Lectures 2 & 3: Getting started in bioinformatics & computational biology

- Lay of the bioinfo-compbio land
- Reading papers | Framing the problem
- Choosing a good problem
- Data types and repositories
- Programming lang. & software ecosystems
- Organizing a comp. biology project
- Managing data and code
- Resources @ MSU
- Getting help

Bioinformatics & Computational Biology

Computational biology

- The study of biology using computational techniques.
- Goal: learn new biology, knowledge about living systems. *It is about science.*

Bioinformatics

- The creation of tools (algorithms, databases) that solve problems.
- Goal: build useful tools that work on biological data. *It is about engineering.*

Margaret Dayhoff – The first bioinformatician

Applying math & computational techniques to the sequencing of proteins and nucleic acids.

- 1965: First collection of protein seqs. Single-letter code for amino acids.
- 1966: 'Evolutionary trees'.
- 1978: First AA similarity-scoring matrix.
- 1980: Launched the Protein Information Resource, the first online database system that could be accessed by telephone line.

Bioinformatics & Computational Biology – Today

"Computational thinking and techniques are so central to the quest of understanding life that today **all biology is computational biology**.

- Brings order into our understanding
- Makes biological concepts rigorous and testable, and
- Provides a reference map that holds together individual insights.

The next modern synthesis in biology will be driven by mathematical, statistical, and computational methods being absorbed into mainstream biological training, turning biology into a quantitative science."

Shifting roles of computational biologists

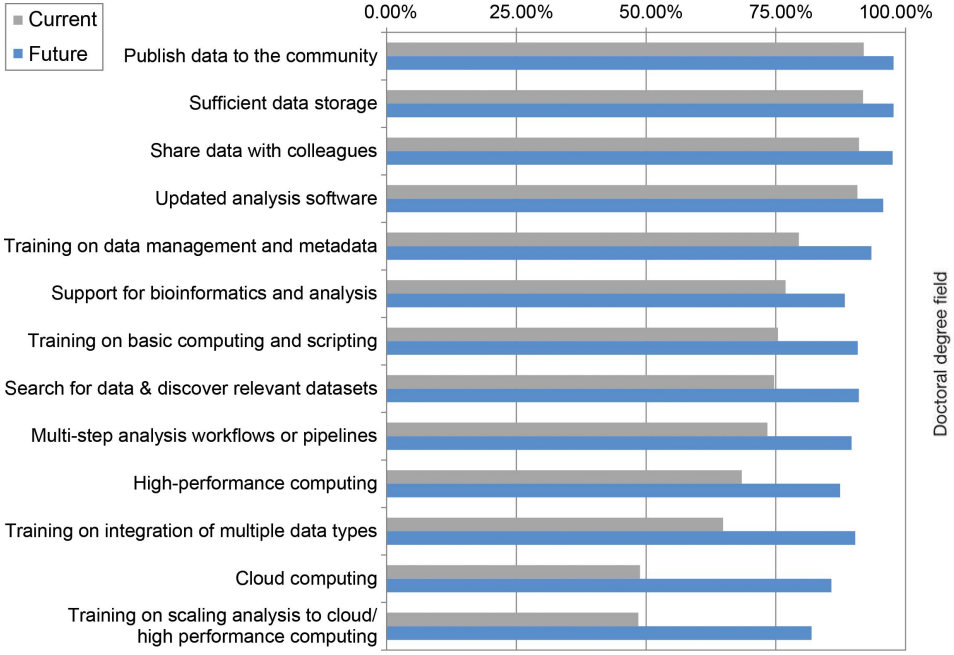
	Past	Current
Role in research	Supportive	Driver of research
A feeling for the biology	Computer science-centered	Biology- and computer science-centered
Environment	Isolated	Integrated
Data generation	Constrained	Resourceful
Data exploration	Largely limited to hypothesis testing	Both exploratory and hypothesis testing

