Day 10

Effective data analysis, Reproducibility, Roundup

- Critically reading papers
- Organizing & managing a data analysis project
- Recap of main ideas & themes
- Some general thoughts

Critically reading literature

- Reading primary research articles
- Reading papers that propose methods & software
- Reading, retention, and reuse
- More than papers

Reading primary research papers

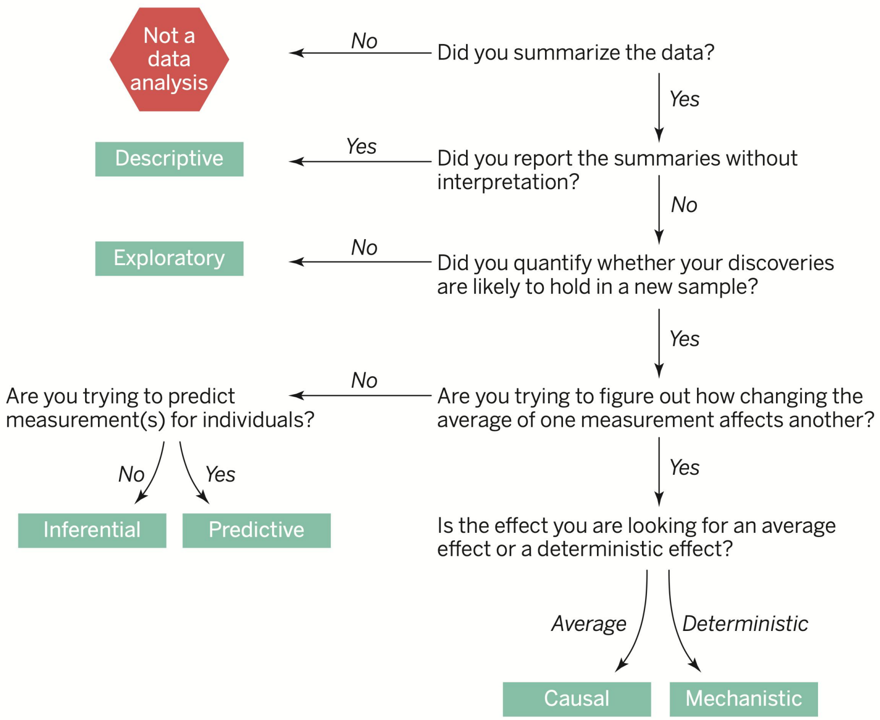
1. Use **Title & Abstract** for only selecting paper.

a. Don't be swayed by high-profile papers, media hype, or current dogma.

2. Read the **Introduction**:

a. Identify *the* question. What is the big challenge the authors are trying to solve?

b. What are the *specific* questions this paper is going to answered?



Leek, Peng (2015) 10.1126/science.aaa6146

Reading primary research papers

3. Read **Data & Methods**: [Be critical!]

a. For each specific Q, note data (type & source) & method (algorithms/techniques, software, & approach).

b. Are the data & methods describes sufficient to answer the Qs raised in the Intro?

c. Make <u>detailed notes</u> on: 1) what's unclear, 2) what you might do differently.

4. ALWAYS read the **Supplementary Materials**

These days much of the good stuff is in here!

Reading primary research papers

5. Read the **Results**: [Be critical!]

a. Go figure-by-figure, panel-by-panel. Based on your reading of Data & Methods, is there enough information to know/reproduce that analysis?

b. Try to interpret each figure/panel, then read the figure legend and the part of the results that explains it. [**Supplemental figures/tables** abound!]

   i. Do your interpretations match that of the authors'?

   ii. Are the results answering the specific Qs?

c. Make <u>detailed notes</u> on: 1) what's unclear, 2) what you might do differently.

6. Read the **Discussion/Conclusions**, **Title**, & **Abstract**:

a. Step back to think about contributions, limitations, open Qs, & next steps.

7. Read what other researchers (**papers that cite this paper**) say about this paper.

Methods & Software

Read software/methods papers

- Use Google Scholar to find recent application papers that use the software/method & read those.
- Search and read blogs and watch YouTube videos.
- Together, these will not only help you understand the methods but also key **assumptions and parameters** that you need to think about for your project.

