

# Primer 1: Organizing & managing a computational biology project

- Context for this class
- Organizing a comp. biology project
- Managing data and code
- Version control
- Programming lang. & software ecosystems
- Getting help

# Midterm project presentation + report

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.

# Final project report + presentation

- Final report (regular sections similar to a research paper):
  - Abstract
  - Introduction
  - Data and Methods
  - Results & Discussion
  - Limitations & Future Directions
  - References
  - Glossary
- Code & Results in a well-organized GitHub repository
  - Well-documented code download/process data, perform computational analyses, generate all the results including plots/tables)
  - Detailed documentation on how to run everything

# Organizing a computational biology project

## project\_directory

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

- **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

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.

# 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 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.

# Managing data and code

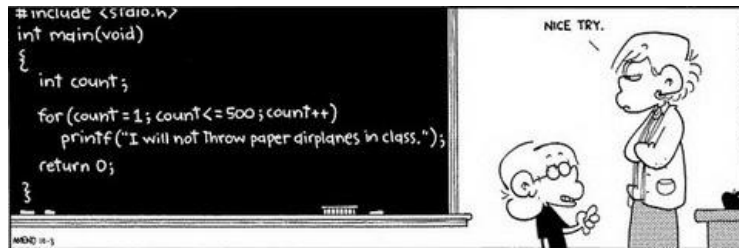
## Data

- 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>



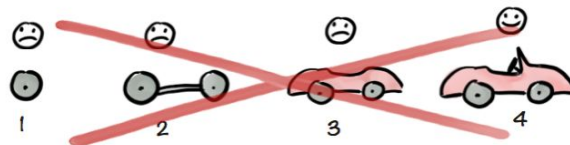
# Managing data and code

## Code

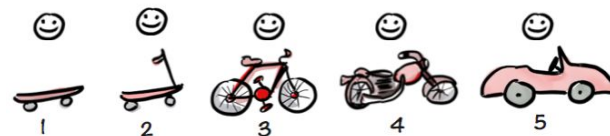
- Write code for both computers & humans.
  - Give descriptive, interpretable variable & function names.
  - Comment your code at the top: purpose, expected usage, example inputs/outputs, dependencies.
  - Record imports, constants, random seeds at the top.
  - Comment each block/function: the intended 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

Not like this....



Like this!



Spotify

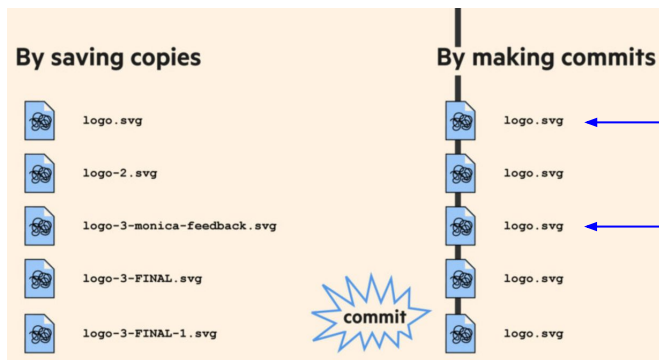
Reusing existing code:

- 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

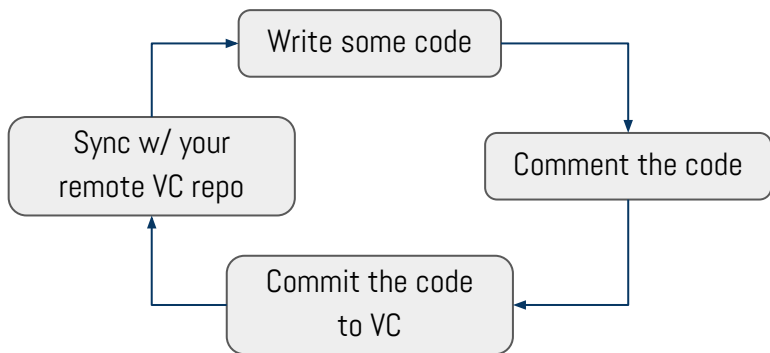


Arjun Krishnan  
12:34pm January 3th 2018

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](https://team.org/feedback314)



