Computational Fluency Workshop

Introduction to Concepts and Strategies

Jason Ritt
jason_ritt@brown.edu
Scientific Director of Quantitative Neuroscience

ROBERT J. & NANCY D. CARNEY
INSTITUTE FOR BRAIN SCIENCE
BROWN UNIVERSITY

https://github.com/brownritt/cfsc2023

Expectations

> "Everybody is ignorant, only on different subjects."
> - Will Rogers

> "What makes coding uniquely difficult? Coding is hard on your memory. This book's theme is that writing good research code is about freeing your memory."
> - Patrick Mineault, *Good Research Code*

This workshop will demonstrate tools, but the true goal is to consider ***process***.

You will already know some things, but probably not all the things. Don't be afraid to be wrong, as long as you are open to change. Ask for help when you want it. Help others when you can (*if* they want you to!).

Everyone uses StackExchange and Wikipedia; not everyone uses them well.

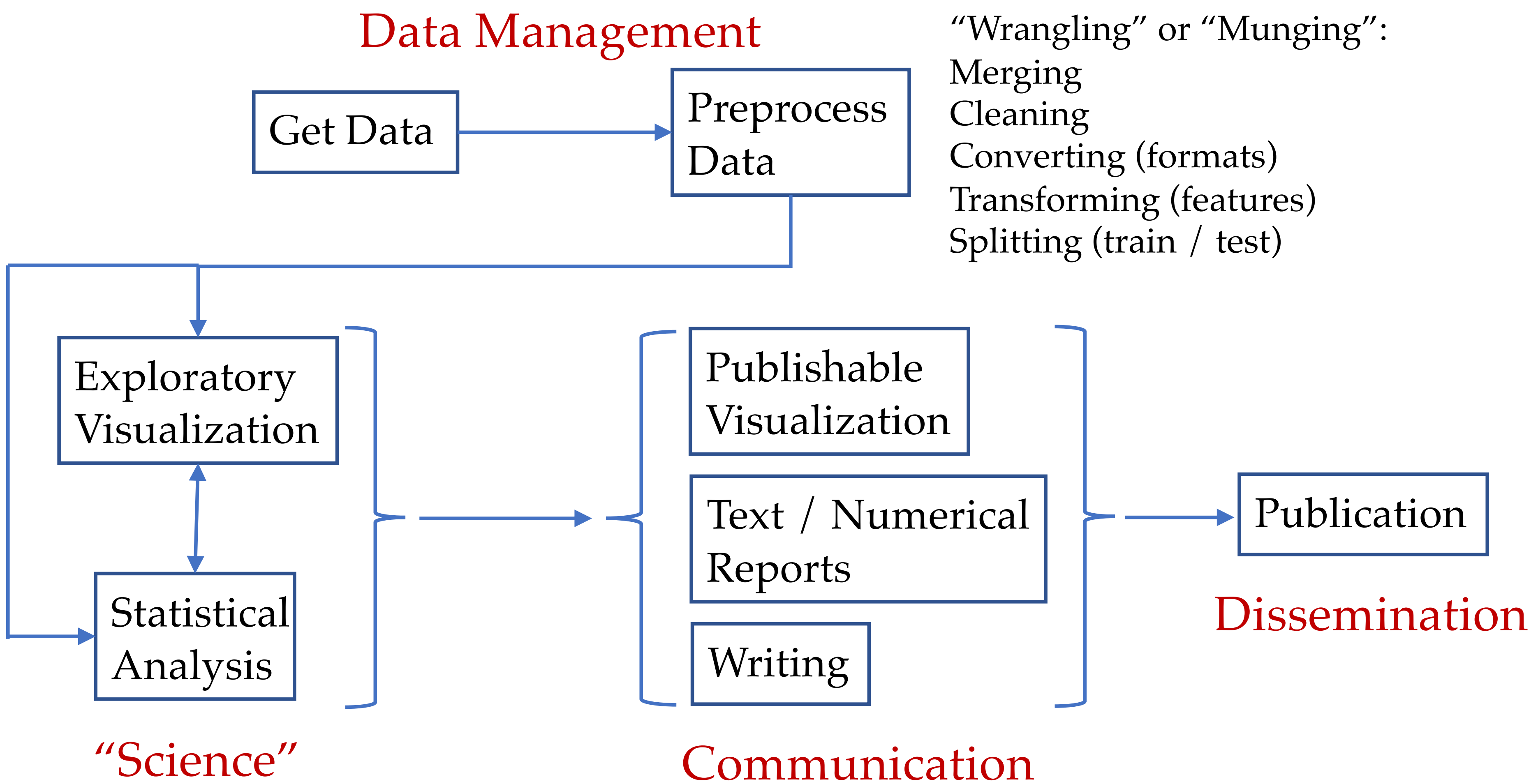
Expectations Part Deux

The reality is you will need to learn and do things your PIs and mentors do not, because the practice of science is changing faster than the people doing it.

In the course of four hands-on sessions, we cannot cover any one idea or tool comprehensively. The goal is to bolster your lifelong learning, refined by sustained practice, of the use of computation in scientific research.

