Primer 2: Kick-starting & getting help in a computational biology project

- Context for this class
- You & your learning
- Methods & software
- Examining data & doing sanity checks
- Visual exploratory analysis
- Preliminary data analysis
- Getting help

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

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, Sample datasets, and Toy examples.

You and your learning

Primary research papers

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

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.

Methods & Data

- This is where most (if not all) of the action is:
  - Data types & sources
  - Algorithms/techniques, software, & approach
- Learn to read critically:
  - Are the data & methods describes sufficient to answer the Qs raised in the Intro?
- ALWAYS read the Supplementary Materials
  - These days much of the good stuff is in here!

You and your learning

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

You and your learning

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.

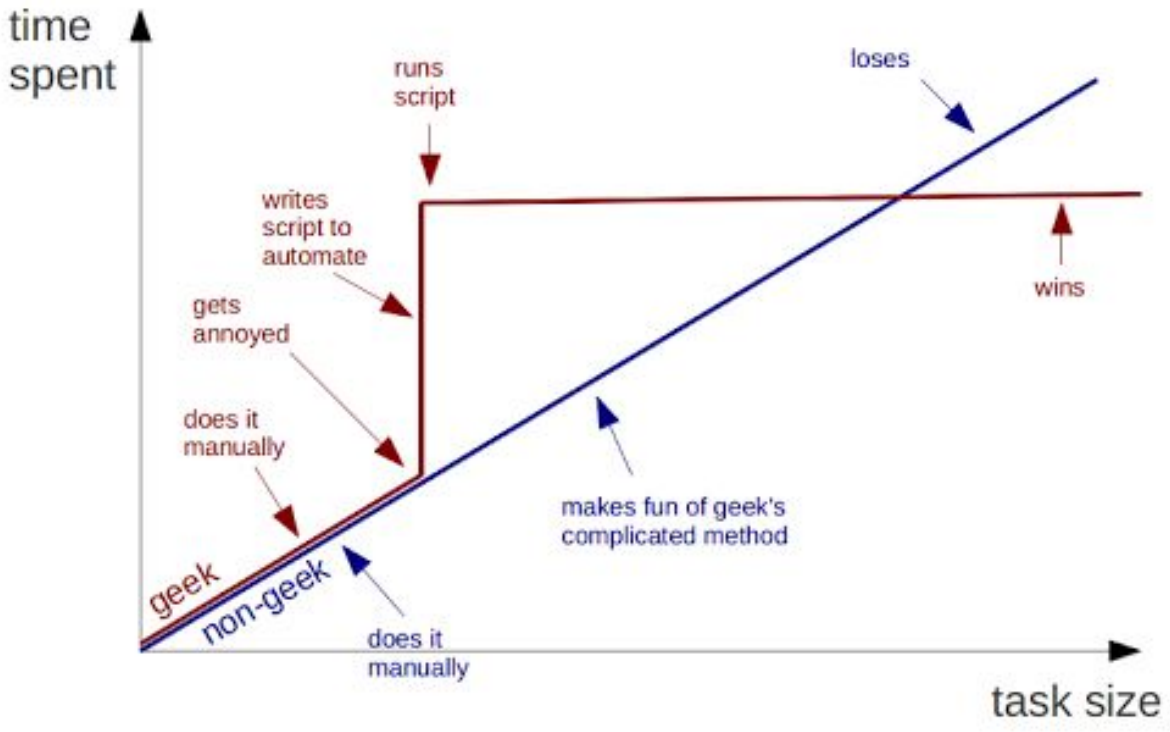
Great way to learn:

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

You and your learning – Reading, Retention, and 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 (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.

Automate everything (as much as possible)



Methods & 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.

