

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.
- Method/software:
 - Usage & I/O format for each.
- Thorough exploration & spot checks of data:
 - Tables & plots to showcase various aspects of your datasets/problem.
- Preliminary analysis:
 - Simple baselines, Sample datasets, and Toy examples.

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.
- Method/software:
 - Usage & I/O format for each.
- Thorough exploration & spot checks of data:
 - Tables & plots to showcase various aspects of your datasets/problem.
- Preliminary analysis:
 - Simple baselines, Sample 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: 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, 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 notebook** (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 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 + 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.
- **Method/software:**
 - Usage & I/O format for each.
- Thorough exploration & spot checks of data:
 - Tables & plots to showcase various aspects of your datasets/problem.
- Preliminary analysis:
 - Simple baselines, Sample datasets, and Toy examples.

Picking up a new method / 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.

Picking up a new method / 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.

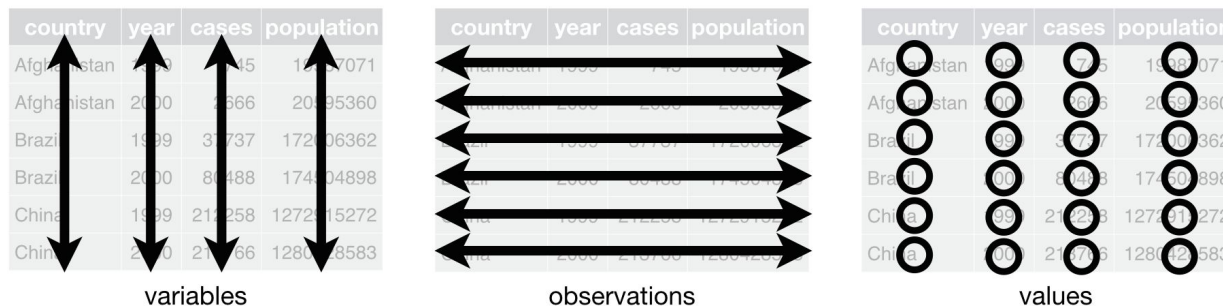
Midterm project presentation + report

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.
- Method/software:
 - Usage & I/O format for each.
- **Thorough exploration & spot checks of data:**
 - Tables & plots to showcase various aspects of your datasets/problem.
- Preliminary analysis:
 - Simple baselines, Sample datasets, and Toy examples.

Data examination & spot 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 & spot checks

- Data structure, dimensions, and scale:
 - Structure (rectangular, list of entries, dictionary, etc.) & format (plain text, spreadsheet)
 - Top & bottom entries; Number of rows/columns
- Exploring specific aspects of the data (e.g., different columns)
 - Top & bottom 10 entries sorted by values in that column
 - 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

Data examination & spot 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

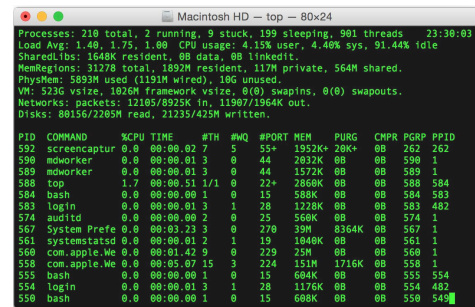
cut Cut columns

sort Sort lines

uniq Report/omit repeating lines

grep Print lines matching pattern

<https://explainshell.com>

A screenshot of a terminal window titled "Macintosh HD -- top -- 80x24". The window displays the output of the 'top' command, showing system statistics and a list of running processes. The statistics include: Processes: 218 total, 2 running, 9 stuck, 199 sleeping, 901 threads; Load Avg: 1.48, 1.75, 1.00; CPU usage: 4.15% user, 4.40% sys, 91.44% idle; Shared libs: 1648K resident, 0K data, 0K linkedin; MemRegions: 31278 total, 1892M resident, 117M private, 564M shared; PhysMem: 5893M used (1191M wired), 10G unused; VM: 523G vszize, 1026M framework vszize, 0(0) swappins, 0(0) swapouts; Networks: packets: 12105/8925K in, 11987/1964K out; Disks: 80156/2205M read, 21235/425M written. The process list shows columns for PID, COMMAND, %CPU, TIME, #TH, #MQ, #PORT, MEM, PURG, CNPR, PGPR, PPID. The 'top' process is highlighted in green.

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.
- Method/software
 - Usage & I/O format for each.
- Thorough exploration & spot checks of data:
 - Tables & plots to showcase various aspects of your datasets/problem.
- **Preliminary analysis:**
 - Simple baselines, Sample datasets, and Toy examples.

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.
- Make visualization an integral part of every stage of your project, including early exploration.
- 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

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

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.