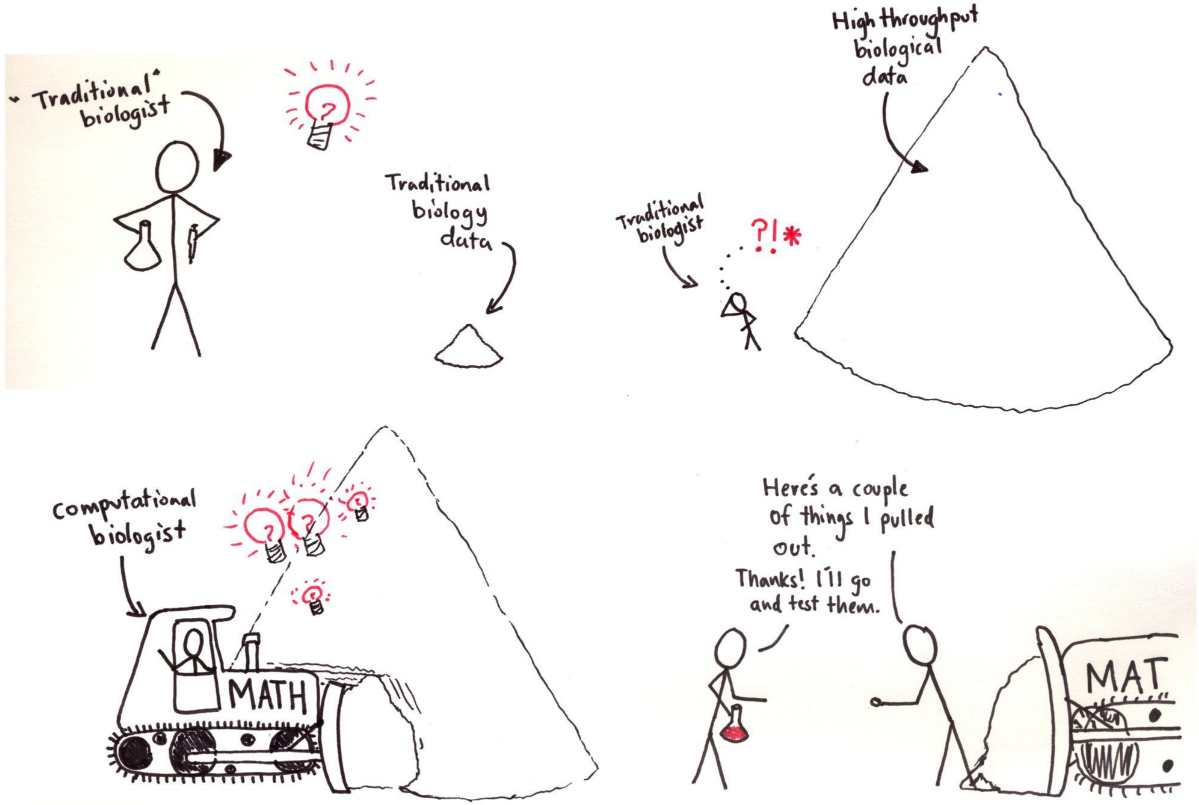
Lecture 2: Introduction and Overview

Getting started

- Data types and repositories
- Reading papers | Supp. materials
- Choosing a good problem
- Organizing a comp. biology project
- Programming lang. & software ecosystems
- Managing data and code
- High-performance computing @ MSU
- Getting help

What is computational biology?



@redpenblackpen

Topics, Lectures, and Paper discussions

- Genome assembly and annotation
- Sequence alignment and pattern finding
- Comparative genomics
- Genetic variation and quantitative genetics
- Regulatory genomics
- Functional genomics and data integration
- Molecular and digital evolution
- Molecular docking and molecular dynamics simulations
- Protein residue coupling and structure prediction
- Modeling cellular pathways
- Metabolomics and metabolic flux analysis
- Large-scale biological networks

Arjun presents

Each topic:

- Lecture
- Paper discussion

You present as team of two.

Data types and repositories

Literature search

PubMed

Google Scholar

Curated public data

Dryad

Repositive

Gene-expression & *-seq data

# data sets

NCBI GEO (Gene Expression Omnibus)

EBI ArrayExpress

# raw reads

NCBI SRA (Sequence Read Archive)

EBI ENA (European Nucleotide Archive)

# consortia

ENCODE | Roadmap Epigenomics

TCGA | ICGC

Genomes & genome browsers

# all encompassing

Ensemble

# genome browsers

UCSC Genome Browser

Integrative Genome Viewer

# ref. gene/transcript sequences

RefSeq

# ref. gene/transcript annotations

GENCODE

# everything protein

UniProt

Functional annotations

# biological processes, molecular functions, & cellular components

Gene Ontology

# coherent gene sets defined based on high-throughput approaches

MSigDB

Reading primary research papers | Supp. materials

Title & Abstract

Introduction

Data & Methods

Results

Discussion

References

1. Use **Title & Abstract** for only selecting paper. Read them last!

2. Read the **Introduction**:

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

   b. What are the then current approaches for solving that problem? What are their limitations that, according to the authors, need to be addressed?

