PRIMER 1
Organizing & managing a computational biology project

- Context for this class
- You & your learning
- Methods & software
- Organizing a comp. biology project
- Managing data and code
- Version control
- Programming lang. & software ecosystems

Midterm project presentation

In addition to the usual things (background, problem, approach, etc.):

- Clear flowchart of approach:
  - Raw data ⟶ Preprocessing & quality control ⟶ Preliminary/exploratory analysis ⟶ Analysis/Model-building steps ⟶ Expected outcomes.
- Thorough exploration & sanity checks of data:
  - Tables & plots to showcase various aspects of your datasets/problem.
- Method/software
  - Usage & I/O format for each.
- Preliminary analysis with:
  - Simple baselines, Samples datasets, and Toy examples.

Midterm project presentation

In addition to the usual things (background, problem, approach, etc.):

- **Clear flowchart of approach:**
  - Raw data ⟶ Preprocessing & quality control ⟶ Preliminary/exploratory analysis ⟶ Analysis/Model-building steps ⟶ Expected outcomes.
- Thorough exploration & sanity checks of data:
  - Tables & plots to showcase various aspects of your datasets/problem.
- Method/software
  - Usage & I/O format for each.
- Preliminary analysis with:
  - Simple baselines, Samples datasets, and Toy examples.

Reading papers: Learning to do research

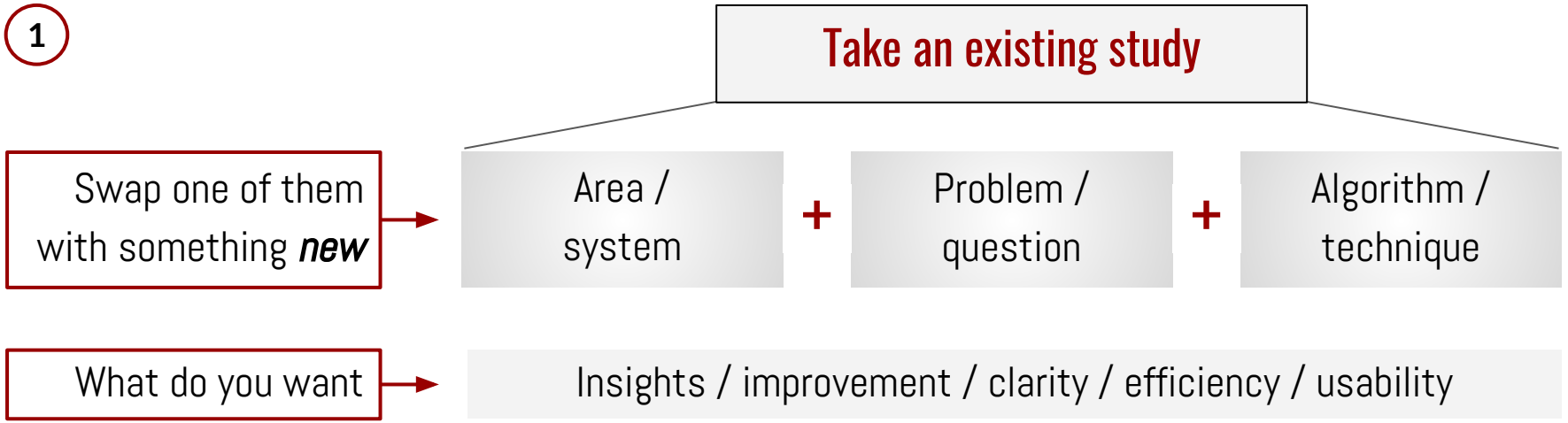
What can you learn from a paper?

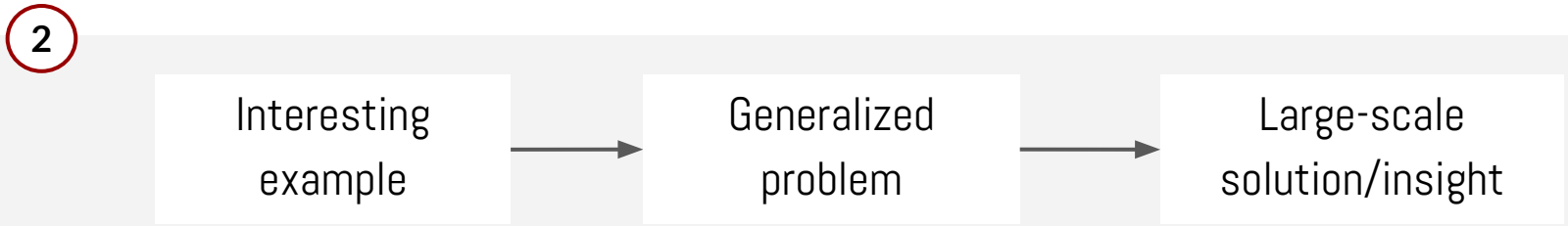
- Learn how to frame a problem
- Choose the methods/tools
- Set up an analysis workflow
- Establish groundwork, &
- Generate a series of supportive results towards answering the central question.