- Don't use software/code without understanding it.
- Don't blindly adopt any technique without putting it into the context of your project and your capabilities.

Methods & Software

Explore the actual software/code

- Read the documentation: Overview and parts of it that correspond to the assumptions & parameters relevant to your project.
- Look into the exact data **input & output formats**.
- After installation, **replicate an example** run exactly as-is from the documentation/website or from an independent online tutorial.
  - If neither is available, email the (first & corresponding) authors asking for example data & detailed instructions on how to run their code.
- Online user groups and stack overflow are your friends.

- Don't use software/code without understanding it.
- Don't blindly adopt any technique without putting it into the context of your project and your capabilities.

You and your learning – Reading, Retention, and Reuse

- Be critical. Don't be swayed by high-profile papers, media hype, or current dogma.
- Don't Repeat Yourself: Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
    - Use a reference manager (e.g. Zotero), put *everything* you read into it. Use tags to group papers by subfield/method/data.
    - Create and maintain a single (R/Jupyter Notebook; Google Doc; Evernote) with notes/text-excerpts/figures from all papers & reading materials. Add notes about each paper / dataset / method.
- Create and maintain a single source of all the technical terms and vocabulary for your project.
- Contextualize what you read in relation to everything else you know / have read. Specifically consider limitations. Analyze information in terms of you and your project.

Do not limit yourselves to papers and textbooks

Online blogs/tutorials/talks/lectures

- Available at ***all*** levels of expertise
- Can be tastefully paired with primary research articles
- Cover many aspects of science absent in primary literature, including things not to do.

Great way to learn:

- Practical aspects of many theoretical ideas
- Visually, via demonstrations, plots, animations, videos

Organizing & managing a data analysis project

- Organizing a project
- Managing data and code
- Version control
- Programming lang. & software ecosystems
- Getting help

Organizing a data analysis project

`project_directory`

- `data`
  - primary & processed data + `readme.txt` + `runlog.sh`
- `src`
  - all your code/scripts
- `bin`
  - all compiled code + installed binaries + `readme.txt`
- `doc`
  - literature notes + analysis notes + intermediate/final report
- `results`
  - YYYY-MM-DD sub_directories
    - `runlog.sh` + R/Python notebooks

Based on Noble (2006) PLoS Comp. Biol.

Organizing a data analysis project

`project_directory`

- `data`
  - primary & processed data + `readme.txt` + `runlog.sh`
- `src`
  - all your code/scripts
- `bin`
  - all compiled code + installed binaries + `readme.txt`
- `doc`
  - literature notes + analysis notes + intermediate/final report
- `results`
  - YYYY-MM-DD sub_directories
    - `runlog.sh` + R/Python notebooks

No manual editing of data; Write scripts

Details on when & where data was downloaded

No code in this dir; Should point to & run code from `src`; this file should have all the command-lines used to run the code/scripts to process data here

Including those used for data download, processing, and analysis; Well documented with detailed comments within the code + external documentation.

Details on when and from where external software was downloaded; also include installation instructions if it was not straightforward.

Based on Noble (2006) PLoS Comp. Biol.

Organizing a data analysis project

`project_directory`

- `data`
  - primary & processed data + `readme.txt` + `runlog.sh`
- `src`
  - all your code/scripts
- `bin`
  - all compiled code + installed binaries + `readme.txt`
- `doc`
  - literature notes + analysis notes + intermediate/final report dir
- `results`
  - YYYY-MM-DD sub_directories
    - `runlog.sh` + R/Python notebooks

One file named with YYYY-MM-DD date of each analysis; Should contain free-text details on the thoughts/ideas behind that day's analyses.

Used at the later stages of a project to pull all the results into a report/paper.

At each stage of an analysis, gather your results (as text files) & make plots to visualize & interpret.

Should point to & run code from `src`; This file should have all the command-lines used to run the code/scripts to produce the results here.

Based on Noble (2006) PLoS Comp. Biol.

