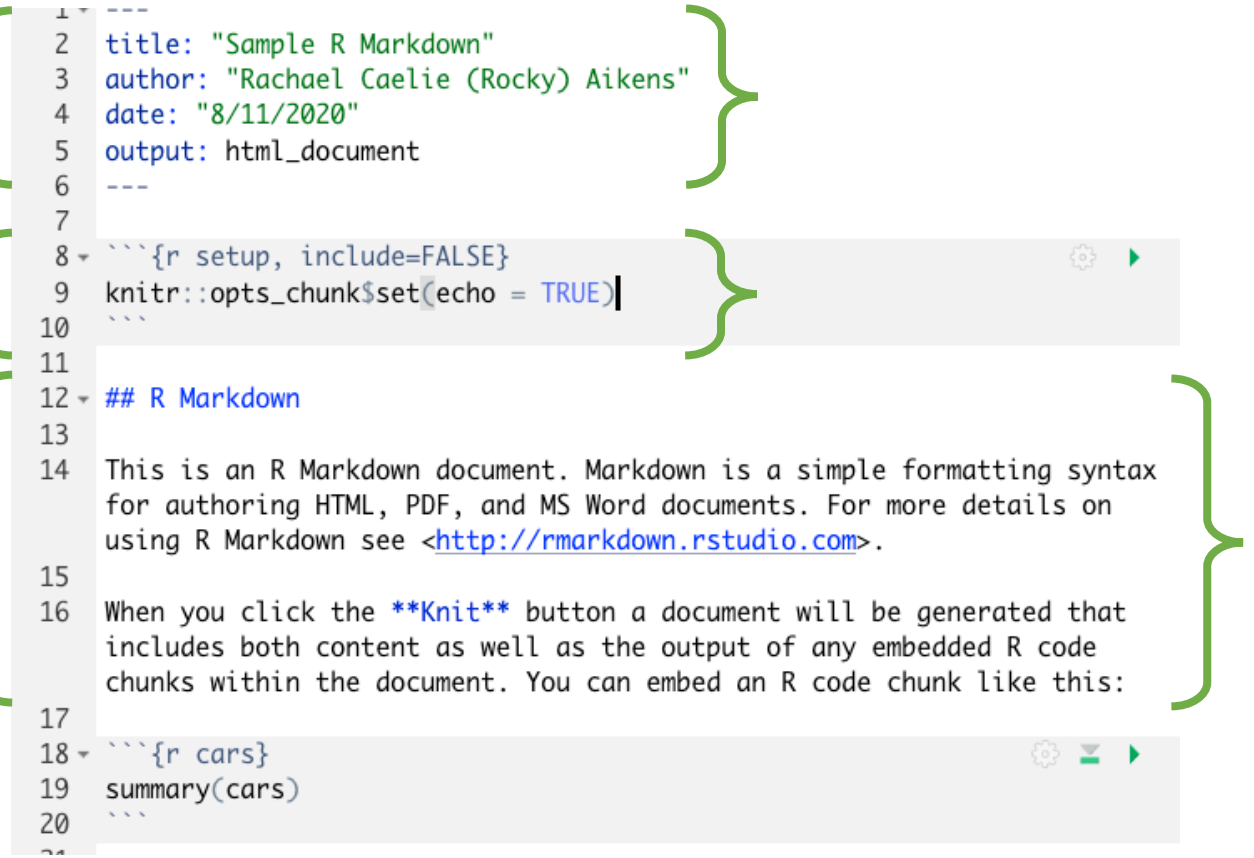
Anatomy of an R Markdown

YAML header specifies how document should be compiled

First code chunk is the setup chunk: specifies options for how code should be shown and run

- I also like to load packages here

Long-form documentation goes outside code chunks



Play with R Markdown

1. Make a new section by starting a text line with a #
2. Insert a new R chunk
   - Mac: Command-Option-I
   - Windows/Linux: Ctrl-Alt-I
3. Write some code that will produce an output
   - e.g. `` `hist(rnorm(100))` ``
4. Run the code by pressing the green arrow at the top right of each chunk
5. Click knit again to see the new product