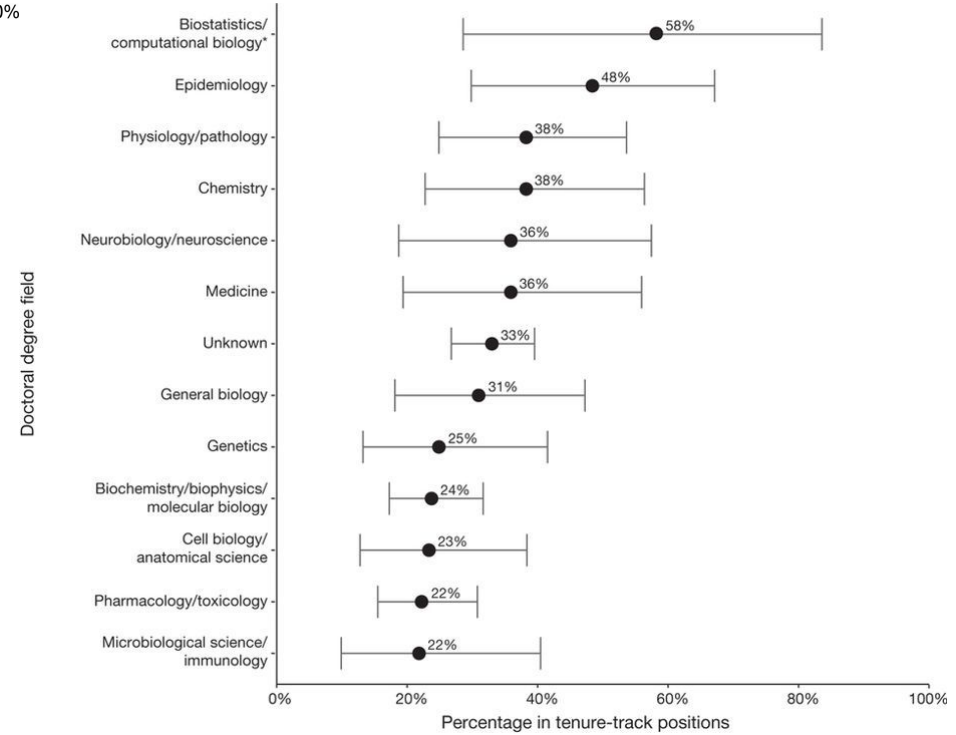
Markowetz (2017) PLoS Comp. Biol.
Yanai & Chmielnicki (2017) Genome Biol.

Some broad research areas & related analytical methods

Genome assembly and annotation	de Bruijn graphs, Hidden Markov Models
Sequence alignment and pattern finding	Dynamic programming
Molecular evolution and comparative genomics	Tree construction, Suffix trees
Genetic variation and quantitative genetics	Statistical inference, Multiple testing
Regulatory genomics	Expectation maximization, Gibbs sampling
Functional genomics and data integration	Dimensionality reduction, Machine learning
RNA/Protein structure prediction	Maximum entropy modeling
Molecular docking and dynamics simulations	Atomic, physical simulation
Artificial life and digital evolution	Artificial life simulation
Modeling signaling, regulatory pathways	Dynamical simulation, State space, Bifurcations
Metabolic reconstructions and dynamic models	Linear programming
Large-scale biological networks	Graph theory, Label propagation