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

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

   a. For each specific Q, note data (type & source) & method (algorithms/techniques, software, & approach). Pay attention to the **Supplemental materials**. These days much of the good stuff is in here!

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

Reading primary research papers | Supp. materials

Title & Abstract

Introduction

Data & Methods

Results

Discussion

References

4. 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.

5. Read the **Discussion/Conclusions**, Title, & Abstract:
   a. Step back to think about contributions, limitations, open Qs, & next steps.

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

Reading papers | Supp. materials

Journals to follow

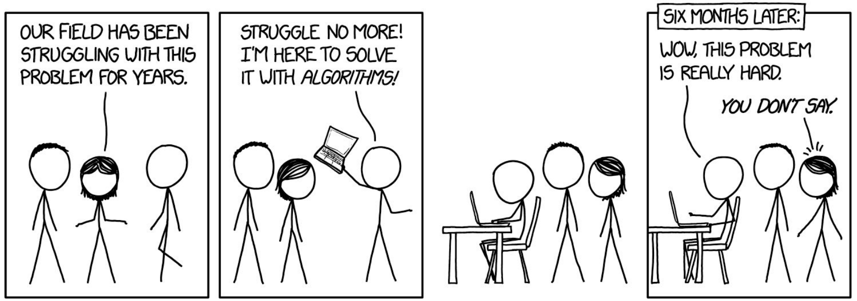
Bioinformatics
bioRxiv Bioinformatics
bioRxiv Genomics
BMC Bioinformatics
Briefings in Bioinformatics
Cell Systems
Genome Biology
Genome Research
Molecular Systems Biology
Nature Genetics
Nature Methods
Nucleic Acids Research
PLoS Computational Biology

Cell
eLife
Nature
Nature Biotechnology
PNAS
Science
Science Translational Medicine

PubMed / Google Scholar Alerts

Choosing a good problem for computational biology

xkcd.com/1831


OUR FIELD HAS BEEN STRUGGLING WITH THIS PROBLEM FOR YEARS.
STRUGGLE NO MORE! I'M HERE TO SOLVE IT WITH *ALGORITHMS!*
SIX MONTHS LATER:
WOW, THIS PROBLEM IS REALLY HARD.
YOU DON'T SAY.

New | Area / system | Problem / question | Algorithm / technique

I Want | Insights / improvement / clarity / efficiency / usability

Interesting example → Generalized problem → Large-scale solution/insight

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 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.

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 scattered packages	NumPy, SciPy + Hundreds of scattered packages
Data wrangling & visualization	Tidyverse	Pandas, Seaborn

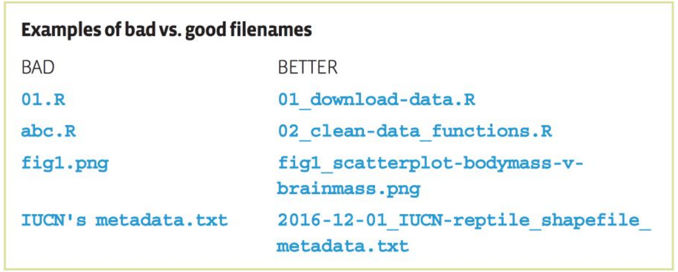
There are hundreds of software packages for bioinformatics & computational biology written in various languages (C, C++, R, & Python) that can be run from the command-line.