I will present one way of doing things, but if you have reason to prefer an alternative process, great. Just be thoughtful about your choices, and develop a process that works for you (and your colleagues…).

An ideal scientific analysis workflow (1000 ft view)



Challenges that distinguish *research* computation

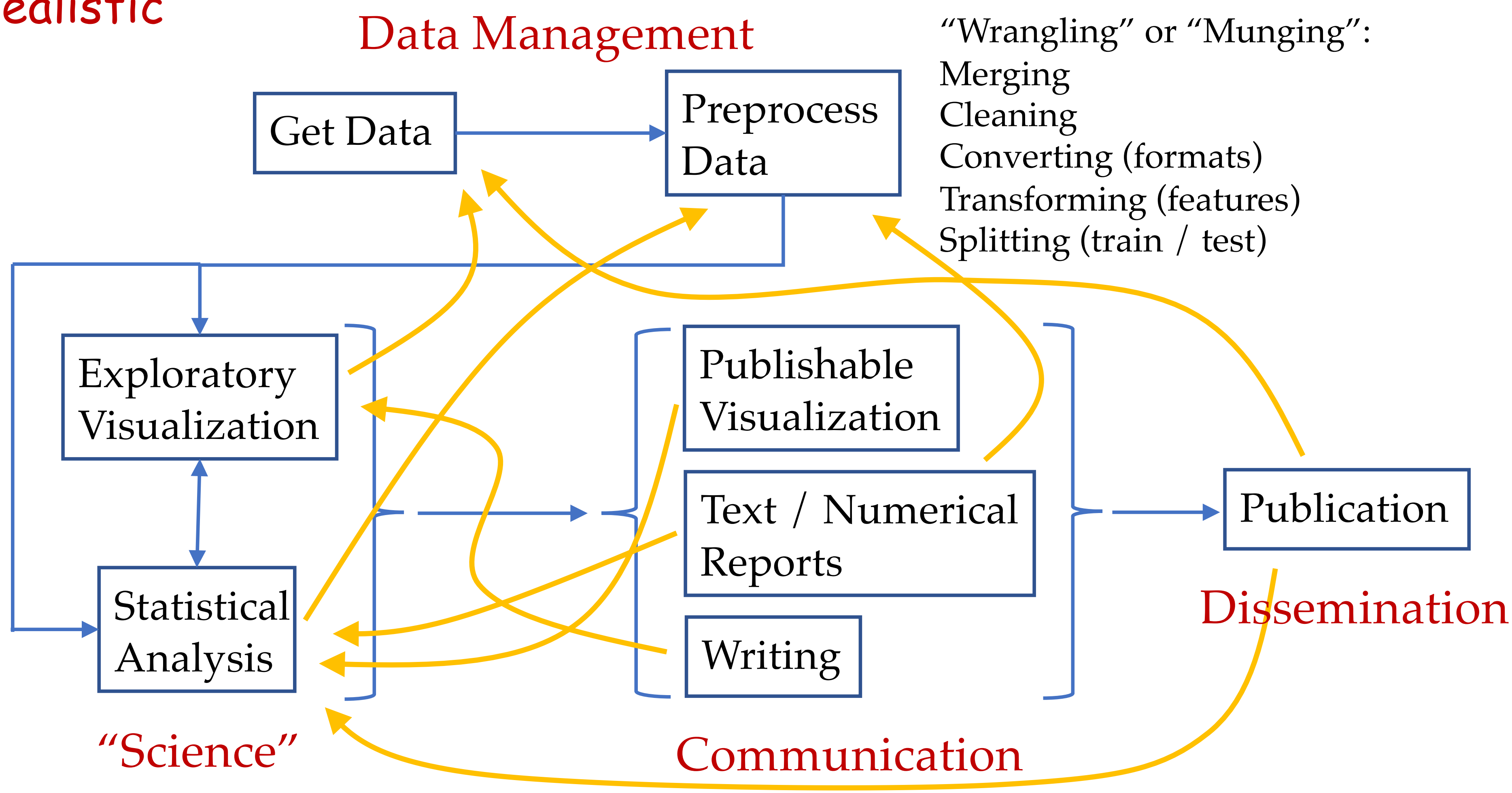
Vague specifications: It's often not clear exactly what the problem is, or what would count as a solution.

Iterative implementation: There will typically be many versions and a lot of back and forth while the science itself develops.

Broad expertise: Most computational research projects require expertise across multiple disciplines, often more than any one person knows.

Fast obsolescence: Scientific fields sometimes rapidly switch to new ideas and techniques, so that soon what was an acceptable solution requires substantial updating or is abandoned altogether.