Opportunities & Unmet needs





Barone et al. (2017) PLoS Comp. Biol.; Xu et al. (2018) Nat. Biotech

Community – Meetings / Conferences; MSU Seminars

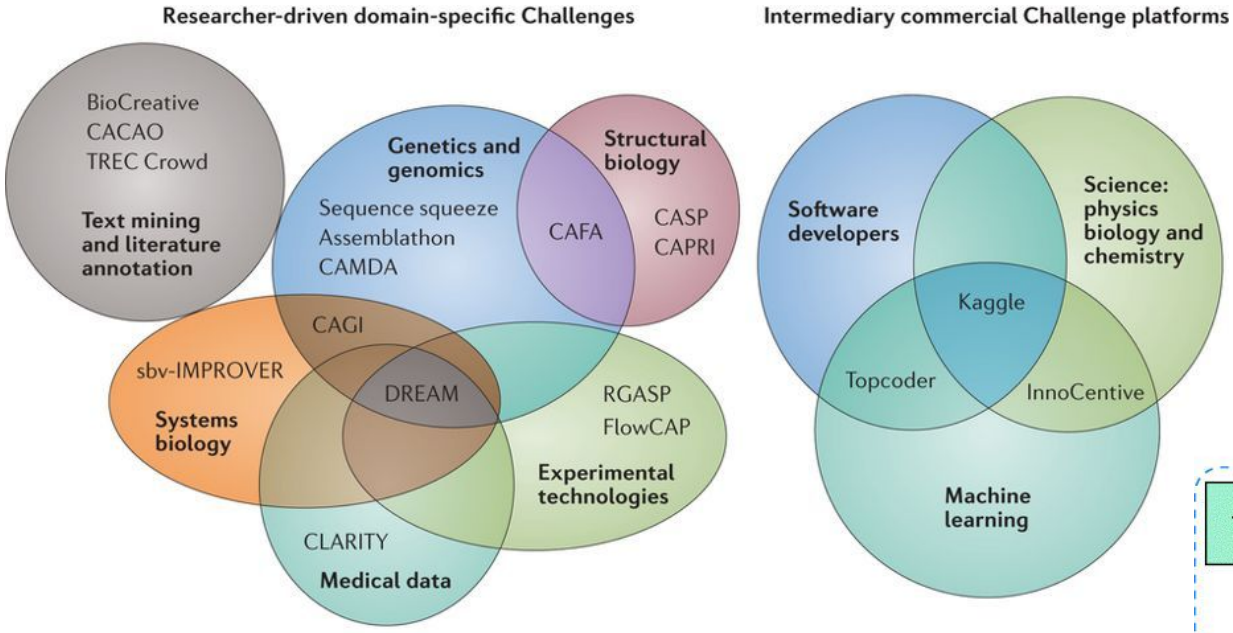
Some National/International Conferences

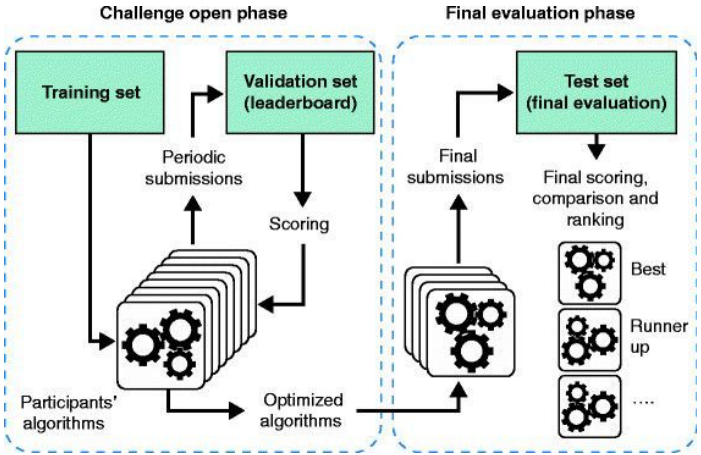
- Intelligent Systems for Molecular Biology
- Research in Computational Molecular Biology
- Pacific Symposium on Biocomputing
- ACM Conference on Bioinformatics, Computational Biology, & Health Informatics
- Rocky Mountain Bioinformatics Conference
- Great Lakes Bioinformatics Conference
- Cold Spring Harbor Laboratories Meetings (Network Biology, Genome Informatics)

Relevant MSU Seminar Series

- Science at the Edge
- Machine Learning (CSE)
- Computational Math, Sci, & Engg
- Institute for Quant. Health Sci. & Engg

Community – Open Challenges





Saez-Rodriguez et al. (2016) Nat. Rev. Genet.; Boutros et al. (2014) Genome Biol.

