Putting Everything Together

Outline

- Organizing the analysis
- "From soup to nuts"
- Sharing
- Archiving
- A real example

Organizing the analysis

Directory structure

- Highly personal -- no single "correct" structure
- Also varies by project

Directory structure

- `data`
    - `data/original`
    - `data/processed`
- `analyses`
    - markdown files
    - scripts (but see below)
- `package` (optional)
    - for general stand-alone use, *and / or*
    - well-written helper routines, called by main analysis (`.md` files)
- `results`
    - fitted models, full simulation output, etc.
    - maybe `results/cached` + `results/final`
- `output`
    - plots, tables, etc., to include directly in paper
    - (sometimes) notebook output
- `app` (optional; e.g., shiny)

Makefiles

- `make` -- since 1976
- Created to provide the commands necessary to compile a software project
- Also great for organizing a reproducible analysis
- Organize commands into "rules" that build specific outputs
- Example

Makefile rules

- Like directory structure, personal and varies by project
- For nearly all projects:
    - `all`: runs everything, typically the default
    - `clean`: delete all output (including processed data, cached results)
    - `download`: (re-)download data from its original source
- Also common:
    - `preprocess`
      (or `clean_data`, or ...)
    - `main_analysis`
      (might simply call other rules, e.g., `eda`, `fitmodels`, etc.)
    - `output`: create plots, tables, etc.

Results caching

- Analysis should be divided into steps, with results saved after each step

Results caching

- Analysis should be divided into steps, with results saved after each step

- One "step" might be:
    - A digestible amount of code
    - A computationally intensive procedure
    - A logical breaking point
    - The general goal:

      If someone wonders, *how did they do XYZ?*
        - There is an "obvious" markdown file to look in
        - It is simple to read, quick to run, and (ideally) easy to edit

Results caching

- Analysis should be divided into steps, with results saved after each step

- One "step" might be:
  - A digestible amount of code
  - A computationally intensive procedure
  - A logical breaking point
  - The general goal:

    If someone wonders, *how did they do XYZ?*
    - There is an "obvious" markdown file to look in
    - It is simple to read, quick to run, and (ideally) easy to edit

- Each markdown file should clearly show its inputs and outputs

  Example

Results caching

- Analysis should be divided into steps, with results saved after each step

- One "step" might be:
    - A digestible amount of code
    - A computationally intensive procedure
    - A logical breaking point
    - The general goal:

      If someone wonders, *how did they do XYZ?*
        - There is an "obvious" markdown file to look in
        - It is simple to read, quick to run, and (ideally) easy to edit

- Each markdown file should clearly show its inputs and outputs

  Example

- Makefiles can also be used to specify file dependencies

”From soup to nuts”

One command

The user should be able to run your entire analysis -- everything -- with a single command.

One command

The user should be able to run your entire analysis -- everything -- with a single command.

For example:

- `make`
- `docker run myanalysis`
- `run_all.sh`

Data (soup)

- *The code should automatically (re-)download the data from its original source*

Data (soup)

- *The code should automatically (re-)download the data from its original source*

- **Don't:**

> *Download the data from www.somewhere.com. Go to "data" then select "stage 2" and download the .zip file.*
> *Unzip and follow the instructions in the README.*
> *After running the preprocessing script as described in the README, place the .csv files into the data directory and ...*

Data (soup)

- *The code should automatically (re-)download the data from its original source*

- **Don't:**

> *Download the data from www.somewhere.com. Go to "data" then select "stage 2" and download the .zip file.*
> *Unzip and follow the instructions in the README.*
> *After running the preprocessing script as described in the README, place the .csv files into the data directory and ...*

- **Do:**
  - Use `curl`, `wget`, etc.
  - Automate everything
  - Retain the original data, but keep it clearly separated from "cleaned" data

Final results (nuts)

"Final results":

- Plots
- Tables
- Numerical results (e.g., "$p = 0.02$")

Final results (nuts)

"Final results":

- Plots
- Tables
- Numerical results (e.g., "$p = 0.02$")

All final results should be output verbatim by your analysis

Final results (nuts)

- Numerical results (e.g., "$p = 0.02$")
    - Embed in a notebook
    - Should be easy to find

Final results (nuts)

- Numerical results (e.g., "$p = 0.02$")
  - Embed in a notebook
  - Should be easy to find

- Tables
  - `xtable`
  - `kable`
  - `stargazer`

Final results (nuts)

- Numerical results (e.g., "$p = 0.02$")
  - Embed in a notebook
  - Should be easy to find

- Tables
  - `xtable`
  - `kable`
  - `stargazer`

- Plots
  - `annotate()` (ggplot2)
  - `tikzDevice`

Final results (nuts)

- Numerical results (e.g., "$p = 0.02$")
  - Embed in a notebook
  - Should be easy to find

- Tables
  - `xtable`
  - `kable`
  - `stargazer`

- Plots
  - `annotate()` (ggplot2)
  - `tikzDevice`

- If you absolutely must edit by hand:
  - `diff` and `patch`

Sharing

Easily accessible

To make your analysis easily accessible,

- Post the code somewhere it is easy to browse (e.g., github)
- Post a fully self-contained docker image
- Post (or link to) the data somewhere you can download it directly
- Post any packages on CRAN, PyPi, etc.

Easily accessible

To make your analysis easily accessible,

- Post the code somewhere it is easy to browse (e.g., github)
- Post a fully self-contained docker image
- Post (or link to) the data somewhere you can download it directly
- Post any packages on CRAN, PyPi, etc.

- Keep the versions of your project in sync and leave a trail, e.g.,
    - Include your dockerfile on github
    - Have your docker image build directly from github
        - In the docker build, output a timestamp and the git commit that is used
- Cross-reference the websites

Archiving

Discussion

- Where might you archive your analysis (e.g., github)?
- What are strengths / weaknesses of those options?

Notable options

- Zenodo
- Open Science Foundation

Notable options

- Zenodo
- Open Science Foundation

Also, keep your own copy!

A real example

A real example

The paper: https://arxiv.org/abs/2105.03529

The code: https://github.com/adamSales/rebarLoop