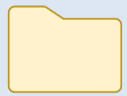
Already familiar with R markdown? Try these:

1. Check out what the gear icon at each chunk does
2. Navigate the document using the dropdown menu on the bottom left of your text pane 

3. Try File >> New File >> R Markdown and select Presentation or Shiny

More Markdown Resources

- Intro to R markdown: https://shiny.rstudio.com/articles/rmarkdown.html
- A very thorough reference guide to R Markdown: https://bookdown.org/yihui/rmarkdown/
- The R Markdown cheat sheet: https://shiny.rstudio.com/articles/rm-cheatsheet.html



R

- What goes here?
  - .R scripts with any functions that will be used multiple times

Code style

- Learning style conventions can go a long way to making your code easily readable by others
- Here is what the Venerable Master Hadley Wickham uses for R style: http://adv-r.had.co.nz/Style.html

- For python: https://www.python.org/dev/peps/pep-0008/

Code documentation with Roxygen

- Roxygen2 is a documentation system in R.
- Adhering to Roxygen2 formatting gives your documentation a standardized structure

```
#' Title
#'
#' Add a Description here
#'
#' @param n What is n?
#' @param prevalence What is prevalence?
#'
#' @return What is returned?
#' @export
#'
#' @examples
```

I usually delete these unless I'm writing an R package

R isn't the only language with standard documentation styles
Check out python docstring conventions:
https://www.datacamp.com/community/tutorials/docstrings-python

Document a function

1. In the files pane, open R/generate_data.R
2. Put your cursor somewhere in the body of `generate_cross_sectional`
3. Select code >> Insert Roxygen Skeleton
   - Mac: Alt + Cmd + Shift + R
   - Windows: Ctrl+Alt+Shift+R
4. By inspecting the code, write up a Title, description, and parameter definitions for this function

```
#' Title
#'
#' Add a Description here
#'
#' @param n What is n?
#' @param prevalence What is prevalence?
#'
#' @return What is returned?
#' @export
#'
#' @examples
```