Reading primary research papers – Learning to frame the problem

Great way to 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

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 Med.

PubMed Alerts
Google Scholar Alerts

Reading primary research papers

Title & Abstract

Introduction

Data & Methods

Results

Discussion

References

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

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.

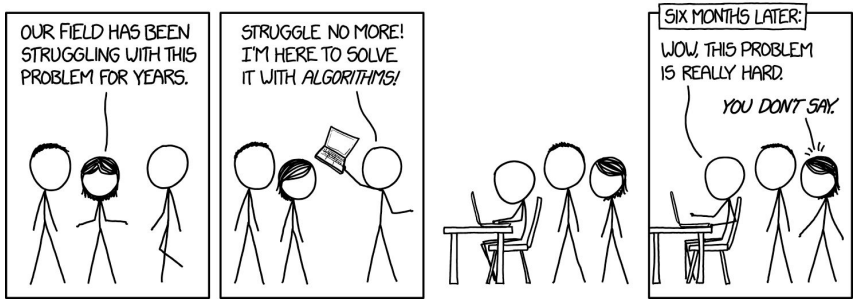
Reading primary research papers

Reading, Retention, and Reuse

- Make reading papers a habit.
- Critically analyze what you read/hear. Don't be swayed by high-profile papers, media hype, or current dogma.
- Use a reference manager (e.g. Zotero), put *everything* you read into it. Add add notes about specific take-homes. Use tags to group papers by subfield/method/data.
- Create and maintain a single source of all the technical terms and vocabulary for your project.
- Create and maintain a single source (R/Jupyter Notebook) with notes/text-excerpts/figures from all papers & reading materials.
- 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.

Choosing a good computational biology problem

xkcd.com/1831



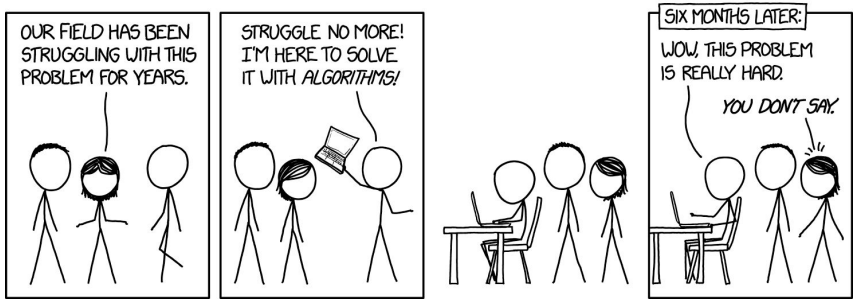
New | Area / system | Problem / question | Algorithm / technique

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

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

Choosing a good computational biology problem

xkcd.com/1831



Explore and prototype early to fail fast and learn

- Exploration + prototyping: critical for determining if the problem is well-defined & tractable.
- Perform preliminary analysis with simple baselines, sample datasets, and toy examples.
- Don't speculate or make assumptions. Instead, implement something and check them.
- The value lies not in the code/plots you produce, but in the lessons you learn.

Data types and repositories – some examples

Genomes & proteomes

all encompassing

Ensemble

comparative genomics

COGs | InParanoid | OrthoMCL

ref. gene/transcript sequences & annotations

RefSeq | Entrez | GENCODE

sequences variation

1000 Genomes | dbSNP

everything protein

UniProt | InterPro | SCOP | CATH | PDB

Functional annotations & relationships

biol. processes, mol. functions, cellular components

Gene Ontology

pathways

Reactome, KEGG, WikiPathways

networks

BioGRID, TRANSFAC, STRING

Phenotype-, Disease-association

OMIM | GWAS Catalog | ClinVar | COSMIC

Genome-Phenome

dbGaP | UK Biobank

Functional/regulatory genomics

data sets

NCBI GEO | EBI ArrayExpress

raw reads

NCBI SRA | EBI ENA

consortia

ENCODE | Roadmap | GTEx | TCGA

curated public data

Dryad | Repositive | Expression Atlas

Model organism databases

MGI | RGD | TAIR | FlyBase | WormBase | ZFin | SGD

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

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.

