PRIMER 2

Kick-starting & getting help in a computational biology project

- Context for this class
- Examining data & doing sanity checks
- Visual exploratory analysis
- Preliminary data analysis
- Writing & debugging code
- Getting help

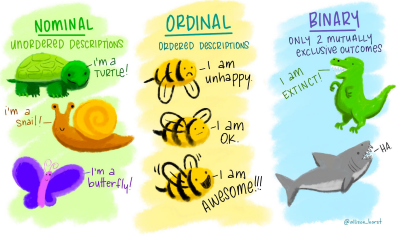
Midterm project presentation

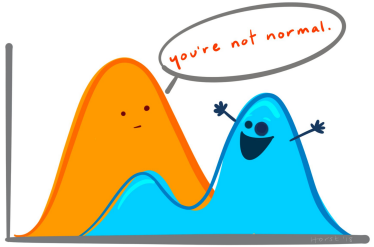
In addition to the usual things (background, problem, approach, etc.):

- 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 & sanity checks of data:**
  - Tables & plots to showcase various aspects of your datasets/problem.
- Preliminary analysis with:
  - Simple baselines, Samples datasets, and Toy examples.

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)



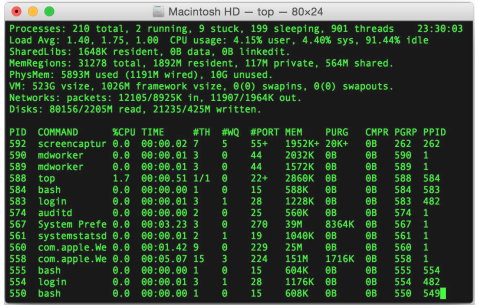


Data examination & spot checks – the Linux command-line

`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



Data examination & spot checks

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

"TIDY DATA is a standard way of mapping the meaning of a dataset to its structure."

–HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

each row an observation

Wickham, H. (2014). Tidy Data. Journal of Statistical Software *59* (10). DOI: 10.18637/jss.v059.i10

Midterm project presentation

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 & sanity checks of data:
  - Tables & plots to showcase various aspects of your datasets/problem.
- **Preliminary analysis with:**
  - Simple baselines, Samples 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.

Preliminary data analysis: public data 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

