Code Management for Reproducible Research

Allie Sherris, PhD Candidate, E-IPER
asherris@stanford.edu

SE3 SkillShare, February 24, 2021

pollev.com/asherris

Why should we care about code management?

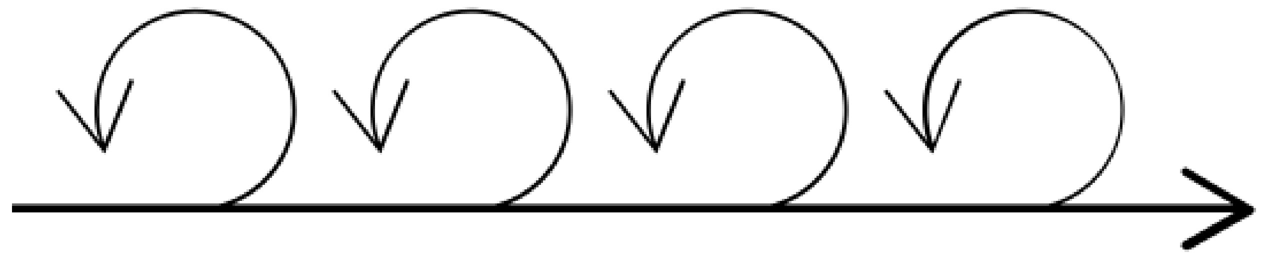
"OF COURSE YOU CAN'T REPLICATE MY EXPERIMENTS. THAT'S THE BEAUTY OF THEM."

Why should we care about code management?

"I am not disorganized — I know *exactly* where everything is! The newer stuff is on top and the older stuff is on the bottom."

Why should we care about code management?

Developing a code management style takes time and iteration



Sharon Showalter – Noun Project

Workshop goals

1. Give you tools and resources to develop your code management process
2. Share lessons learned from my own process** and other examples

Feel free to jump in with questions (or tips!) throughout

**Not necessarily the ideal!!

Outline

1

Writing legible code

2

Organizing reproducible projects

3

Going public

Not covered

- Version control (Git/Github)
- Coding (including tidyverse, “here” package)
- Dynamic reports (R Markdown/knitr/Sweave)
- Keeping a lab notebook / project log
- Publication norms

Takeaways (slides available at github.com/arsherris)

Writing legible code

- Don’t do anything by hand
- Develop a coding style
- Comments!!
- Short chunks and scripts
- Keep functions in separate scripts
- Start from scratch with each session
- Invest time in developing your style

Organizing projects

- Logically organize your project directory
- Have a “run project” button
- Have a setup/configuration script
- Keep raw and processed data separate
- Save output as a last resort

Going public

- Find the repositories of scholars you admire
- Match the manuscript methods to the code
- Share your code publicly
- Create a legible report / README
- Document your software environment

Resources (slides available at github.com/arsherris)

Writing legible code

R for Data Science (online book)

Tidyverse style guide

Google style guide

R Markdown (online book)

Stanford R Community

Organizing projects

Reproducible Research (JHU Coursera MOOC)

Reproducible Data Science (Harvard EdX MOOC)

Reproducible manuscripts with R Markdown

Jade Benjamin Chung's lab manual

Going public

Github

RPubs

Stanford Digital Repository

Report Writing for Data Science in R (free book)

Jade Benjamin Chung's Github

Antonio Gasparrini's Github

1. Writing legible code

An early script….