Based on Noble (2006) PLoS Comp. Biol.

Managing data and code

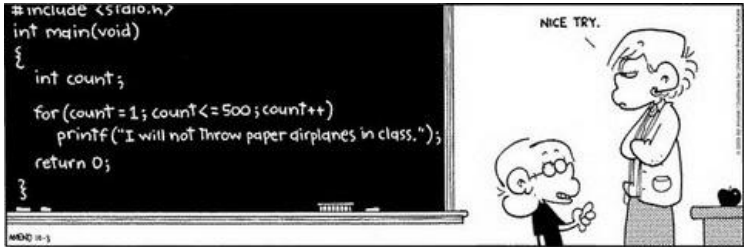
Data

- 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

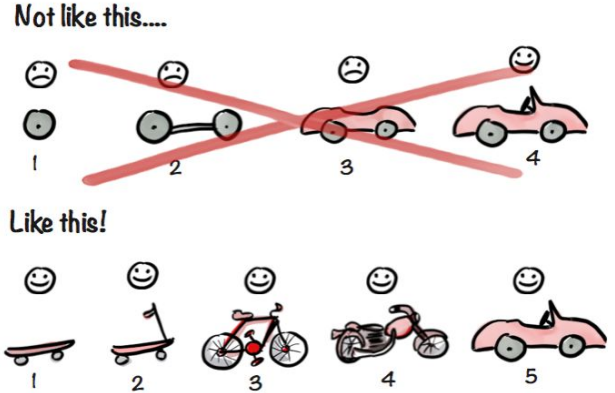


Managing data and code

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.
- Properly acknowledge code borrowed from elsewhere; Check license.
- Program for the general case, and put the specifics outside the code as arguments & parameters.

Continuously functional & testable



Spotify

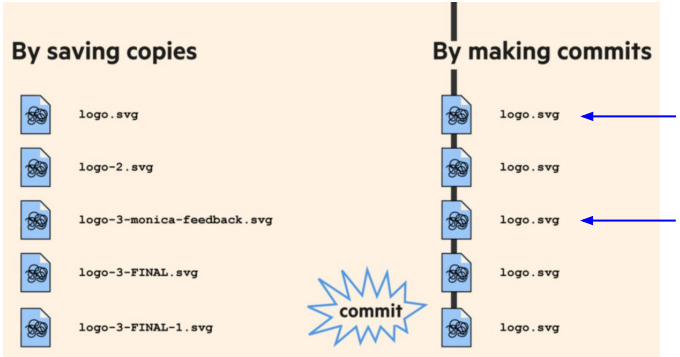
twitter.com/JennyBryan/status/952285541617123328

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

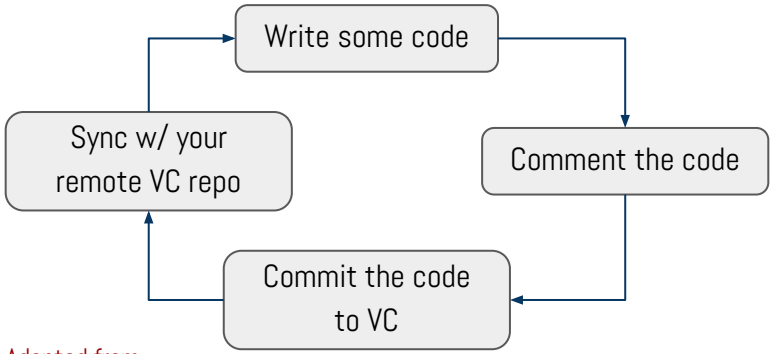
Updated background color

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



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

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.