Annotations & relationships

process, function, component
Gene Ontology

pathways
Reactome, KEGG, WikiPathways

networks
BioGRID, STRING

Phenotype/Disease

OMIM | GWAS Catalog | ClinVar | COSMIC

Genome-Phenome

dbGaP | UK Biobank | FinnGen

Functional/regulatory genomics

data sets
NCBI GEO | EBI ArrayExpress

raw reads
NCBI SRA | EBI ENA

consortia
ENCODE | GTEx | TCGA

curated public data
Expression Atlas, Recount2, ARCHS4

Model organism databases

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

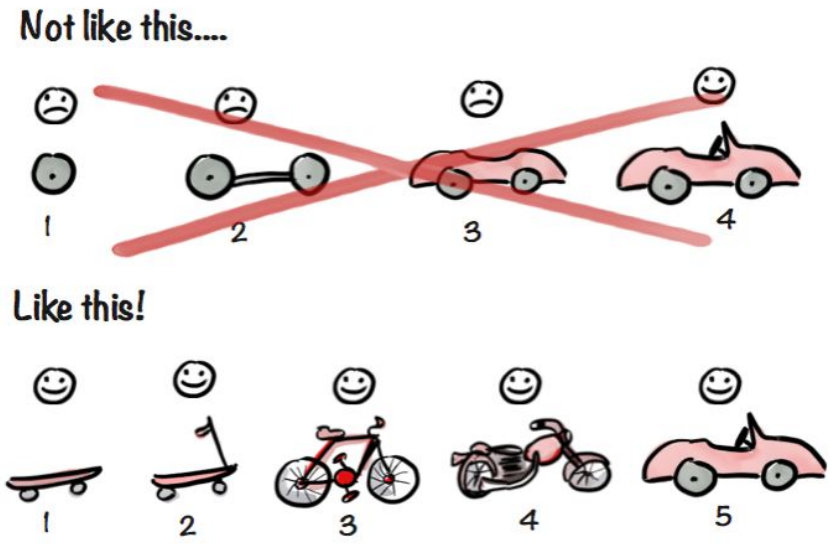
Writing code for the first time

- Give variables and functions longer meaningful names.
  - Use autocomplete to call functions/variables
- Make the code talk to you.
  - Generously add `print` statements that help you closely monitor the progress of your code through the various steps
    - Time stamps
    - Description of which step is currently running
    - File names
    - File paths
    - Variable names
    - Variable dimensions
    - Variable type/class
    - The first few elements of large variables

Writing code for the first time

- Write modular code
  - Decouple unrelated parts
  - Test at every stage of development