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

Managing data and code

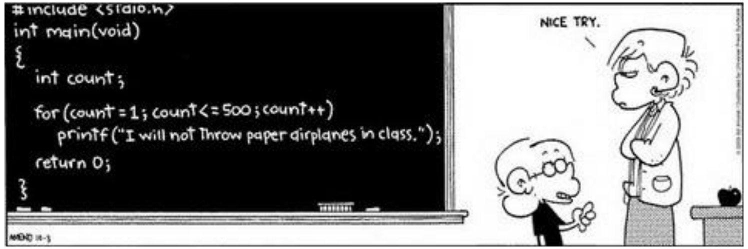
Data

- 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`



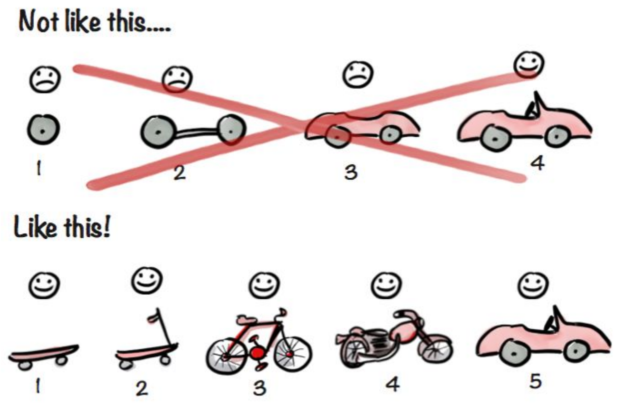
```
#include <stdio.h>
int main(void)
{
  int count;
  for (count = 1; count <= 500; count++)
     printf("I will not Throw paper airplanes in class.");
  return 0;
}
```

Managing data and code

Code

- Write code for both computers & humans.
- Properly acknowledge code borrowed from elsewhere; Check license.
- Give descriptive & interpretable variable and 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.

Code development



Spotify

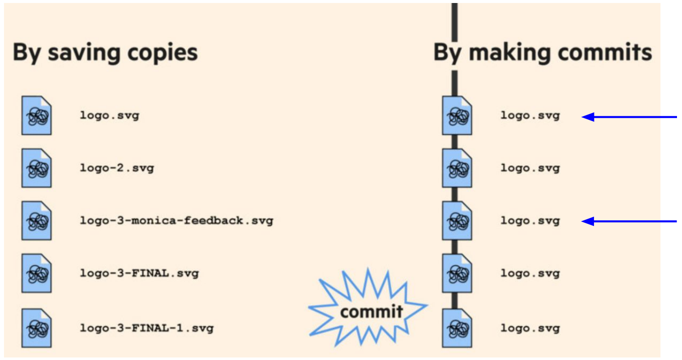
twitter.com/JennyBryan/status/952285541617123328

One of the most useful things I've learned from hanging out with (much) better programmers: don't wring hands and speculate. Work a small example that reveals, confirms, or eliminates something.

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

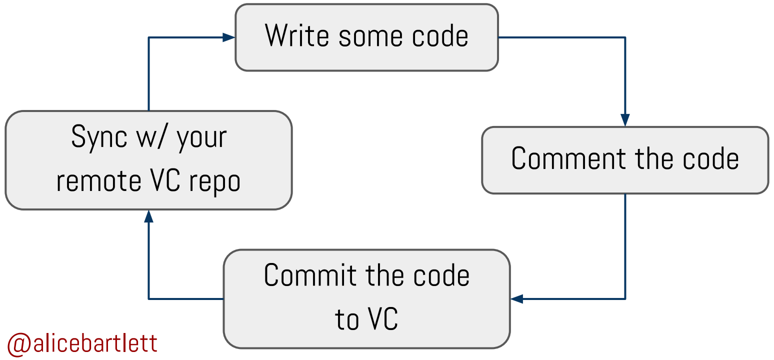
Updated background color

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

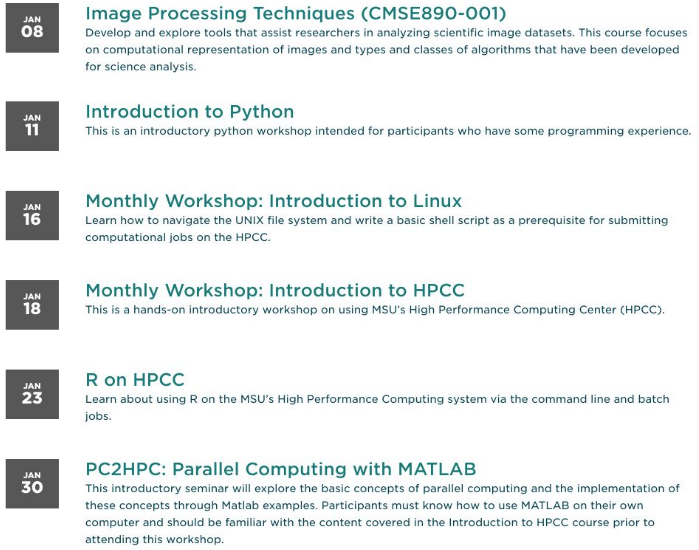


repository	Your project folder
commit	A snapshot of your repo
remote	A computer with the repository on it
clone	Get the repository from the remote for the first time
push	Send commits to a remote
pull	Get commits from a remote
merge	Combine two branches

@alicebartlett

High-performance computing @ MSU

- Excellent documentation: wiki.hpcc.msu.edu
- Training resources: www.icer.msu.edu/education-events/training-resources
- Seminars and workshops: www.icer.msu.edu/upcoming-workshops



JAN 08 **Image Processing Techniques (CMSE890-001)**
Develop and explore tools that assist researchers in analyzing scientific image datasets. This course focuses on computational representation of images and types and classes of algorithms that have been developed for science analysis.