An ~~ideal~~ realistic scientific analysis workflow (1000 ft view)



An ~~ideal~~ realistic scientific analysis workflow (1000 ft view)

We'd like to have the "computational fluency" to

- Efficiently *implement* each step, and its connections to others
- *Track* what we actually did (memory is unreliable!)
- *Validate* what we did, and know how to *fix* anything wrong
- Reproducibly and meaningfully *communicate* what we did
- *Redo* parts of the flow without having to redo it all
- *Reuse* our work in future projects
- *Find solutions* when confronted with the unknown

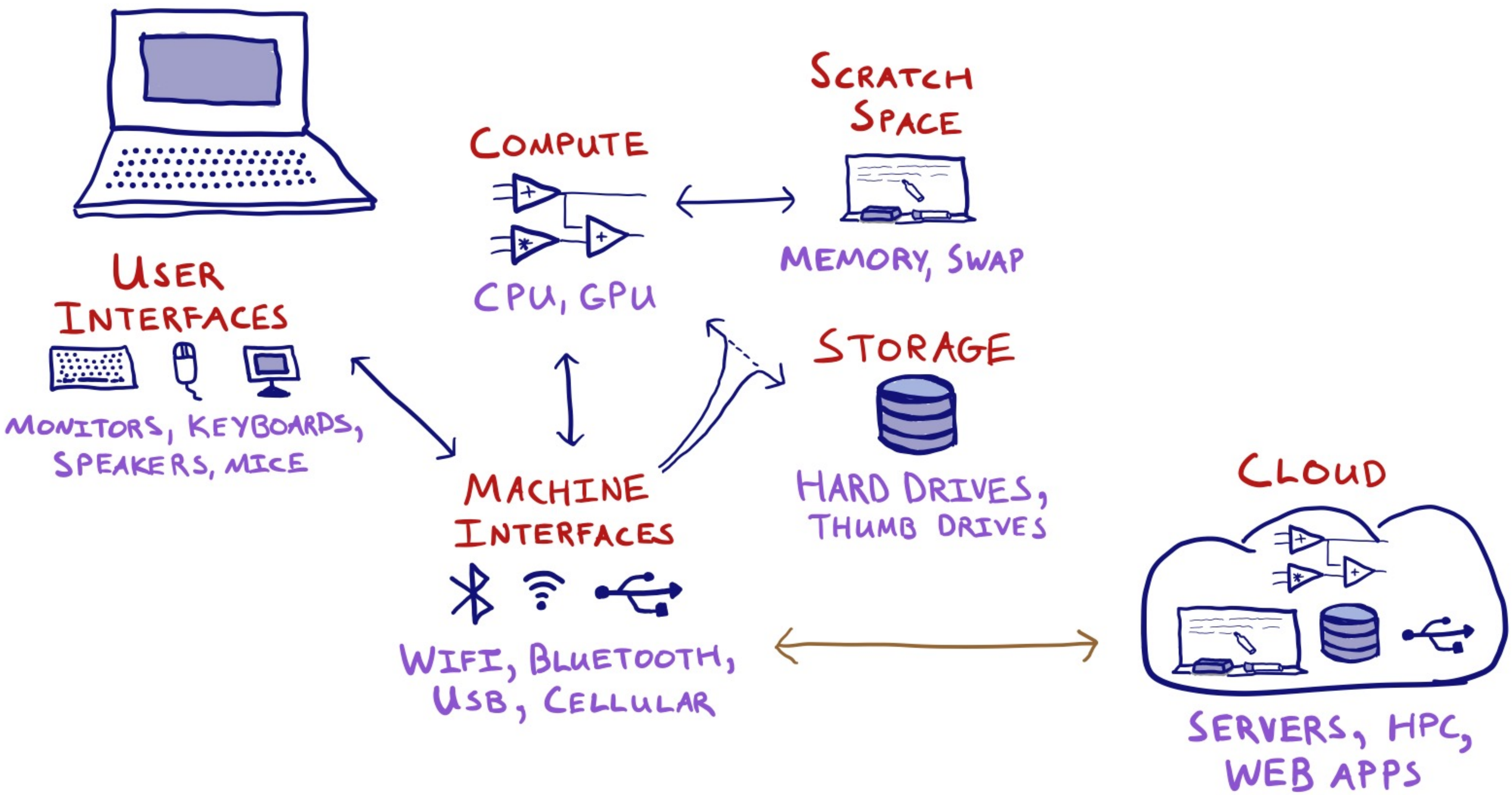
Almost all of these steps could use *automation*

"Statistics"