A cleaner script…

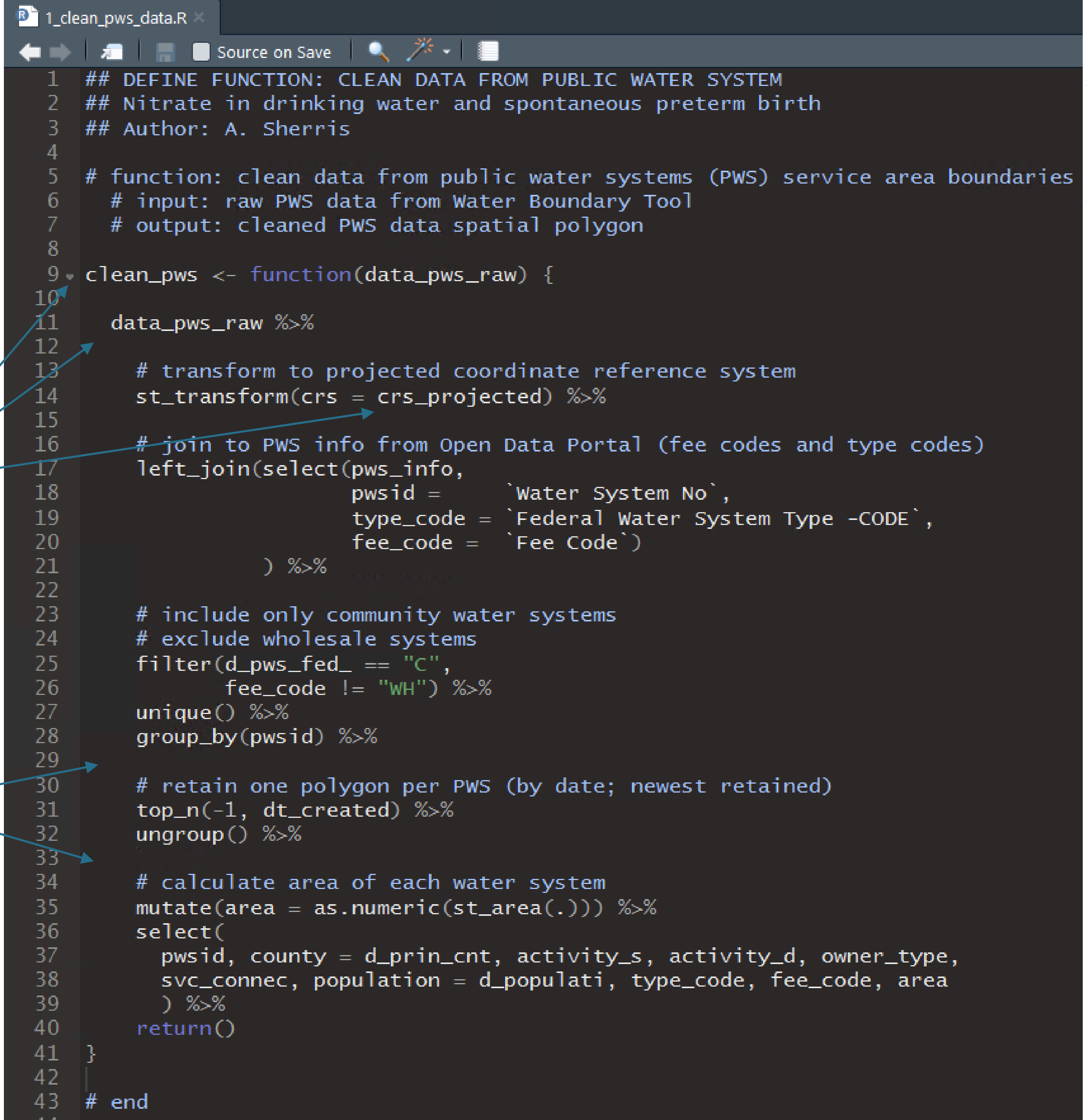
- https://github.com/arsherris

Coding style

consistent_naming

Spaces between commented lines

Note the end of script →



```r
## DEFINE FUNCTION: CLEAN DATA FROM PUBLIC WATER SYSTEM
## Nitrate in drinking water and spontaneous preterm birth
## Author: A. Sherris

# function: clean data from public water systems (PWS) service area boundaries
  # input: raw PWS data from Water Boundary Tool
  # output: cleaned PWS data spatial polygon

clean_pws <- function(data_pws_raw) {

  data_pws_raw %>%

    # transform to projected coordinate reference system
    st_transform(crs = crs_projected) %>%

    # join to PWS info from Open Data Portal (fee codes and type codes)
    left_join(select(pws_info,
                     pwsid =     `Water System No`,
                     type_code = `Federal Water System Type -CODE`,
                     fee_code =  `Fee Code`)
              ) %>%

    # include only community water systems
    # exclude wholesale systems
    filter(d_pws_fed_ == "C",
           fee_code != "WH") %>%
    unique() %>%
    group_by(pwsid) %>%

    # retain one polygon per PWS (by date; newest retained)
    top_n(-1, dt_created) %>%
    ungroup() %>%

    # calculate area of each water system
    mutate(area = as.numeric(st_area(.))) %>%
    select(
      pwsid, county = d_prin_cnt, activity_s, activity_d, owner_type,
      svc_connec, population = d_populati, type_code, fee_code, area
      ) %>%
    return()
}

# end
```

← Script name and author

← Define function purpose, input, and output

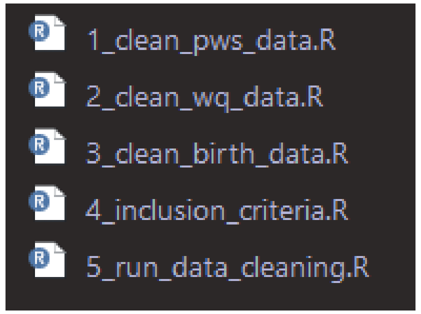
Comments for each separate action

Consistent spacing

← New line after each pipe (%>%)

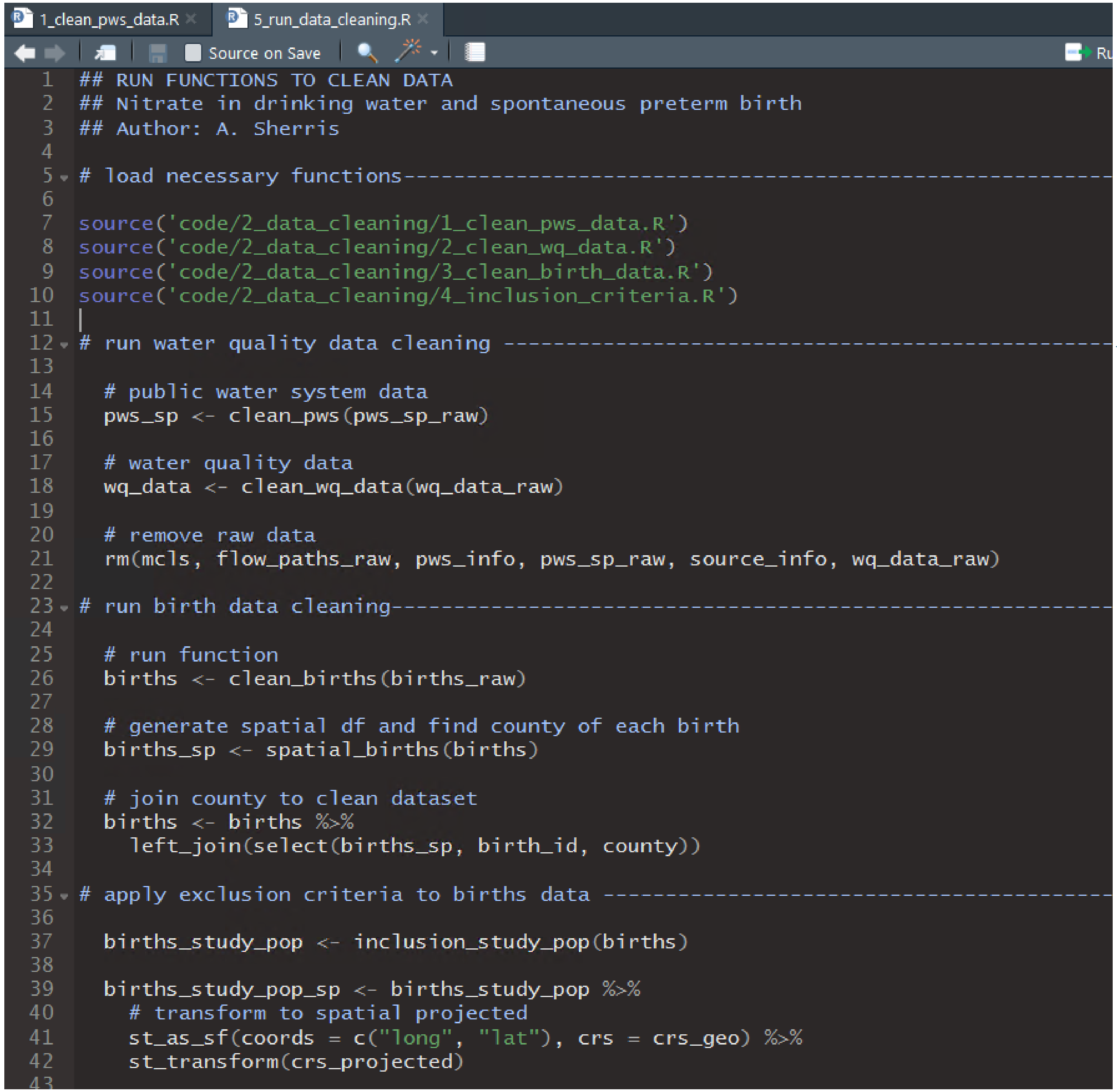
Short lines (<80 characters)