I usually delete these unless I'm writing an R package

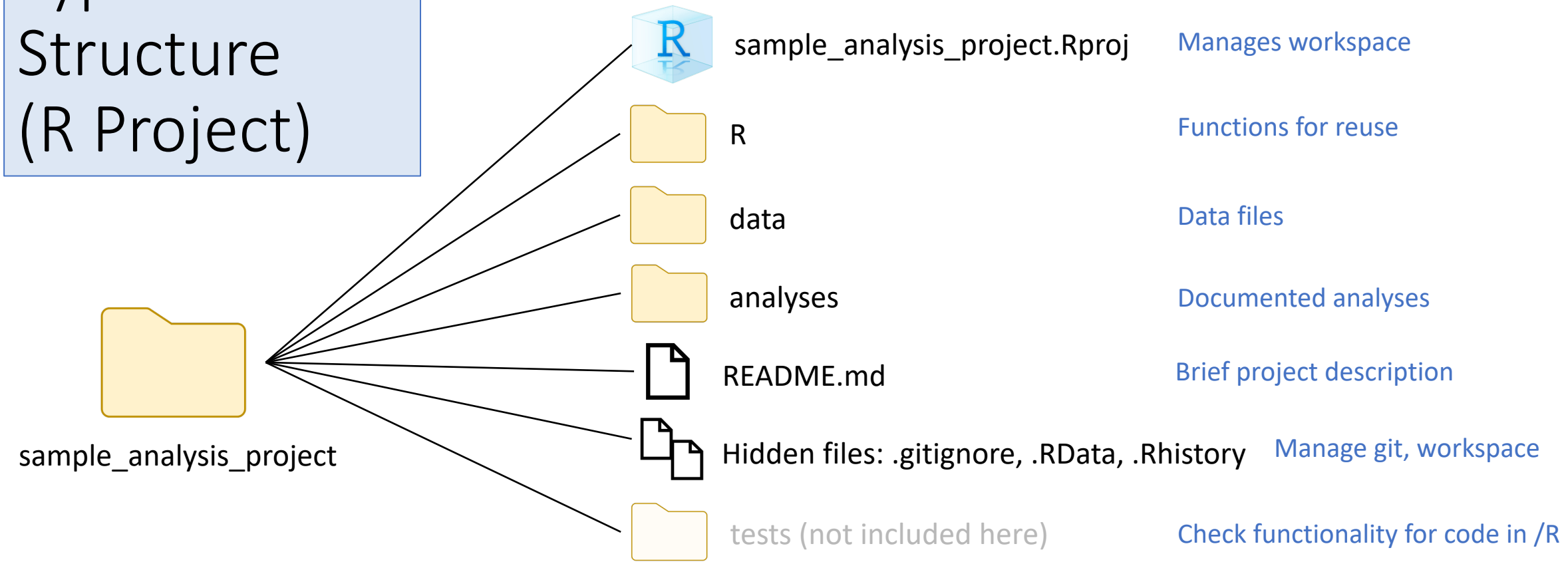
Document a function

- Try this: Write a description that is many characters long
  (maybe you already did)

```
#' I just cannot stop describing this it's crazy I could go on and on and on, but oh dear what will
happen when I type more than 80 characters and my formatting starts looking ugly?
```

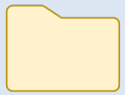
- Reflow Comment
  - Code >> Reflow Comment
  - Or, on Mac: Cmd+Shift+/
  - Or on Windows: Ctrl+Shift+/

Typical Structure (R Project)



data

- What goes here?
  - Data files
    - I use .csv, but it doesn’t really matter what format you use if it makes you happy



test

- What goes here?
  - Unit tests!
  - For more on testing in R see: https://testthat.r-lib.org/

Let’s make a quick analysis markdown

- **Research question:** Suppose that we are studying a disease (e.g. autism) in which a range of disease severities appear, and the more severe cases tend to be more reliably diagnosed.

  How does this diagnostic process distort the observed distribution of people with this disease?

To Do:

1. Articulate a basic model for diagnosis probability that increases with severity
2. Generate a simulated sample data set
3. Visualize diagnosed vs undiagnosed people in the simulated data.

Make an analysis Markdown

1. Open analyses/basic_modeling.Rmd
2. Knit it and read what's there
3. Complete the TODO items in each code chunk
   - Hint: the functions in the R/ folder will each be used
4. As you finish each chunk, edit the chunk header to read `` `eval = TRUE` `` ```` ```{r generate and diagnose data, eval = TRUE} ````
   - (There's nothing to do for the final chunk)
5. Knit the document

Revisiting the Setup Chunk

- In analyses/basic_modeling.Rmd, look at the setup chunk

Set global options for how code chunks are run/shown

Load code files and packages

Other global parameters
- ggplot theme
- Seed for random number generator

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, warning = FALSE,
                      message = FALSE, fig.align = "center")

source("../R/generate_data.R")
source("../R/diagnose_data.R")

library(tidyverse)

theme_set(theme_light())

set.seed(123)
n <- 50000
```
```

Edit the setup chunk to set `` `echo = FALSE` `` in the global chunk options. Then re-knit.

Quick Preview: R package structure

- Recently, I've started writing my project code directly as an R package. This has some benefits:
  - Easy to load entire codebase (Ctrl/Cmd + Shift + L)
  - Easy to compile/view roxygen2 documentation
  - Easy to implement automated testing
  - When I'm ready to share my code, people can download it directly to RStudio with:

```
devtools::install_github("raikens1/packagename")
```

More Resources

- Github
  - Using Git and Github in Rstudio: https://happygitwithr.com/
- R code
  - Hadley Wickham's style guide: http://adv-r.had.co.nz/Style.html
  - R for Data Science textbook: https://r4ds.had.co.nz/
  - Advanced R: http://adv-r.had.co.nz/
- R markdown
  - Intro to R markdown: https://shiny.rstudio.com/articles/rmarkdown.html
  - The R Markdown cheat sheet: https://shiny.rstudio.com/articles/rm-cheatsheet.html
  - A very thorough reference guide to R Markdown: https://bookdown.org/yihui/rmarkdown/
- Writing R packages: http://r-pkgs.had.co.nz/