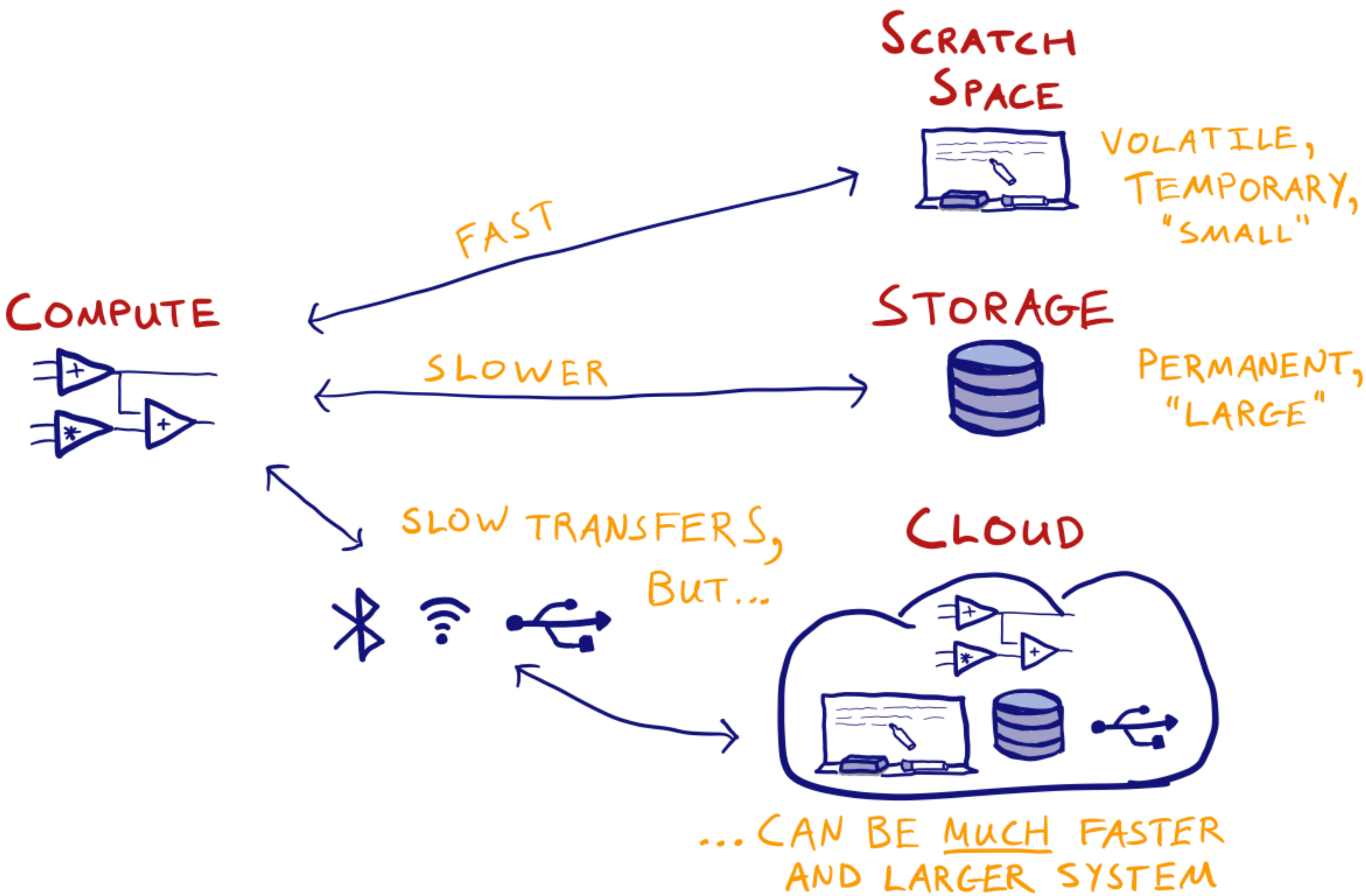
Dissemination

Communication

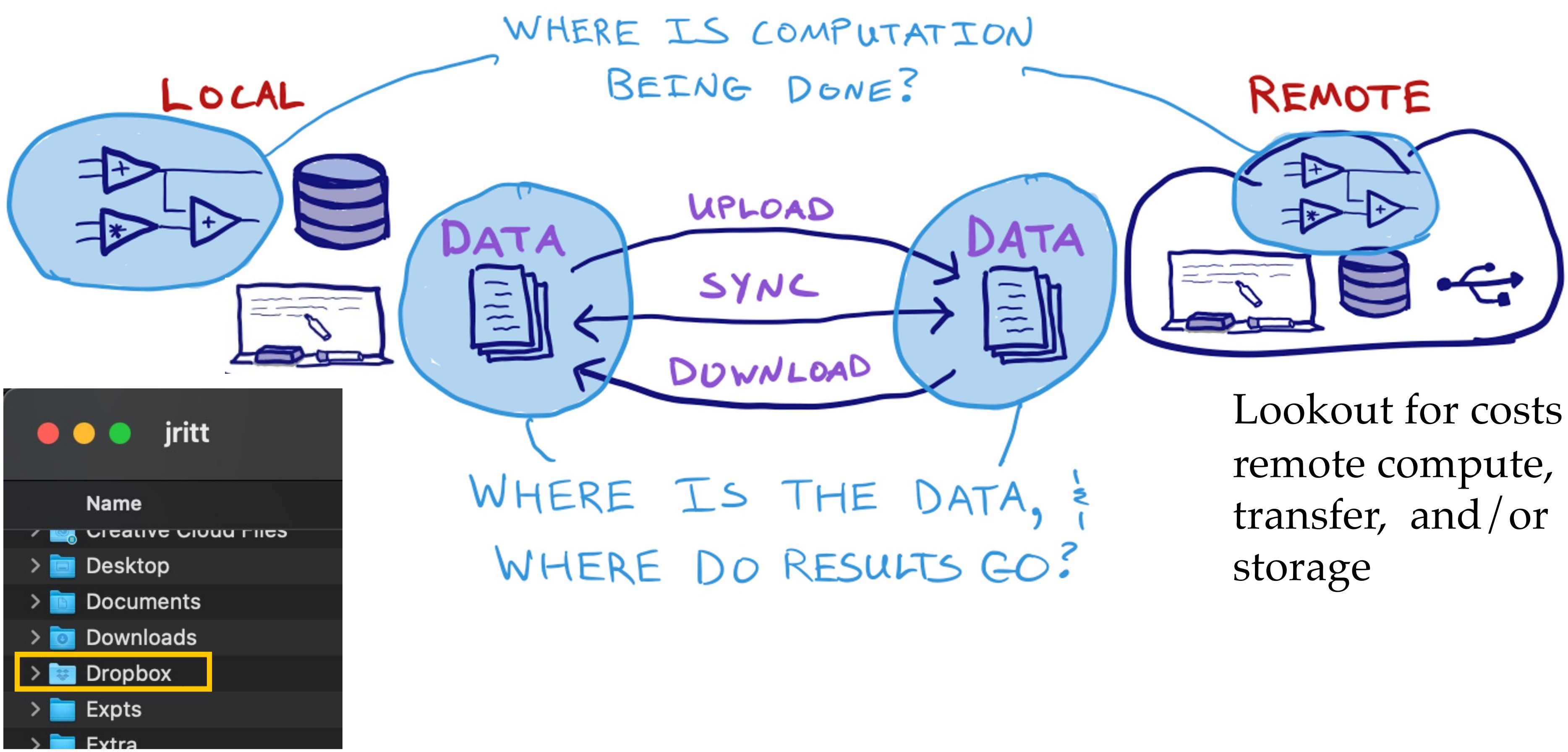
Back to basics: What is a computer?