**repository**  
**commit**  
**remote**  
**clone**  
**push**  
**pull**  
**merge**

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



# Programming languages & software ecosystems

Language, IDE, Notebook

Pre-built external packages

Scientific computing

Data wrangling & visualization

- R | RStudio | R Notebook
- CRAN, Bioconductor
- In-built + Hundreds of packages
- Tidyverse

- Python | Rodeo | Jupyter
- PyPI, Biopython
- NumPy, SciPy + Hundreds of packages
- 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

- **Linux** | [rik.smith-unna.com/command\\_line\\_bootcamp](http://rik.smith-unna.com/command_line_bootcamp), [commandline.guide](http://commandline.guide), & [swcarpentry.github.io/shell-novice](http://swcarpentry.github.io/shell-novice)
- **Python** | Introduction: [learnpythonthehardway.org/book](http://learnpythonthehardway.org/book) & [developers.google.com/edu/python](http://developers.google.com/edu/python) | Data analysis: [jakevdp.github.io/WhirlwindTourOfPython](http://jakevdp.github.io/WhirlwindTourOfPython) | Visualization: [www.r-graph-gallery.com](http://www.r-graph-gallery.com)
- **R** | Introduction: [swcarpentry.github.io/r-novice-inflammation](http://swcarpentry.github.io/r-novice-inflammation) & [swirlstats.com](http://swirlstats.com) ('R Programming' & 'Data Analysis') | Data analysis: [r4ds.had.co.nz](http://r4ds.had.co.nz) | Visualization: [python-graph-gallery.com](http://python-graph-gallery.com)
- **Git & GitHub** | [swcarpentry.github.io/git-novice/](http://swcarpentry.github.io/git-novice/), [speakerdeck.com/alicebartlett/git-for-humans](http://speakerdeck.com/alicebartlett/git-for-humans), & [rogerdudler.github.io/git-guide/](http://rogerdudler.github.io/git-guide/)
- **Probability and Statistics** | Nature Collection (Statistics for Biologists | Practical Guides | Points of Significance): [www.nature.com/collections/qghhqm](http://www.nature.com/collections/qghhqm)
- **Genetics and Molecular Biology** | [learn.genetics.utah.edu/](http://learn.genetics.utah.edu/) & [www.genomicseducation.hee.nhs.uk](http://www.genomicseducation.hee.nhs.uk)



# Getting help

 ... so many excellent blog posts!



Many video lessons/courses  
on YouTube & elsewhere

*No shame!*

## StackOverflow Importer

Do you ever feel like all you're doing is copy/pasting from Stack Overflow?

Let's take it one step further.

from stackoverflow import quick\_sort will go through the search results of [python] quick sort looking for the largest code block that doesn't syntax error in the highest voted answer from the highest voted question and return it as a module. If that answer doesn't have any valid python code, it checks the next highest voted answer for code blocks.

```
>>> from stackoverflow import quick_sort, split_into_chunks

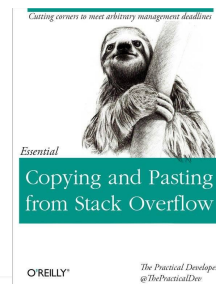
>>> print(quick_sort.sort([1, 3, 2, 5, 4]))
[1, 2, 3, 4, 5]

>>> print(list(split_into_chunks.chunk("very good chunk func")))
['very ', 'good ', 'chunk', ' func']

>>> print("I wonder who made split_into_chunks", split_into_chunks.__author__)
I wonder who made split_into_chunks https://stackoverflow.com/a/35107113

>>> print("but what's the license? Can I really use this?", quick_sort.__license__)
but what's the license? Can I really use this? CC BY-SA 3.0

>>> assert("nice, attribution!")
```



# 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>