Shorter scripts (or chunks)



1_clean_pws_data.R
2_clean_wq_data.R
3_clean_birth_data.R
4_inclusion_criteria.R
5_run_data_cleaning.R

Functions are in separate scripts



```
## RUN FUNCTIONS TO CLEAN DATA
## Nitrate in drinking water and spontaneous preterm birth
## Author: A. Sherris

# load necessary functions-----------------------------------------------------------

source('code/2_data_cleaning/1_clean_pws_data.R')
source('code/2_data_cleaning/2_clean_wq_data.R')
source('code/2_data_cleaning/3_clean_birth_data.R')
source('code/2_data_cleaning/4_inclusion_criteria.R')

# run water quality data cleaning --------------------------------------------------

  # public water system data
  pws_sp <- clean_pws(pws_sp_raw)

  # water quality data
  wq_data <- clean_wq_data(wq_data_raw)

  # remove raw data
  rm(mcls, flow_paths_raw, pws_info, pws_sp_raw, source_info, wq_data_raw)

# run birth data cleaning-----------------------------------------------------------

  # run function
  births <- clean_births(births_raw)

  # generate spatial df and find county of each birth
  births_sp <- spatial_births(births)

  # join county to clean dataset
  births <- births %>%
    left_join(select(births_sp, birth_id, county))

# apply exclusion criteria to births data ------------------------------------------

  births_study_pop <- inclusion_study_pop(births)

  births_study_pop_sp <- births_study_pop %>%
    # transform to spatial projected
    st_as_sf(coords = c("long", "lat"), crs = crs_geo) %>%
    st_transform(crs_projected)
```

← Source functions

← Internal breaks (can be collapsed)

Style guides

- Tidyverse style guide: https://style.tidyverse.org/
- Google style guide: https://google.github.io/styleguide/Rguide.html
- R for Data Science (online book) : https://r4ds.had.co.nz/index.html

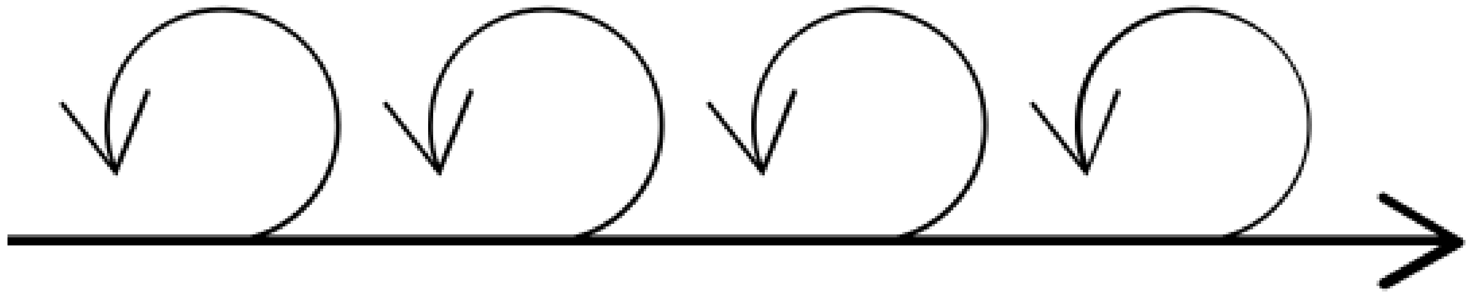
Crucial to reproducibility: Don’t do anything by hand!

For example:

- Loading data/functions by hand
- Fixing outliers/errors in raw data spreadsheets
- Copying/pasting coefficients from model output
- Doing anything in the console

Start from scratch with each session

Take time to develop your coding style!