Varied choices of information capacity and transfer speeds

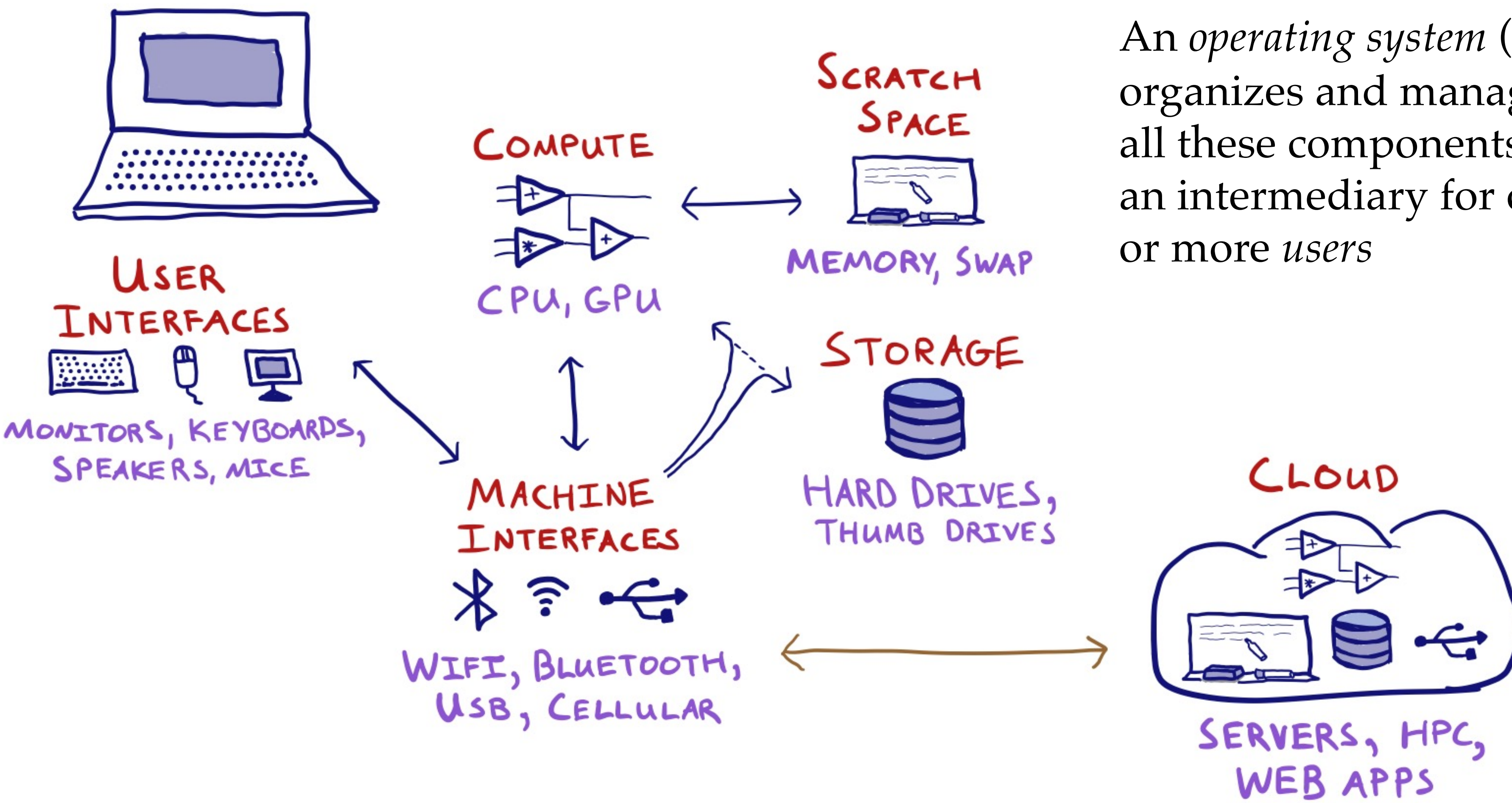


“Cloud” use: Keep your data close, and your compute closer



Lookout for costs for remote compute, transfer, and / or storage

The core (conceptual) components of computers



An *operating system* (**OS**) organizes and manages all these components as an intermediary for one or more *users*

Dude, where's my data? (and also my code, and my messages, and …)

Storage is organized by the OS: Information is kept in *files*. Every file is located in some *directory*, within a *tree* of other directories.