- 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

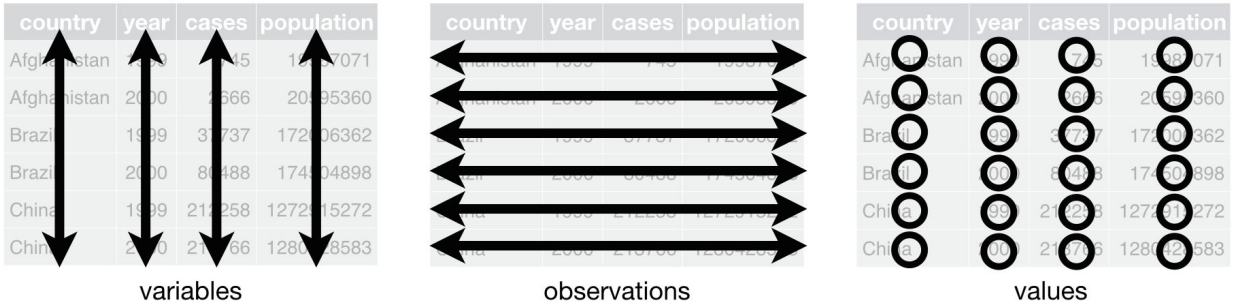
Explore the actual software/code

- 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.

- 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.

Data examination & sanity checks

"Tidy datasets are all alike, but every messy dataset is messy in its own way." –– Hadley Wickham



- Each variable must have its own column.
- Each observation must have its own row.
- Each value must have its own cell.

Data examination & sanity checks

- Number of rows & columns
- Head & tail (top & bottom rows)
- Exploring each column
  - Top & bottom 10 entries sorted by values in that column; Note the range of values
  - Continuous variables: Central & spread of values (mean, variance, quartiles, IQR)
  - Discrete variables: Unique values and their frequencies
  - Are there columns with mixed data types?
- Missing values:
  - Number of rows/columns with MVs
  - Histograms of number of MVs across rows/columns
  - Strategies for dealing with MVs? Drop rows/columns or Impute MVs.

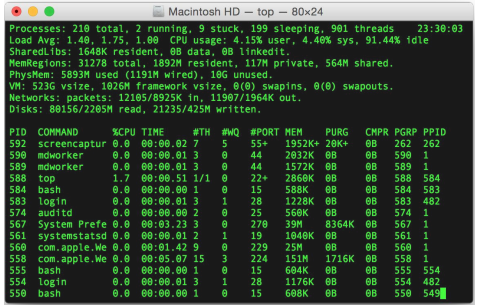
Data examination & sanity checks – the Linux command-line

`cd`	Change directory
`pwd`	Print working directory
`mkdir`	Make directory
`ls`	List
`cp`	Copy
`mv`	Move
`rm`	Remove

`less`	Peruse file
`head`	Print top of the file
`tail`	Print bottom of the file
`cat`	Print the whole thing
`wc`	Word count

`cut`	Cut columns
`sort`	Sort lines
`uniq`	Report/omit repeating lines
`grep`	Print lines matching pattern

https://explainshell.com



Data examination & sanity checks

Similar ideas hold for different types of data

- Large-scale gene expression (bulk and single-cell) — Eren, Gio, Merve
- Large-scale protein expression — Madeline
- Medium-scale metabolite profiles — Nandan
- Individual genome; Multiple genomes — Alexa, Karn
- Gene coexpression and protein-protein interaction networks — Yash
- Genome-wide associations of SNPs to traits — Alex
- Functional annotations of genes to pathways/processes — Tayler
- Drug structures and targets — Norman
- Volumetric Magnetic Resonance Imaging — Kaylen
- Electroencephalogram and Auditory brainstem response — Ethan

Data examination & sanity checks – Data types and repositories

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

Visual exploratory analysis - Let's be Tukey's friend

John Tukey
Known for ***Exploratory Data Analysis***

This is my favorite part about analytics: **Taking boring flat data and bringing it to life** through visualization.

When communicating results … **there is nothing better than a clear visualization to make your point**.

The greatest value of a picture is when it forces us to notice what we never expected to see.

Numerical quantities focus on expected values, **graphical summaries on unexpected values**.

Visualization is often used for evil - twisting insignificant data changes and making them look meaningful. Don't do that crap **if you want to be my friend**. Present results clearly and honestly. If something isn't working - those reviewing results need to know.