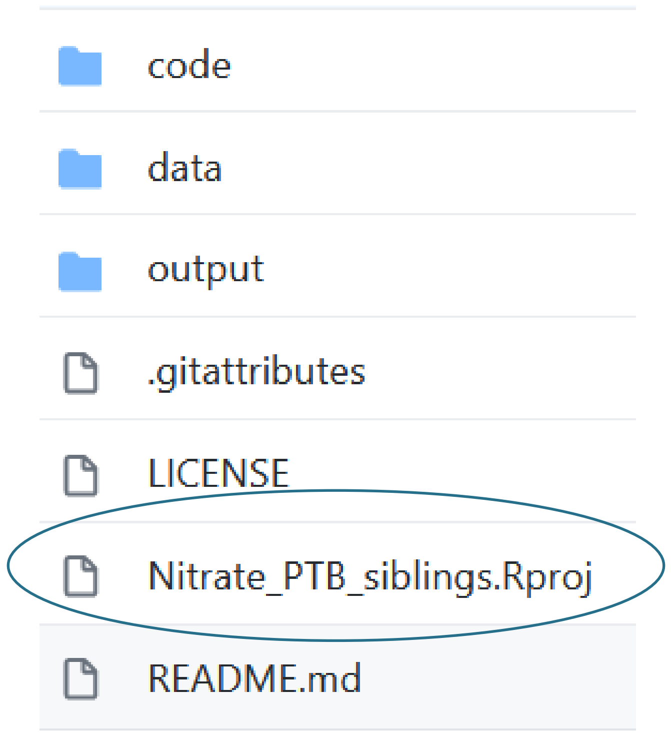


Takeaways: writing legible code

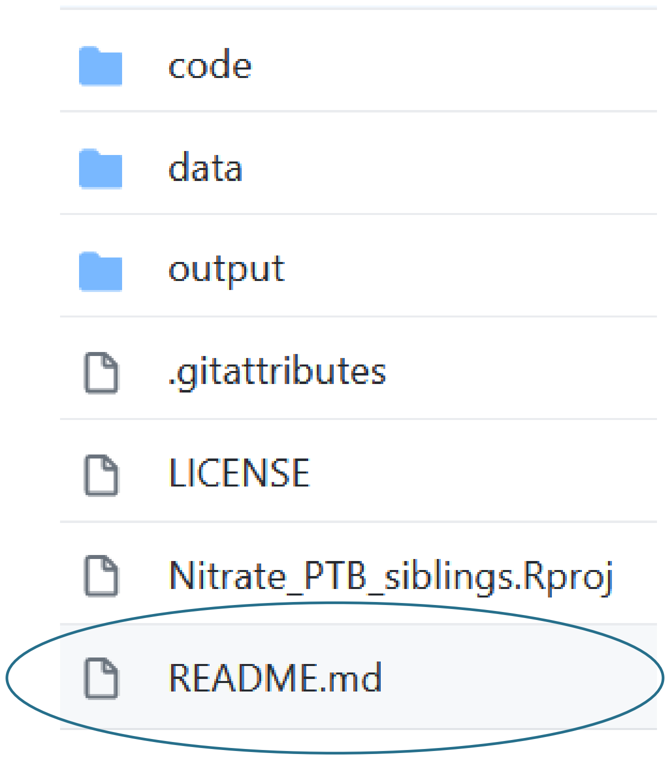
- Don’t do anything by hand
- Develop a coding style
  - Space out your code
  - Use consistent naming
  - Established style guides are very helpful
- Comments!
- Short chunks and scripts
- Keep functions in separate scripts
- Start from scratch with each session
- Invest time in developing your style