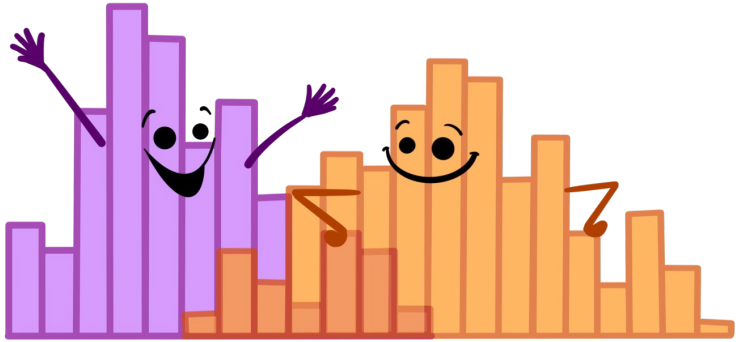
Continuously functional & testable



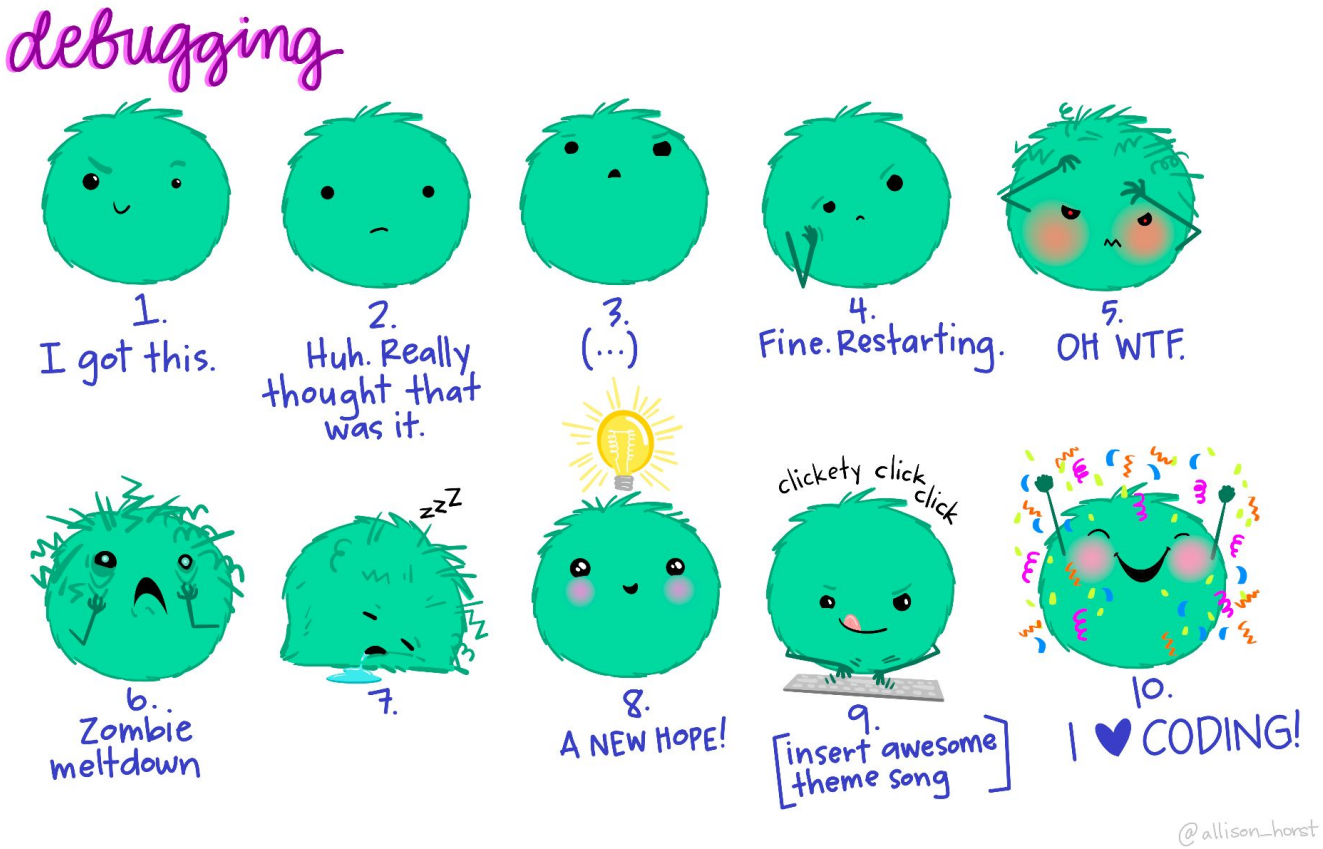
Spotify

Writing code for the first time

- Work with a smaller but representative input dataset that you *fully* understand and from which you know what the output(s) should exactly look like.
- Learn to create data inside your code
  - sampling, repeating, distributions, letters, built-in-datasets
- Run your code on shuffled/permuted data to see if the patterns/signals remain.



Debugging your code



Debugging your code

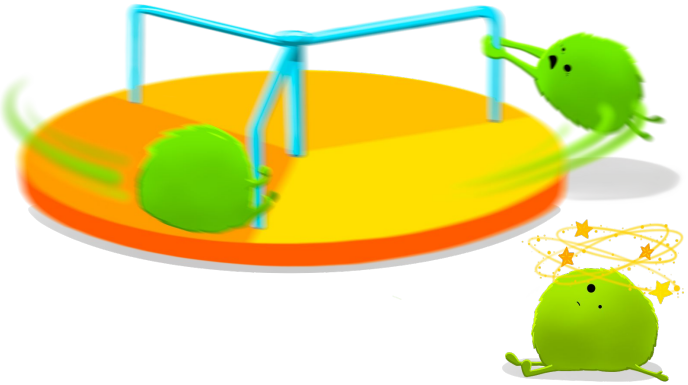
Debugging Bingo

Extra comma	Misspelled variable	Confused factor variable with a numeric one	Extra quotation marks	Put code into a markdown cell
Missing a tilda	Extra parentheses	Used wrong case (upper vs. lower case)	Misspelled function name	Didn't import the data
Confused when using $	Wrong argument(s) to a function	FREE SPACE	Confused a data frame for a variable	Missing pipe operator (%>%)
Didn't load libraries	Missed a comma	Forgot to ask R to print the object	Didn't close parentheses	Misspelled data frame
Confused = with ==	Used color when you meant fill	Forgot to save	Put regular text into a code cell	Missing quotation marks