Managing data and code

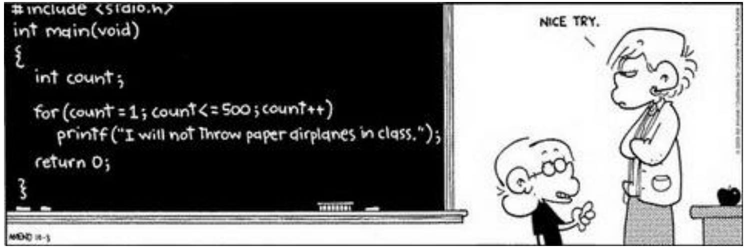
Data

- Give all files meaningful, interpretable, & computable names
    - Machine readable, human readable, works well with default ordering.
- Do not tamper with original/source files
    - `readme.txt` should contain detailed information about when & from where each piece of data was obtained.
- Do not make changes by hand; Automate everything
    - Write scripts that read in the file and generates the desired file.
- Document everything
    - Keep track of all your commands (Linux & running code) in a `runlog.sh`.

Examples of bad vs. good filenames

BAD	BETTER
`01.R`	`01_download-data.R`
`abc.R`	`02_clean-data_functions.R`
`fig1.png`	`fig1_scatterplot-bodymass-v-brainmass.png`
`IUCN's metadata.txt`	`2016-12-01_IUCN-reptile_shapefile_metadata.txt`

https://speakerdeck.com/jennybc/how-to-name-files

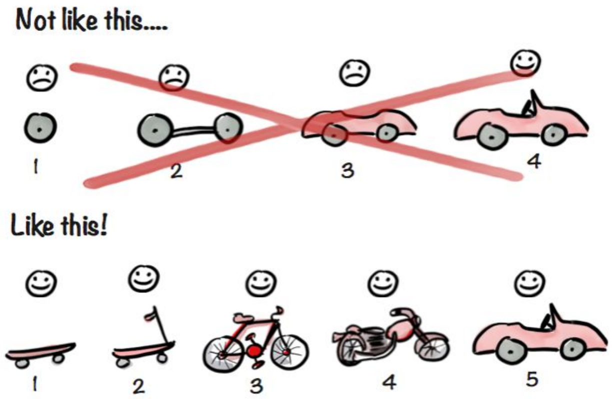


Managing data and code

Code

- Write code for both computers & humans.
  - Give descriptive, interpretable variable & function names.
  - Comment your code at the top: purpose, expected usage, example inputs/outputs, dependencies.
  - Record imports, constants, random seeds at the top.
  - Comment each block/function: the intended computation, arguments, return values.
- Program for the general case, and put the specifics outside the code as arguments & parameters.
- Eliminate effects between unrelated things.

Continuously functional & testable



Spotify

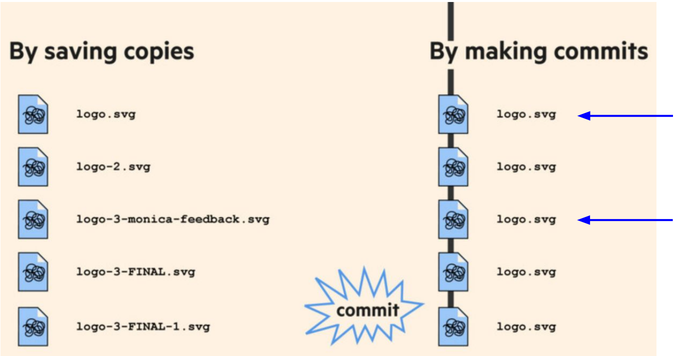
Reusing existing code:

- Begin by adding detailed comments.
- Properly acknowledge code borrowed from elsewhere; Check license.

Managing data and code

Version control

- Storify your project
- Travel back in time
- Experiment with changes
- Backup your work
- Collaborate effectively



Arjun Krishnan
12:34pm January 3th 2018

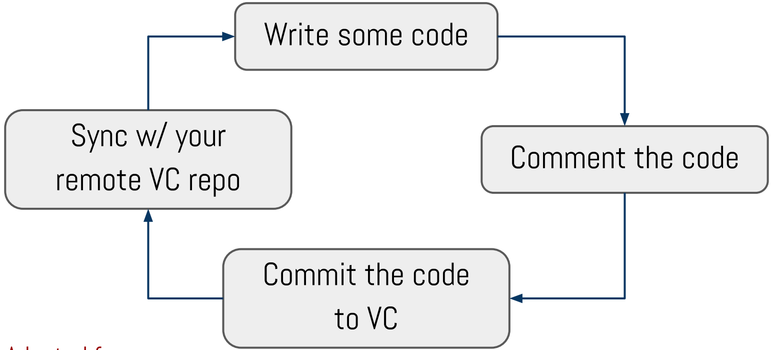
Updated background color

Changed background color to improve contrast.