2. Organizing reproducible projects

Project structure

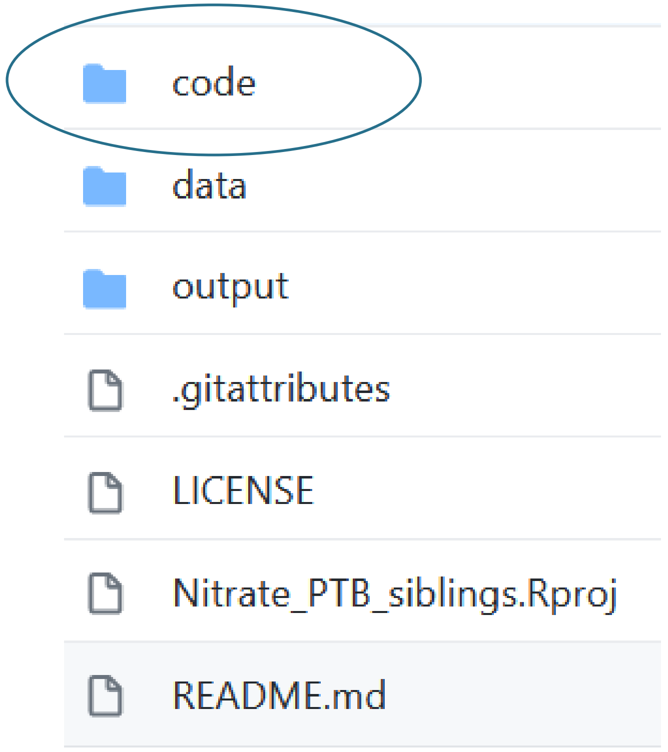

code
data
output
.gitattributes
LICENSE
Nitrate_PTB_siblings.Rproj
README.md

Project structure



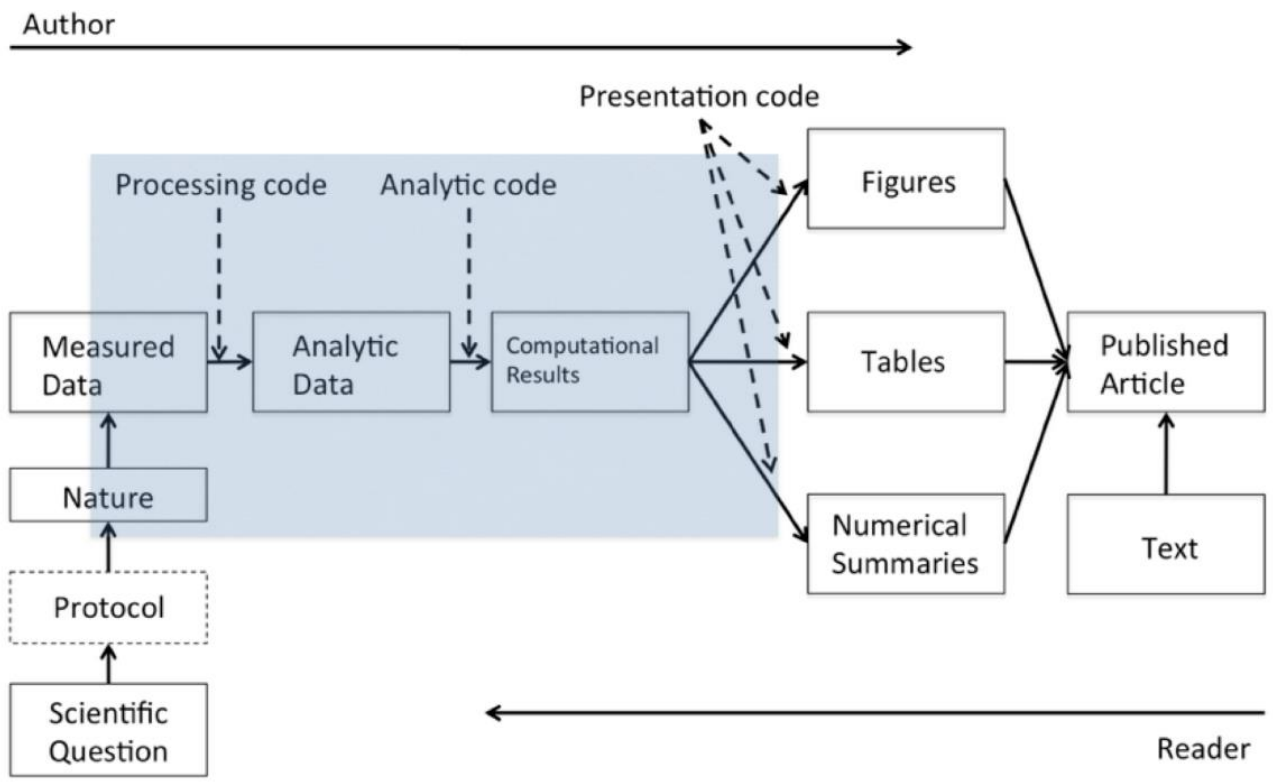
code
data
output
.gitattributes
LICENSE
Nitrate_PTB_siblings.Rproj
README.md