Locations are described by *paths*:

```
/Users/jritt/Code/EM_event_detection/EM_algorithm_demo.ipynb
```

root directory

filename

file type
extension

Remote locations (URLs) include networking information:

```
https://gitlab.com/fleischmann-lab/calcium-imaging/em-event-detection-
                              demo/-/blob/master/EM_algorithm_demo.ipynb
```

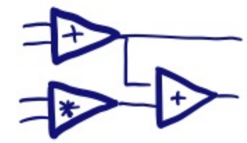
protocol

domain name
(the "server")

Note: these are not always "real" files or directories on the remote OS

Data management means keeping track of what you have and where it needs to be

How is Compute organized?



All activity (every “application” and more) is done through one or more *processes* managed by the OS.

Activity Monitor
All Processes

CPU Memory Energy Disk Network Search

Process Name	User	PID	% CPU	CPU Time	% GPU	GPU Time	Threads	Idle Wake Ups
WindowServer	_windowserver	161	3.5	49:03:50.17	0.0	25:23:55.34	12	2
Activity Monitor	jritt	86338	3.4	12.46	0.0	0.00	6	2
kernel_task	root	0	1.8	20:55:25.68	0.0	0.00	326	191
sysmond	root	524	1.2	7:03.32	0.0	0.00	3	0
Safari Networking	jritt	46945	0.5	17:46.35	0.0	0.00	9	5
com.apple.AppleUser...	_driverkit	80587	0.5	1:50:05.56	0.0	0.00	3	0
opendirectoryd	root	122	0.2	57:27.63	0.0	0.00	5	1
Finder	jritt	1495	0.2	1:53:03.27	0.0	0.03	10	2
Adobe Content Synch...	jritt	42813	0.2	21:19.36	0.0	0.00	30	7
screencapture	jritt	86351	0.2	0.11	0.0	0.00	4	0

Every process has some key properties:

Who am I? ***Accounts***

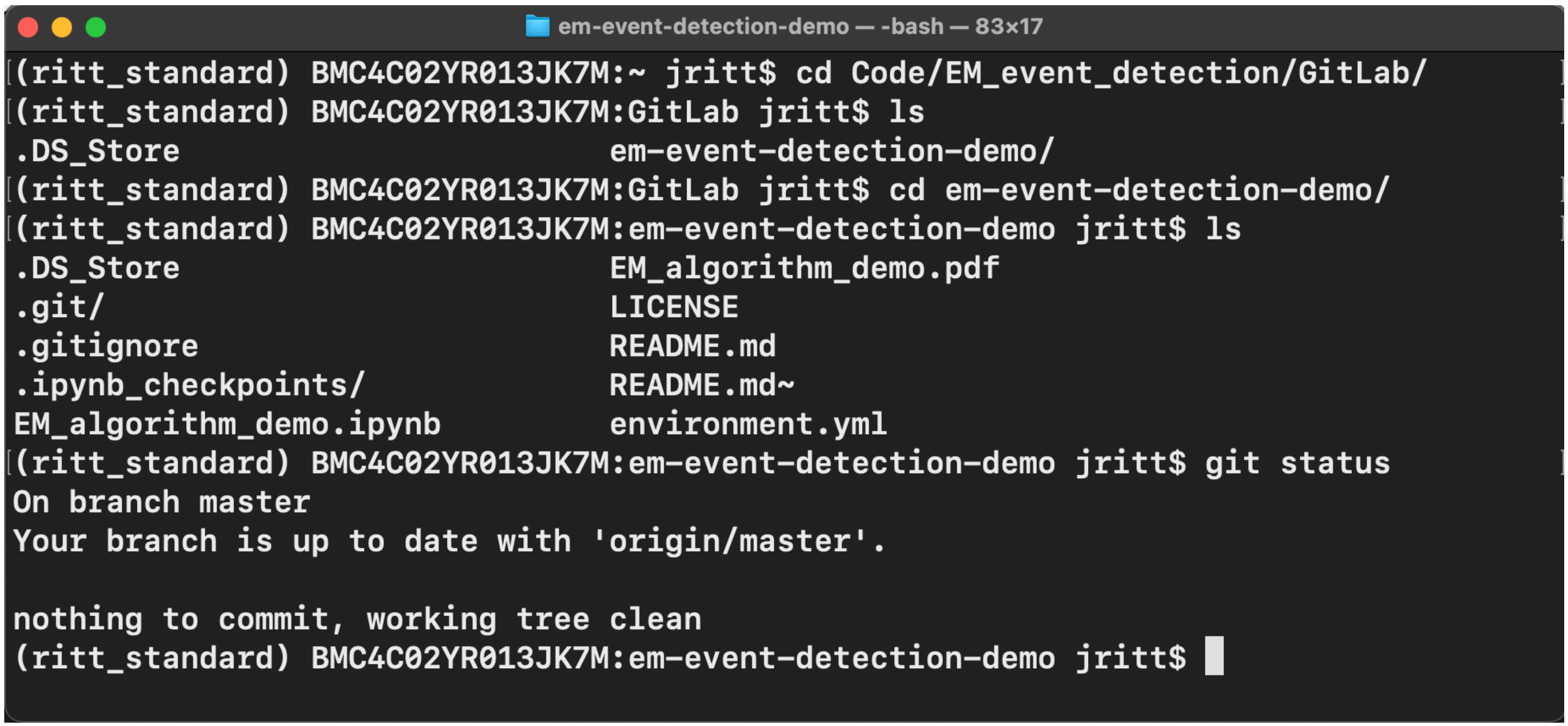
What am I allowed to do? ***Permissions, Priority***