JAN 11 **Introduction to Python**
This is an introductory python workshop intended for participants who have some programming experience.

JAN 16 **Monthly Workshop: Introduction to Linux**
Learn how to navigate the UNIX file system and write a basic shell script as a prerequisite for submitting computational jobs on the HPCC.

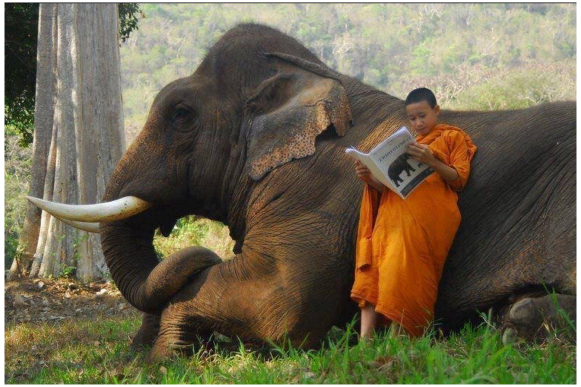
JAN 18 **Monthly Workshop: Introduction to HPCC**
This is a hands-on introductory workshop on using MSU's High Performance Computing Center (HPCC).

JAN 23 **R on HPCC**
Learn about using R on the MSU's High Performance Computing system via the command line and batch jobs.

JAN 30 **PC2HPC: Parallel Computing with MATLAB**
This introductory seminar will explore the basic concepts of parallel computing and the implementation of these concepts through Matlab examples. Participants must know how to use MATLAB on their own computer and should be familiar with the content covered in the Introduction to HPCC course prior to attending this workshop.

Getting help

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



Getting help - there's so much out there!

Google

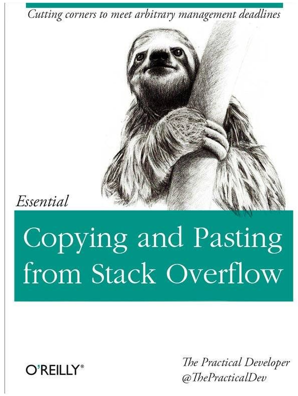
stackoverflow

No shame!

Video tutorials

data school

DataCamp

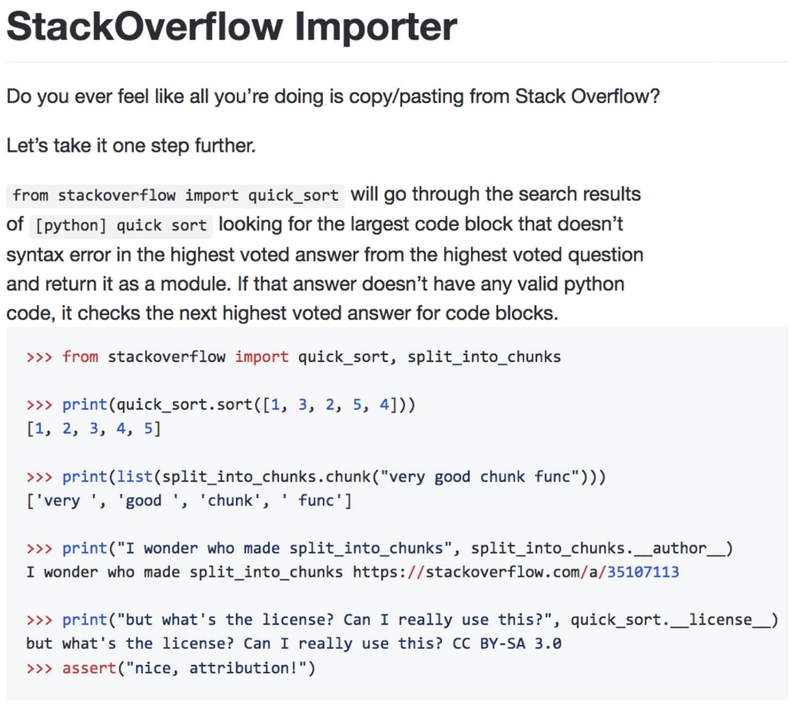

Cutting corners to meet arbitrary management deadlines

Essential

Copying and Pasting from Stack Overflow

O'REILLY®

The Practical Developer
@ThePracticalDev


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!")
```