Project structure

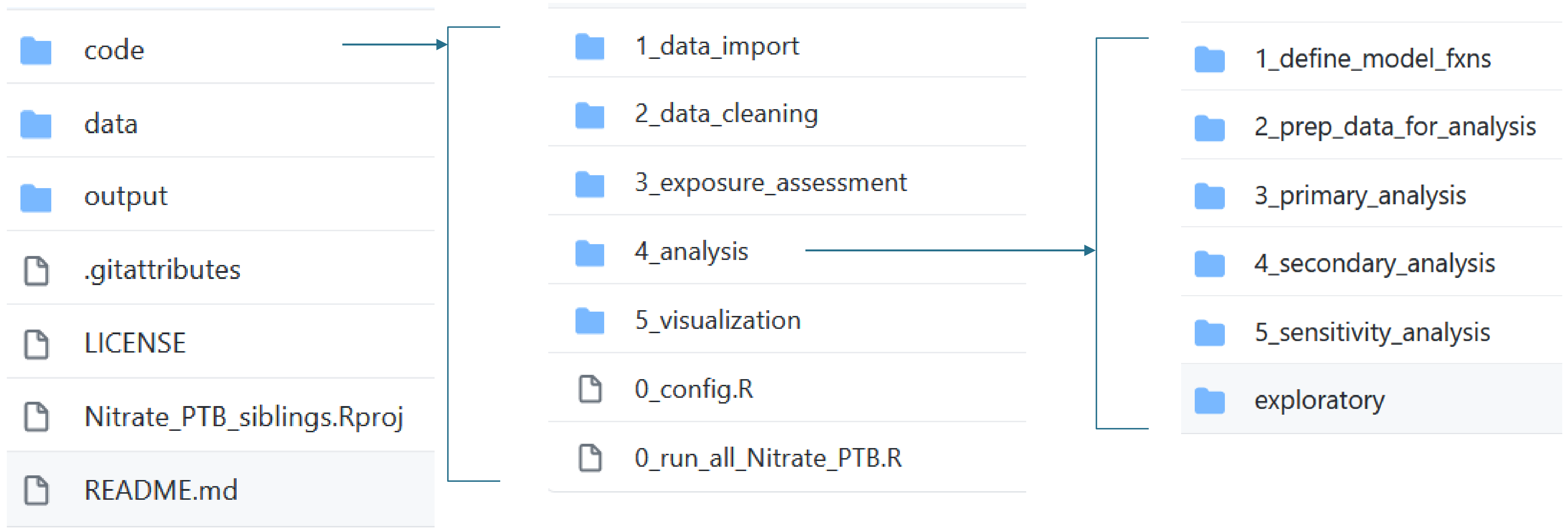


code
data
output
.gitattributes
LICENSE
Nitrate_PTB_siblings.Rproj
README.md

Code organization should reflect project methods



Code organization should reflect project methods

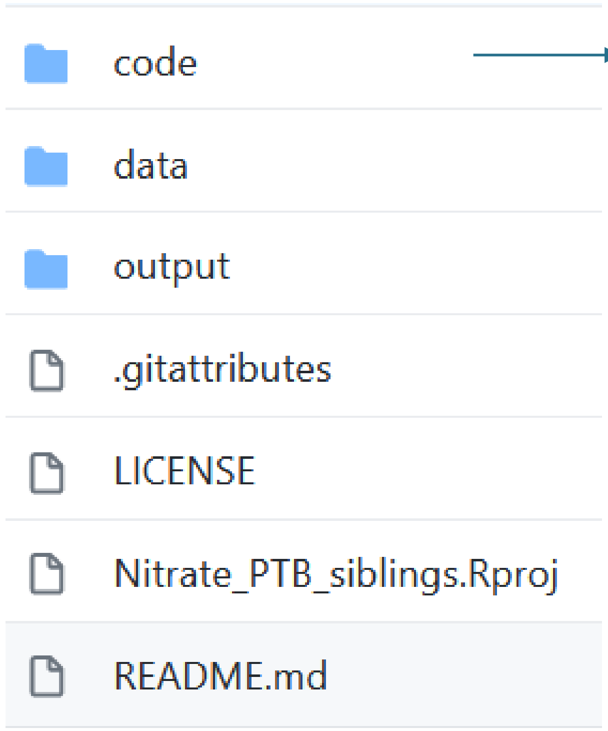


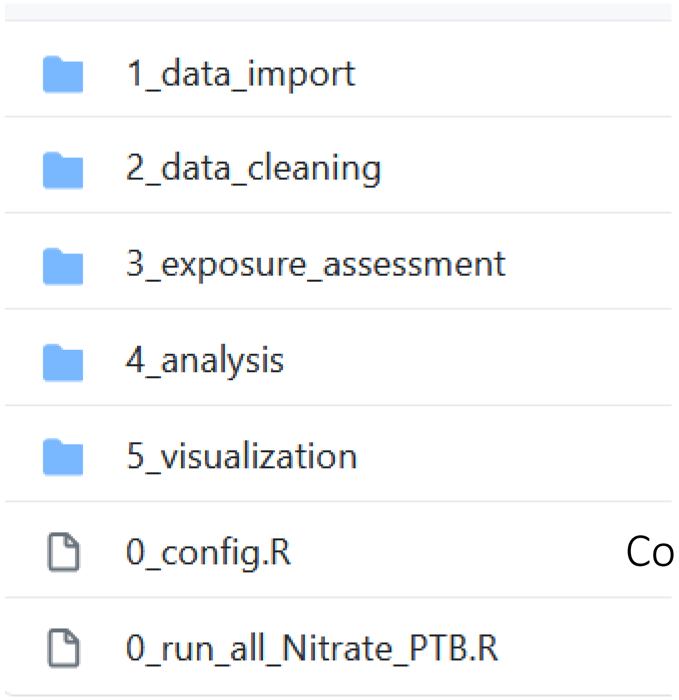
```
code
data
output
.gitattributes
LICENSE
Nitrate_PTB_siblings.Rproj
README.md
```

```
1_data_import
2_data_cleaning
3_exposure_assessment
4_analysis
5_visualization
0_config.R
0_run_all_Nitrate_PTB.R
```

```
1_define_model_fxns
2_prep_data_for_analysis
3_primary_analysis
4_secondary_analysis
5_sensitivity_analysis
exploratory
```

