Good Practice in Research Computing

A/Prof Mik Black
Department of Biochemistry
University of Otago

6 July 2021

ResBaz tidy projects session

Based on the papers:

- **Good enough practices in scientific computing**:
  Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK. PLoS Comput Biol. 2017 Jun 22;13(6):e1005510.
  doi: 10.1371/journal.pcbi.1005510

- **Ten Simple Rules for Reproducible Computational Research**
  Sandve GK, Nekrutenko A, Taylor J, Hovig E. PLoS Comput Biol. 2013 Oct;9(10):e1003285.
  doi: 10.1371/journal.pcbi.1003285

Overview

Sections (we'll cover the blue stuff):

- Data Management
- Software
- Collaboration
- Project Organization
- Keeping Track of Changes
- Manuscripts
- Ten Rules for Reproducible Computational Research

Reproducibility + Efficiency + Continuity

- For you:
  - recreate results (with ease)
  - reuse code and workflows
  - develop more efficient work habits
- For everyone:
  - ensure reproducibility (and reduce burden)
  - establish workflows
  - enable collaboration
  - build upon existing work

Quiz time…

Not really, just some thinking

Spend two minutes thinking about the answers to the following questions:

1. If you were abducted by aliens in the next 30 seconds, how easy would it be for someone else to carry on your research? (after we finished mourning your loss, of course).

2. How much time would you need to spend getting your work to the point where another human being *could* carry on with your project? What would this involve?

3. If you laptop/desktop was unexpectedly destroyed by evil anti-research forces, what impact would this have on your project? (it's ok, we'll buy you a shiny new one from our "anti-anti-research forces" insurance fund).

Lets (briefly) discuss your answers…

Overview

Sections:

- Data Management
- Software
- Collaboration
- Project Organization

Data management

1. Save the raw data.
   - preserve raw data - never edit
   - if needed, computationally create modified versions of raw data (e.g., data cleaning).
2. Ensure raw data are backed up in more than one location
   - NeSI + host institution (i.e., “remote” storage)
   - Data sensitivity issues with cloud-based storage providers
3. Create the data you wish to see in the world
   - non-proprietary formats
   - informative variable and file names
   - make it the data set you wished you had received…

Data management

4. Create analysis-friendly data
    - each column is a variable
    - each row is an observation
    - tidy data aren't just for R…
5. Record all the steps used to process data.
    - use scripts to capture (and automate) every step of data processing and cleaning

Data management

6. Anticipate the need to use multiple tables, and use a unique identifier for every record.
   - e.g., multiple CSV files linked by a common sample ID
   - combine data programmatically
7. Submit data to a reputable DOI-issuing repository so that others can access and cite it.
   - Zenodo: https://zenodo.org
   - Dryad: https://datadryad.org
   - FigShare: https://figshare.com

But first: CAN you share your data?

Software (scripts, code etc)

1. Place a brief explanatory comment at the start of every program.
    - explain what the code does, and how to use it.
    - if approriate, include example of usage.
2. Decompose programs into functions.
    - breaks tasks into small (well-documented) chunks
    - digestible => understandable
3. Be ruthless about eliminating duplication.
    - use functions
    - don’t copy and paste code chunks (write a function)

Software (scripts, code etc)

4. Always search for well-maintained software libraries that do what you need.
    - don’t reinvent the wheel…
5. Test libraries before relying on them.
6. Give functions and variables meaningful names.
    - use tab completion with informative names
7. Make dependencies and requirements explicit.
    - can be as simple as adding a `requirements.txt` file.

Software (scripts, code etc)

8. Do not comment and uncomment sections of code to control a program's behavior.
    - error prone
    - use `if/else` statements instead
9. Provide a simple example or test data set.
    - also good for you to help with testing
    - and great for running workshops!
10. Submit code to a reputable DOI-issuing repository.

Even if you are not working as part of a team…

Even if you are not working as part of a team…

…YOU ARE!!

Collaboration

- Your project is very likely to be part of a larger research programme, and the work you are doing has the potential to contribute to other projects in the future.
- You are also collaborating with yourself: ask “present Murray” about his good friend “future Murray”.

Collaboration

1. Create an overview of your project.
   - `README.txt` or `README.md` in top-level directory
2. Create a shared “to-do” list for the project.
   - even if it is just for you…
3. Decide on communication strategies.
4. Make the license explicit.
5. Make the project citable.

Quiz Time...

Whose project folder looks like this?

Find a partner…

- Spend two minutes discussing how you organise your projects.
- Think about:
  - directory structure
  - documentation
  - where different types of files are kept

This stuff is critical…

Project Organisation

1. Put each project in its own directory, which is named after the project.
2. Put text documents associated with the project in the `doc` directory.
3. Put raw data and metadata in a data directory and files generated during cleanup and analysis in a results directory.
    - critical to separate input data from derived data
    - don’t be afraid to use subdirectories to impose additional order

Project Organisation

4. Put project source code in the src directory.

   - Interpreted: R, Python
   - Compiled: C++, Fortran, Java
   - Shell scripts, SQL snippets

   For small projects, include a “runall” script. For example:

```
TEMP_DIR = ./temp_zip_files
echo "Packaging zip files required by analysis tool…"
mkdir $(TEMP_DIR)
./src/make-zip-files.py $(TEMP_DIR) *.dat
echo "Analyzing…"
./bin/sqr_mean_analyze -i $(TEMP_DIR) -b "temp"
echo "Cleaning up…"
rm -rf $(TEMP_DIR)
```

Project Organisation

5. Put external scripts or compiled programs in the bin directory.
    - may not be relevant for some projects (e.g., if not using compiled code)
    - why external scripts here? Can make distinction for edited vs non-edited files: binary files and external scripts are not directly edited.
6. Name all files to reflect their content or function.
    - Please do this.

Project Organisation

5. Put external scripts or compiled programs in the bin directory.
   - may not be relevant for some projects (e.g., if not using compiled code)
   - why external scripts here? Can make distinction for edited vs non-edited files: binary files and external scripts are not directly edited.
6. Name all files to reflect their content or function.
   - Please do this
   - **PLEASE!!**

Example: project layout

```
|-- CITATION
|-- README
|-- LICENSE
|-- requirements.txt
|-- data
|    |-- birds_count_table.csv
|-- doc
|    |-- notebook.md
|    |-- manuscript.md
|    |-- changelog.txt
|-- results
|    |-- summarized_results.csv
|-- src
|    |-- sightings_analysis.py
|    |-- runall.py
```

Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK. PLoS Comput Biol. 2017 Jun 22;13(6):e1005510.

10.1371/journal.pcbi.1005510

Closing thoughts…

Tools and habits

- Tools **enable** reproducible research
  - there are lots of them
  - they are not hard to learn
- Good habits **ensure** reproducible research
  - there are many good habits: each one will improve your work in some way
  - these **are** hard to learn (actually, that is not true - it is just hard to **unlearn** your bad habits)
- Getting into the habit of using these “good practice tools” as a central part of your workflow is critical
- It takes commitment, but it will be hugely beneficial

THANK YOU…

THANK YOU…

…time for an exercise!