Arjun Krishnan
9:15am January 4th 2018

Incorporated feedback from team

Made all changes based on team.org/feedback314



Adapted from @alicebartlett

repository	Your project folder
commit	A snapshot of your repo
remote	A computer with the repository on it
clone	Get the repository from the remote for the first time
push	Send commits to a remote
pull	Get commits from a remote
merge	Combine two branches

Programming languages & software ecosystems

	R	Python
Language, IDE, Notebook	R | RStudio | R Notebook	Python | Rodeo | Jupyter
Pre-built external packages	CRAN, Bioconductor	PyPI, Biopython
Scientific computing	In-built + Hundreds of packages	NumPy, SciPy + Hundreds of packages
Data wrangling & visualization	Tidyverse	Pandas, Seaborn

There are hundreds of software packages for bioinformatics & computational biology written in various languages (C, C++, R, & Python) that can be run from the command-line.

- Linux command-line
  - Navigating the file system
  - Running code
  - Manipulating data
  - Writing shell scripts

Programming languages & software ecosystems

Notebooks

- Code
- Documentation
- Results: plots, tables, or any other output
- Text descriptions of background/motivations/conclusions

Open science

Code: The field has dramatically shifted in thinking on how to publish code.

- Code used in research should be made available for research use free of charge.
- This is not just code for downloading & using. Original code must be made publicly available for others to use, review, and edit.
- Most common way to share code: GitHub.

Scientific publishing: Preprints

- Rapid publication of new science + free access (e.g. bioRxiv).
- Major source of cutting-edge research.
- Can have multiple (progressively better) versions of each manuscript.
- Preprints have NOT been peer-reviewed for quality and soundness of science. So, read/use with caution.

Getting help – Additional reading

- Fantastic resources on Reproducible code, Data management, Getting published, and Peer review
  http://www.britishecologicalsociety.org/publications/guides-to/

- A Quick Guide to Organizing Computational Biology Projects
  https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424

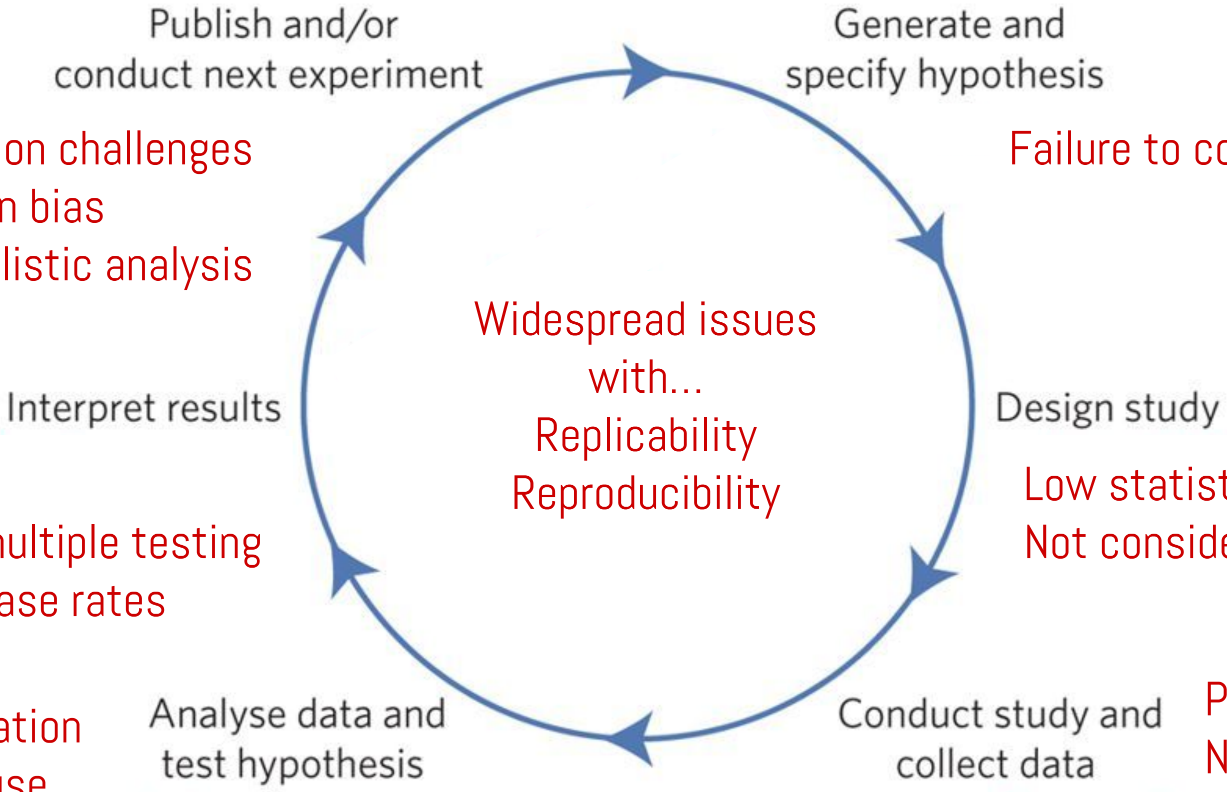
- A Quick Introduction to Version Control with Git and GitHub
  http://dx.plos.org/10.1371/journal.pcbi.1004668