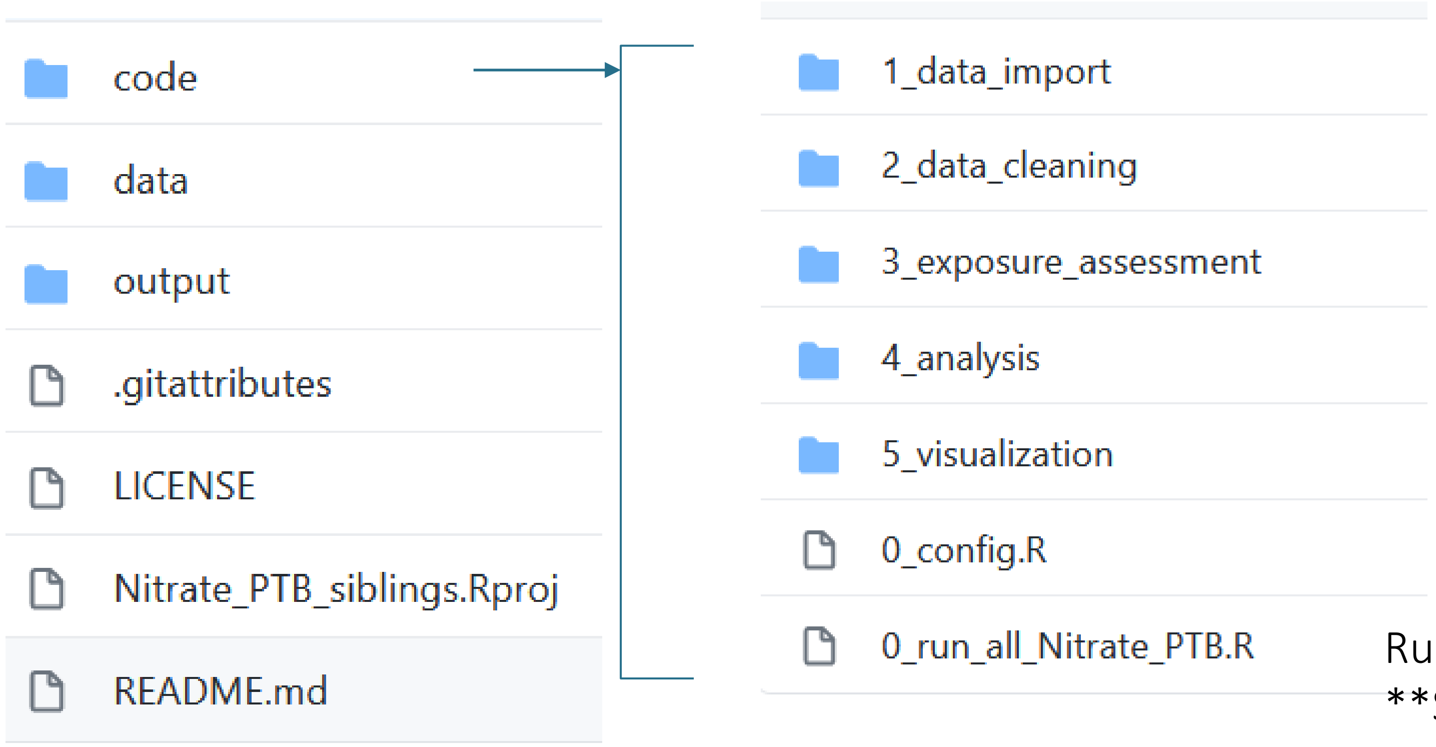
Have a “setup” or “configuration” file


code
data
output
.gitattributes
LICENSE
Nitrate_PTB_siblings.Rproj
README.md


1_data_import
2_data_cleaning
3_exposure_assessment
4_analysis
5_visualization
0_config.R
0_run_all_Nitrate_PTB.R

Configuration file: Sets up project workspace

Project should run with one click!

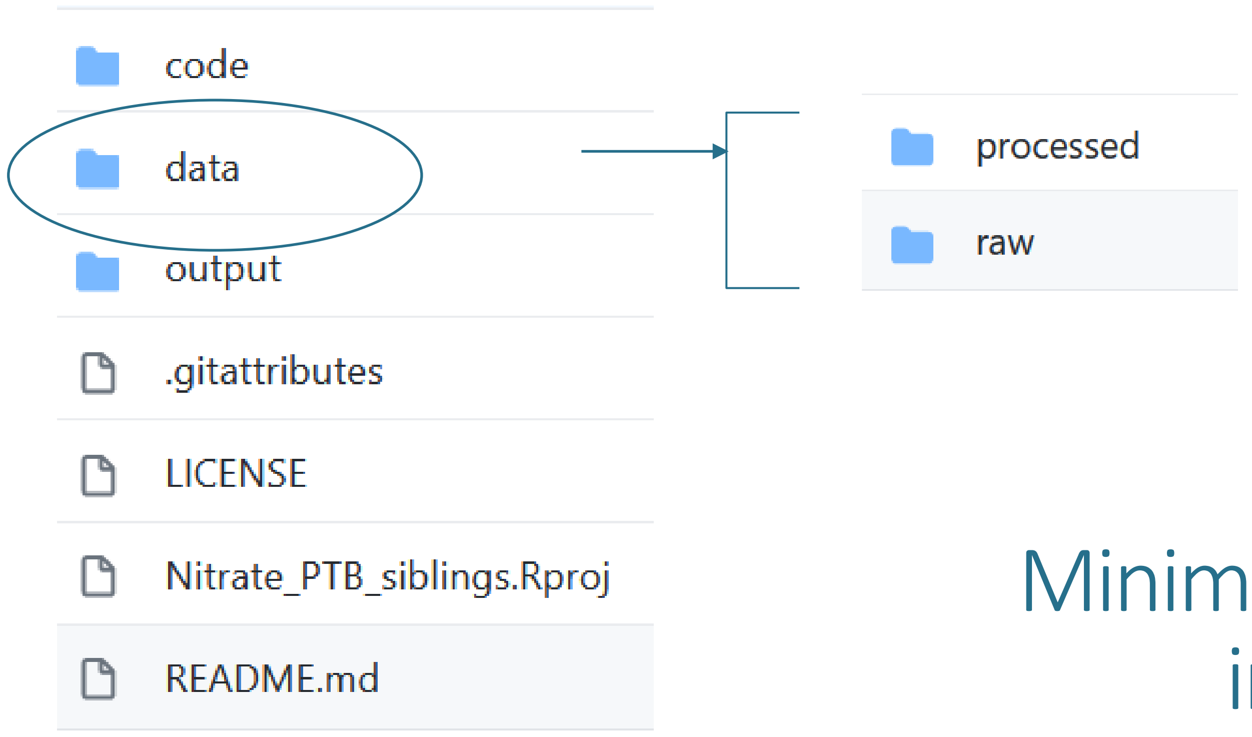


code
data
output
.gitattributes
LICENSE
Nitrate_PTB_siblings.Rproj
README.md

1_data_import
2_data_cleaning
3_exposure_assessment
4_analysis
5_visualization
0_config.R
0_run_all_Nitrate_PTB.R

Run project file: Executes everything!
**Should probably be a Bash script, which generates .Rout file with script output

Keep raw and processed data separate



Minimize what you need to save as interim/processed data
(Try to start from scratch with each session)

Relevant resources

- Reproducible Research (JHU / Coursera MOOC)
- Reproducible Data Science (Harvard / EdX MOOC)
- Jade Benjamin Chung's Lab manual: https://jadebc.github.io/lab-manual/

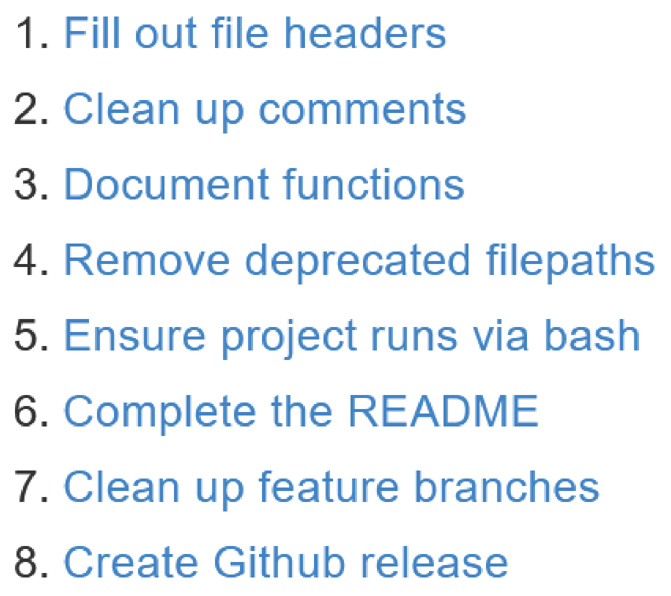
Takeaways: Organizing reproducible projects