Visual exploratory analysis

- Distribution of each continuous variable (each column) [boxplot & violin plot]
- Proportion of observations (x) vs. Cumulative proportion of values (y) [line plot]
- Mean vs. variance of all observations (across samples; dispersion) [scatter plot + smooth]
- Correlation of mean value of each observation with additional metadata [scatter plot + smooth]
- Correlation of variables to each other [scatter plot + smooth]
- Clustering/similarity of variables [heatmap]
- Concordance with prior knowledge/expectations [barplot + boxplot]
- Agreement with existing data [scatter plot + smooth]

Visual exploratory analysis

1. Distribution of gene expression across all the samples [boxplot]
2. Proportion of genes (x) vs. Cumulative proportion of reads (y) [lineplot]
3. Mean across samples vs. variance across samples (dispersion) [scatter plot + smooth]
4. Plot correlation of expression with length, GC-bias, etc. [scatter plot + smooth]
5. Are replicates highly correlated with each other? [scatter plot + smooth]
6. Cluster samples and visualize similarities [heatmap]
7. Do known genes show the right pattern of expression? [barplot of single genes across samples; boxplot of groups of genes across samples] (e.g. genes that show a phenotype)
8. Does it agree with existing transcriptome data?

Preliminary data analysis – Fail fast and learn

- Exploration + prototyping
  - Critical for determining if the problem is well-defined & tractable.
- Perform preliminary analysis:
  - 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.

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.

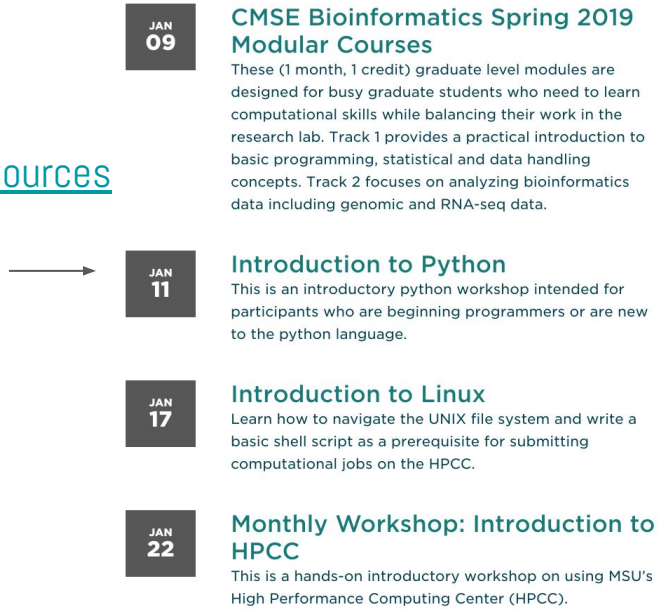
Preliminary data analysis

- Preliminary analysis
  - Simple baselines
    - Most frequent value
    - Average/median value
  - Randomized baselines
    - Identical method run on permuted data (various aspects)
  - Sample datasets & Toy examples
    - Make small datasets by hand to make sure your code or external software works exactly as expected.

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.

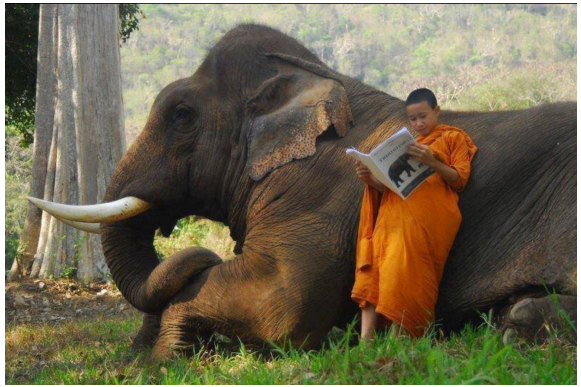


Working/student groups

- 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



You and your science

- **Think about your work**
  - Turn off the autopilot and take control.
  - Constantly critique and appraise your work.
- **Remember the big picture**
  - Don't get so engrossed in the details that you forget to check what is the goal and impact of your whole project/endeavour.

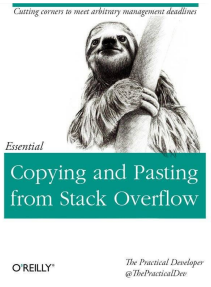
Getting help

Google ... so many excellent blog posts!

stackoverflow

Biostars
BIOINFORMATICS EXPLAINED

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