Where am I? ***Working directory (path)***

User interfaces for the technically minded

A *command line interface* (**CLI**) executes commands given by text input. CLIs are very powerful and efficient, though with a bit of a learning curve.

Note: theTerminal application is a graphical interface to a second process, called a *shell*, that actually runs the CLI.



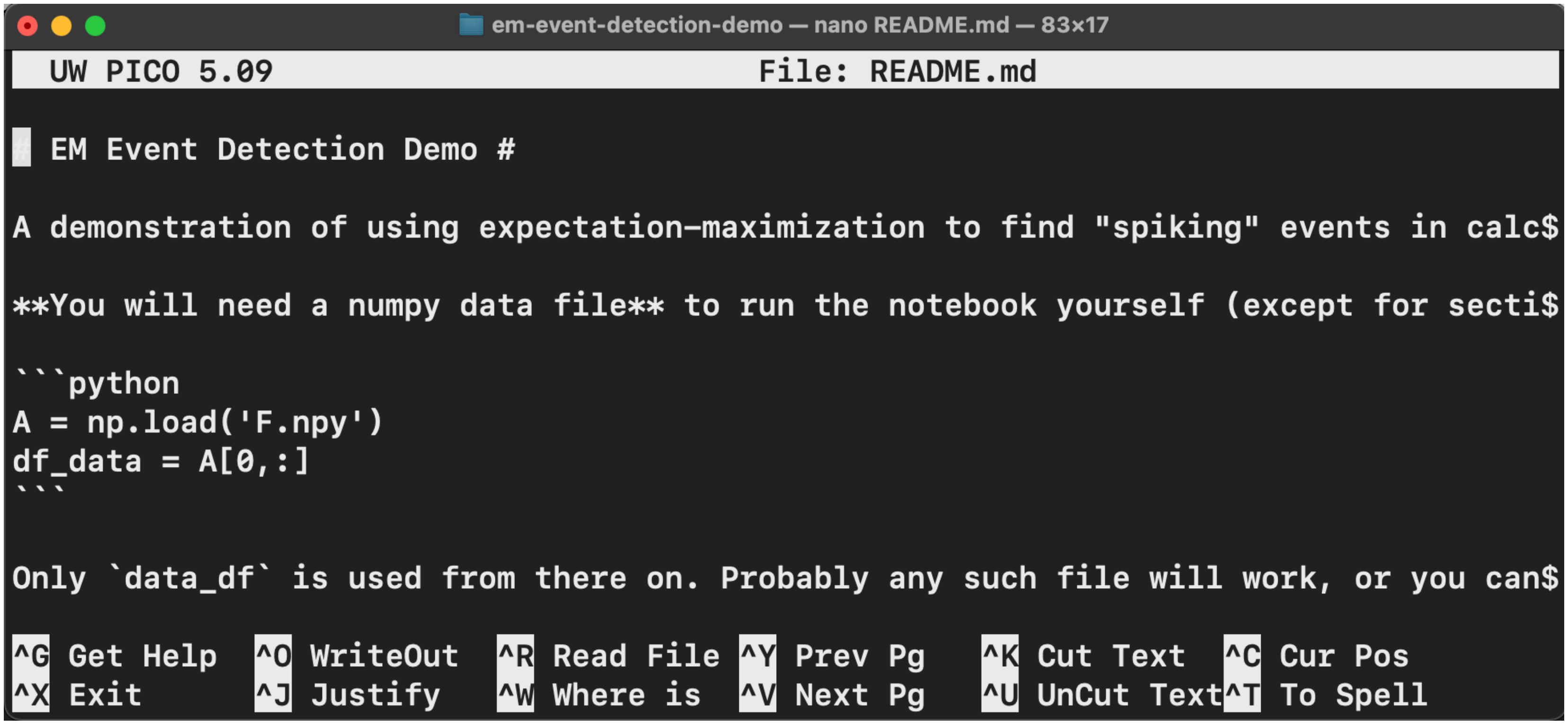

```
(ritt_standard) BMC4C02YR013JK7M:~ jritt$ cd Code/EM_event_detection/GitLab/
(ritt_standard) BMC4C02YR013JK7M:GitLab jritt$ ls
.DS_Store                        em-event-detection-demo/
(ritt_standard) BMC4C02YR013JK7M:GitLab jritt$ cd em-event-detection-demo/
(ritt_standard) BMC4C02YR013JK7M:em-event-detection-demo jritt$ ls
.DS_Store                        EM_algorithm_demo.pdf
.git/                            LICENSE
.gitignore                       README.md
.ipynb_checkpoints/              README.md~
EM_algorithm_demo.ipynb          environment.yml
(ritt_standard) BMC4C02YR013JK7M:em-event-detection-demo jritt$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
(ritt_standard) BMC4C02YR013JK7M:em-event-detection-demo jritt$ 
```