Types of computational research studies

- New analytical/computational method
- Improvement of an existing method
- Evaluation of existing methods
- Development of (re-)usable software, web-service, or database
- New insights w/ new/existing methods

Reading papers: Learning to do research





Reading papers: Look into other types of sources

Review articles

- Biological topics/concepts
- Methodological concepts/approaches

Great way to learn:

- The “thinking” and vocabulary of a sub-field
- Major papers and scientific milestones
- Open questions

Reading papers: Look into other types of sources

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

Reading papers: reading, retention, & reuse

- **Make reading** papers & online materials **a habit**.
- **Critically analyze what you read**/hear. 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 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 glossary** 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.

Midterm project presentation

In addition to the usual things (background, problem, approach, etc.):

- Clear flowchart of approach:
  - Raw data ⟶ Preprocessing & quality control ⟶ Preliminary/exploratory analysis ⟶ Analysis/Model-building steps ⟶ Expected outcomes.
- **Method/software:**
  - Usage & I/O format for each.
- Thorough exploration & sanity checks of data:
  - Tables & plots to showcase various aspects of your datasets/problem.
- Preliminary analysis with:
  - Simple baselines, Samples datasets, and Toy examples.

Picking up a new method or software

Read software/methods papers

- Read the Introduction & Discussion.
- Modern papers also have **graphical schematics of their methods/algorithms**.
- 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.

Picking up a new method or software

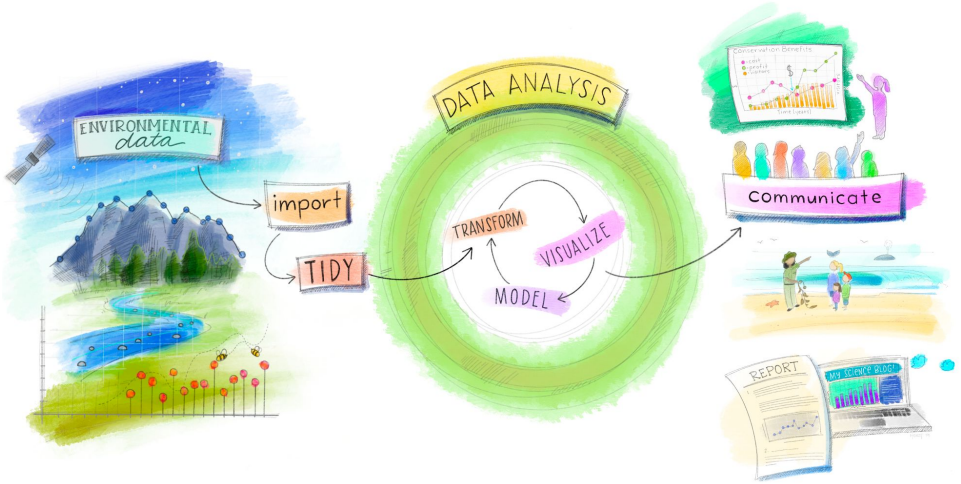
Explore the actual software/code

- 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.

Organizing a computational biology 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 computational biology 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 computational biology 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

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.

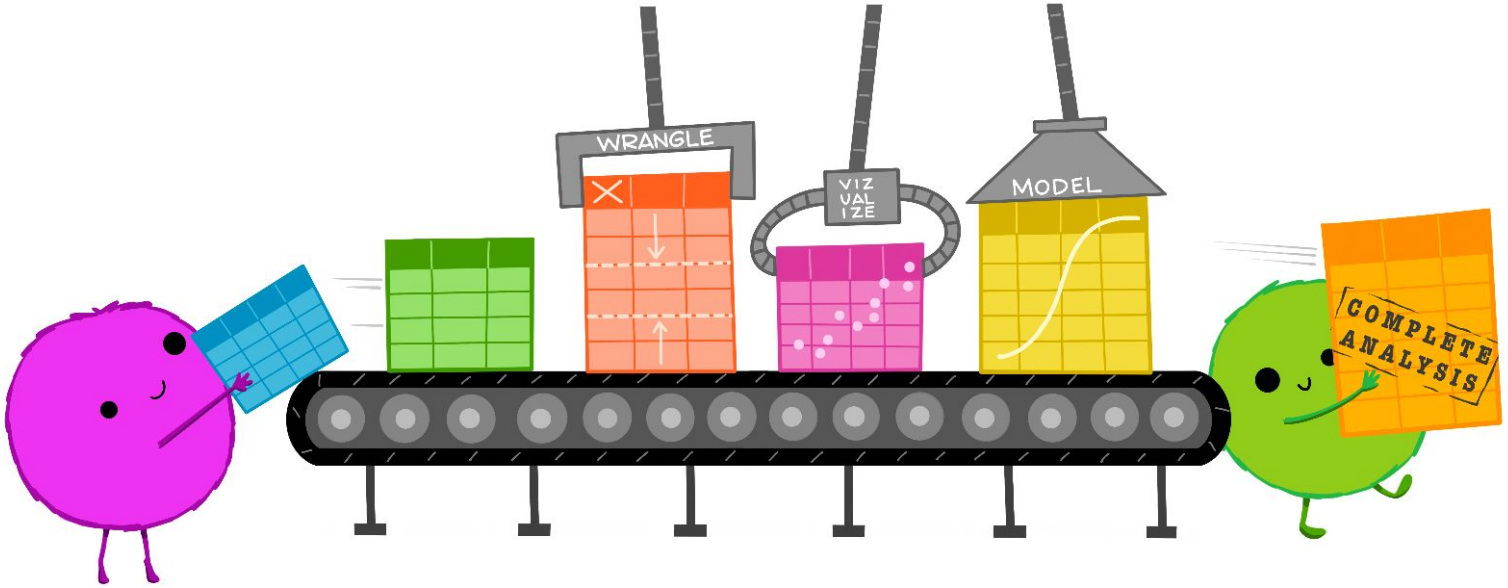
Used at the later stages of a project to pull all the results into a report/paper.