- Ten Simple Rules for Taking Advantage of Git and GitHub
  http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004947

Getting help – Additional reading

- Ten Simple Rules for Creating a Good Data Management Plan
  https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004525
- Ten Simple Rules for Experiments' Provenance
  https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004384
- Ten Simple Rules for the Care and Feeding of Scientific Data
  https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003542
- Ten Simple Rules for Reproducible Computational Research
  https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003285
- Ten simple rules for documenting scientific software
  https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006561

What’s this course about?



Widespread issues with...
Replicability
Reproducibility

Failure to control for bias

Low statistical power
Not considering confounders

Poor quality control
Not recording all metadata

No exploration
Model abuse
P-hacking

P-hacking
Ignoring multiple testing
Ignoring base rates

Visualization challenges
Publication bias
Lack of holistic analysis

https://www.nature.com/articles/s41562-016-0021

Measuring associations between continuous variables

- Linear correlation is not appropriate for most cases.
- It is very easy to find spurious correlations/associations when testing many variables.
- Correlation does not imply causation.

Data visualization

- Visualization is an integral component of your analysis and research, not just for summarizing final/important findings!
- Visual inference is as powerful as statistical inference. Do not underestimate the power of exploratory visual data analysis.
- Allow the reader to: i) confirm that the statistical analysis is appropriate for the study design, and ii) critically evaluate the data.
- Plot (different facets of) your data and overlay additional information/metadata.
- Plots can be deceiving: Bar plots are terrible for continuous data with small sample size. Show the actual data using dot plots and add box/violin plots for data with medium-to-large sample sizes.
- No pie charts or 3D either. Beware of axes.

Some general thoughts

- Conscious ignorance: from unknown unknown → known unknown
  - Dunning-Kruger effect: knowing that something is unknown is as hard as knowing that thing!
  - The importance of feeling stupid: threshold of learning something new!
- Intelligent persistence
  - I don't understand this → What about this don't I understand?
  - Gaps in my knowledge → Gaps in collective knowledge

Some general thoughts

If your experiment needs statistics, you ought to have done a better experiment.

– Ernest Rutherford

He uses statistics as a drunken man uses lamp-posts... for support rather than illumination.

– Andrew Lang

The first principle is that you must not fool yourself, and you are the easiest person to fool.

– Richard Feynman

Some general thoughts

I *firmly* believe in the power of statistical enquiry, data analysis, and visualization.

The point is, because many of the ideas involved are complex and unintuitive, we need to develop a new set of skills to carefully use this power.

I have provided an ‘intense bootcamp’ style introduction to these skills.

Rationality is not about knowing the facts; it's about knowing which facts are relevant.

Some general thoughts

- Thank you for all the discussions and active engagement!
- Keep in touch and let me know all the cool things you go on to do :)