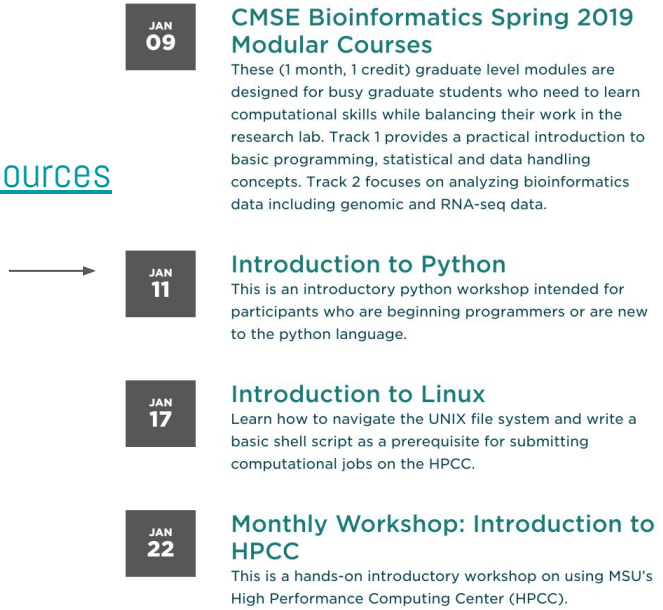
Scientific publishing: Preprints

- Rapidly publication of new science + free access
- Major source of cutting-edge research
- Public archives like bioRxiv.
- Preprints have NOT been peer-reviewed for quality and soundness of science.
  So, read/use with caution.

Resources @ MSU

Institute for Cyber-Enabled Research

- High-Performance Computing Cluster: wiki.hpcc.msu.edu
- Training resources: www.icer.msu.edu/education-events/training-resources
- Seminars and workshops: www.icer.msu.edu/upcoming-workshops
- Regular open office hours:
  - Every Monday & Thursday 1-2 p.m. at BPS Room 1440.

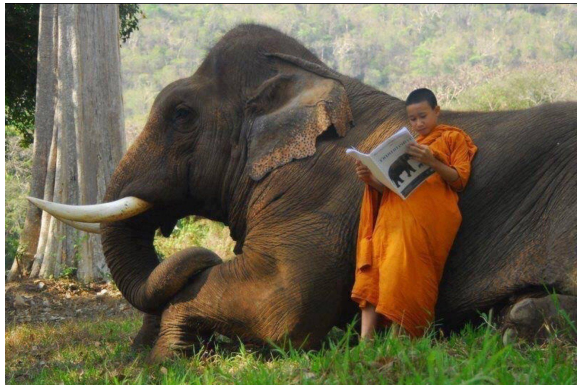


Working/student groups

- R-Ladies: https://rladies-eastlansing.github.io/
- MSU Data Science: http://msudatascience.com/

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
- **Git & GitHub** | swcarpentry.github.io/git-novice/, speakerdeck.com/alicebartlett/git-for-humans, & rogerdudler.github.io/git-guide/
- **Probability and Statistics** | Nature Collection (Statistics for Biologists | Practical Guides | Points of Significance): www.nature.com/collections/qghhqm
- **Genetics and Molecular Biology** | learn.genetics.utah.edu/ & www.genomicseducation.hee.nhs.uk



Getting help

Google ... so many excellent blog posts!

stackoverflow

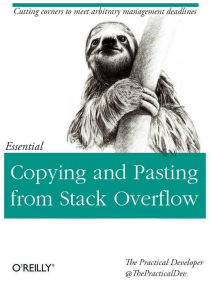
Biostars
BIOINFORMATICS EXPLAINED

Video lessons/courses

data school

DataCamp

... and much more on YouTube



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
- What Is the Key Best Practice for Collaborating with a Computational Biologist? https://www.cell.com/cell-systems/fulltext/S2405-4712(16)30223-X
- 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 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