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 & code: automate everything (as much as possible)



Managing 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

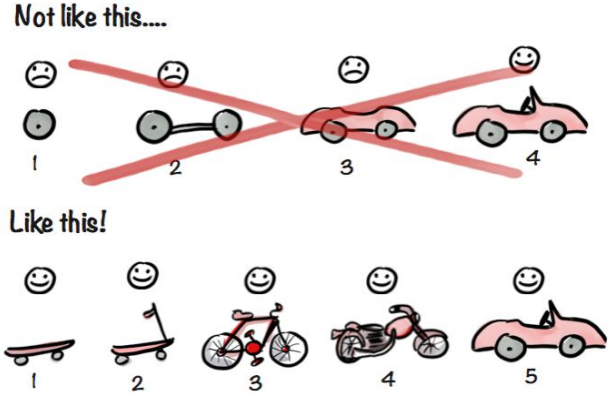
BAD	BETTER
`01.R`	`01_download-data.R`
`abc.R`	`02_clean-data_functions.R`
`fig1.png`	`fig1_scatterplot-bodymass-v-brainmass.png`
`IUCN's metadata.txt`	`2016-12-01_IUCN-reptile_shapefile_metadata.txt`

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

Managing 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

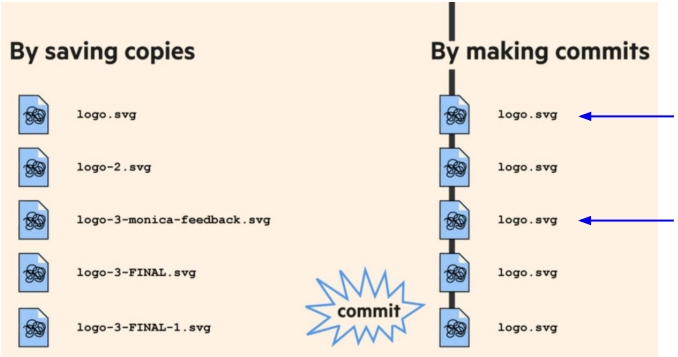
Reusing existing code:

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

Managing data & 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

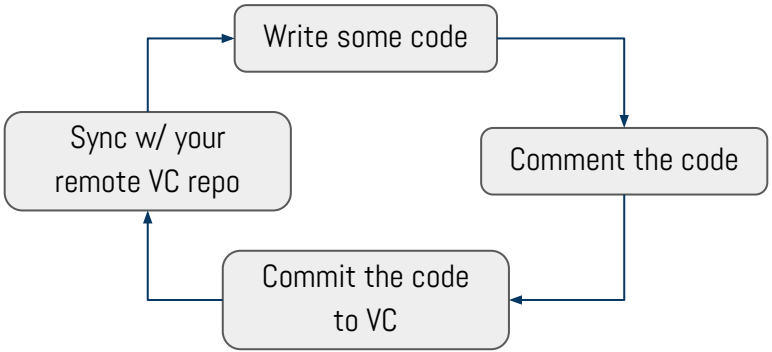
Updated background color

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

Programming languages & software ecosystems

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

Getting help – Additional reading

- Checkout all the references cited in the slides.
- So you want to be a computational biologist? https://www.nature.com/articles/nbt.2740
- A Quick Guide for Developing Effective Bioinformatics Programming Skills http://dx.plos.org/10.1371/journal.pcbi.1000589
- Ten Simple Rules for Effective Computational Research http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003506
- Good Enough Practices in Scientific Computing http://arxiv.org/abs/1609.00037
- Ten simple rules for documenting scientific software https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006561

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