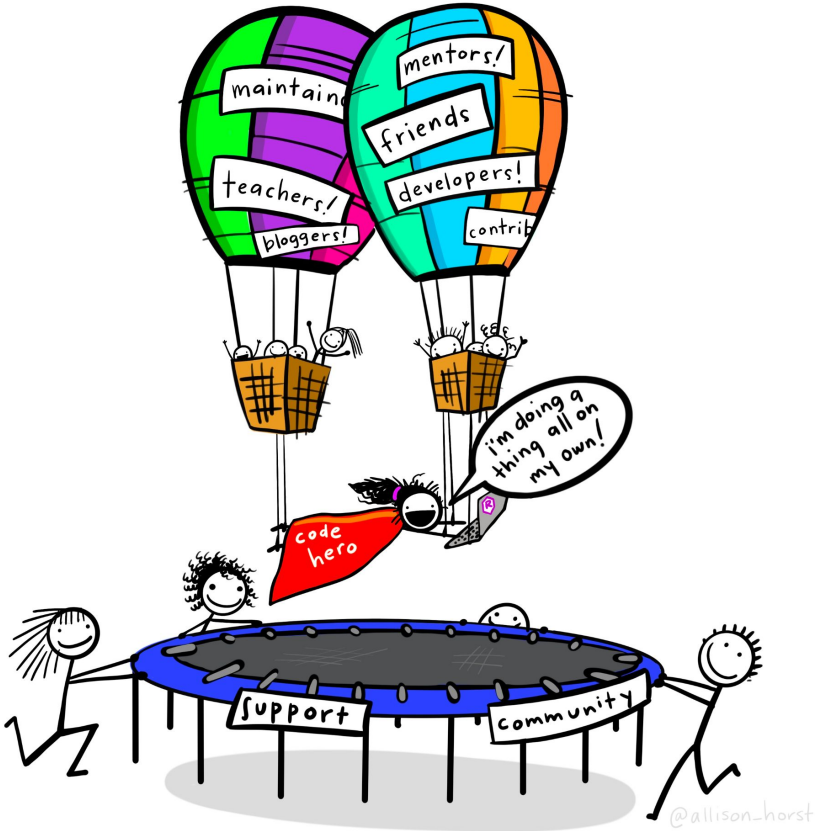
https://twitter.com/cogscimom/status/1354508785365078016

Debugging your code

- Comment things out
- Print things out (contents of variables, variable dimensions, variable types/classes)
- Print a file that contains the output of a code block
- Add conditional statements based on expectations
- Run loops just once (or just a few times)
- Terminate loops when something unexpected happens



Getting help with your code



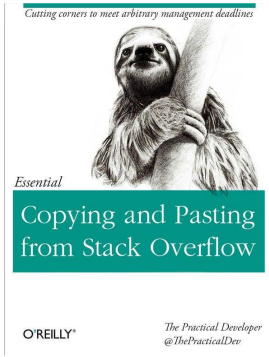
Getting help with your code: just search

- Search error messages *as-is*
- "How to ..."
  - Add "R" or "Python"
  - Add the names of the function or package
- Try variations/phrasings
- Learn to adapt solutions to related problems

Google

So many excellent blog posts & discussion forums!

stackoverflow



Testimonial from an undergrad!

"Arjun, I thought I was good at google'ing and finding stuff but you're much better than me!"

Getting help with your code: reproducible examples

Your code must be easily reproducible for someone else on their computer.

- If they cannot reproduce your error, they may not be able to help.
- If you do not provide an example there is nothing for others to play with to figure out your problem.

stackoverflow

Getting help with your code: reproducible examples

Capture everything

- Include code you tried
- Any library/package calls
- Create all necessary objects (such as workable data)

The reproducible example needs to be complete but minimal.

- Strip away everything that is not directly related to your problem.
- This usually involves creating a much smaller and simpler dataset than the one you're facing in real life.

stackoverflow

Getting help with your code: reproducible examples

It's worth the work!

- Often, you solve your problem in the process of creating a reproducible example.
- If you do not solve your problem, then you've made it as easy as possible for someone else to help you from their computer (increasing the chances you get help!)

stackoverflow

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