- Logically organize your project directory to mirror the research process
- Have a setup/configuration file
- Have a “run project” button
- Keep raw and processed data separate
- Save output as a last resort

3. Going public

Tips for going public

Do some cleanup

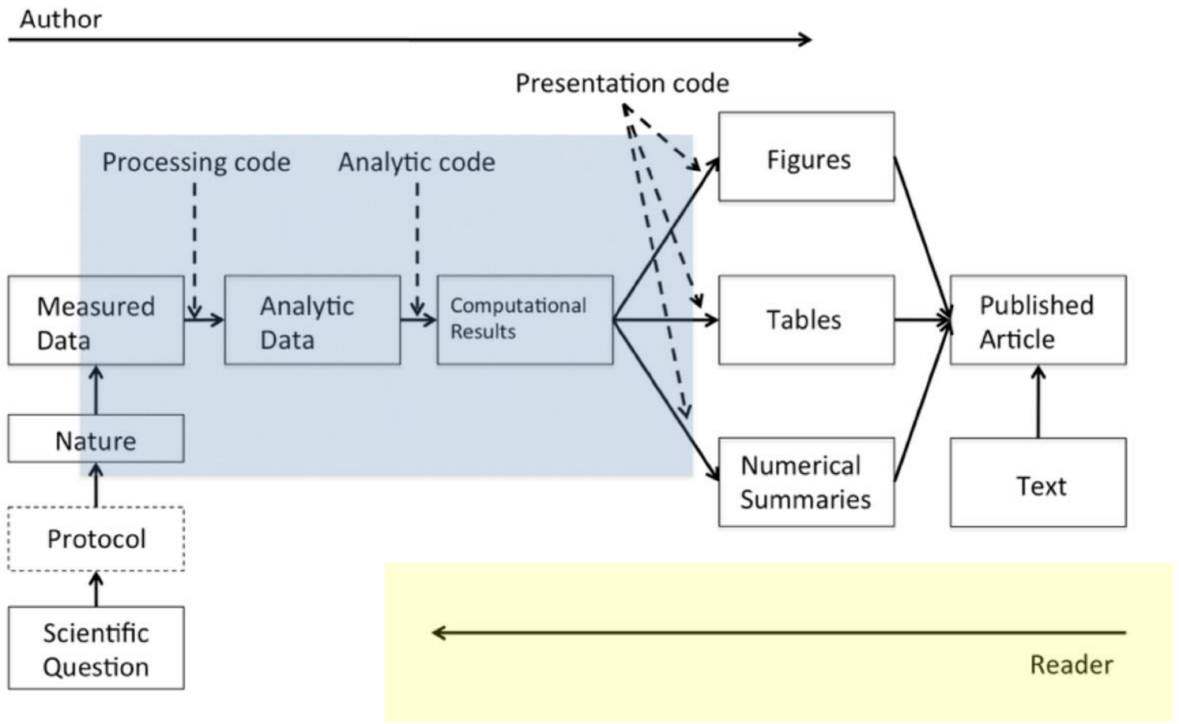


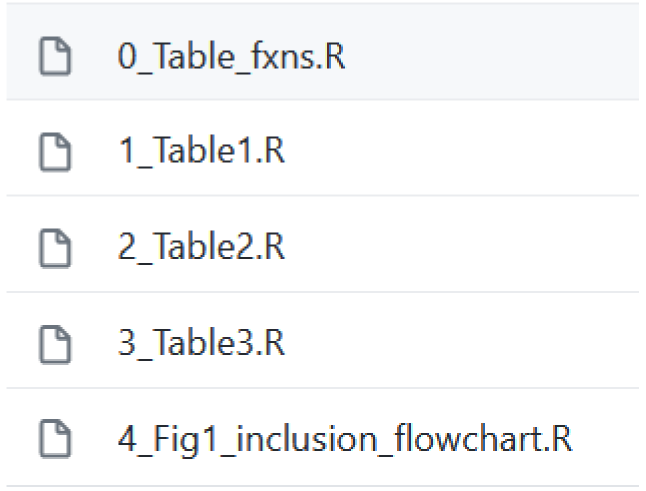
1. Fill out file headers
2. Clean up comments
3. Document functions
4. Remove deprecated filepaths
5. Ensure project runs via bash
6. Complete the README
7. Clean up feature branches
8. Create Github release

https://jadebc.github.io/lab-manual/code-publication.html#checklist-overview

Tips for going public

Make sure the code organization and output reflects the manuscript methods




- 0_Table_fxns.R
- 1_Table1.R
- 2_Table2.R
- 3_Table3.R
- 4_Fig1_inclusion_flowchart.R

Tips for going public

- Clear documentation and README
  - Describe project, data sources, directory structure, how to run code
  - Link to published manuscript, if applicable
- Not everything needs to go in
  - Remove exploratory or raw code
  - Often only clean data included
- Document your software environment!

Report-writing with R Markdown

Report Writing for Data Science in R, R Peng (free book)

https://github.com/rdpeng

Ideal for:

- Manuals
- Short/medium-length technical documents
- Tutorials
- Reports, especially if they will be generated periodically with updated data
- Data preprocessing documents and summaries

Not ideal for

- Very long research articles
- Documenting very complex and time-consuming computations
- Documents that require precise formatting

Find and emulate good examples from your discipline!

Takeaways: going public

Reporting and sharing

- Find and emulate good examples from your discipline
- Match the manuscript methods to the code
- Share your code publicly
- Create a legible report / README
- Document your software environment

Questions?