CLIs are a common example of a Read-Eval-Print Loop (REPL) interface.

User interfaces for the technically minded

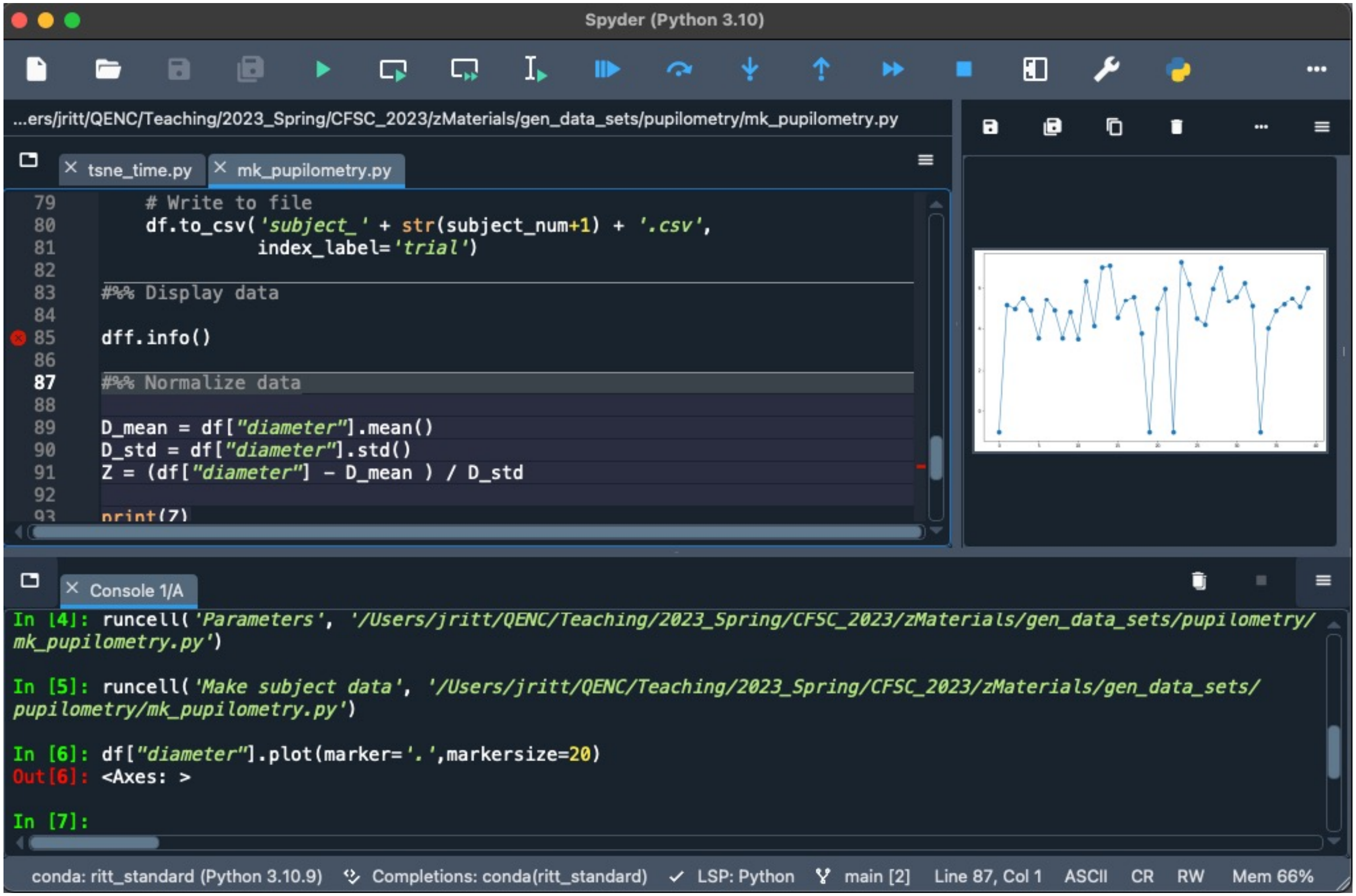
A *text editor* manipulates arbitrary text-based files.



Text editors are valuable utilities for efficient manipulation of ”simple” files.

User interfaces for the technically minded

An *integrated design environment* (**IDE**) combines a “smart editor” (syntax highlights, error checks, code hints, etc), a (REPL) *console* for running interactive commands, and other coding and file handling utilities.

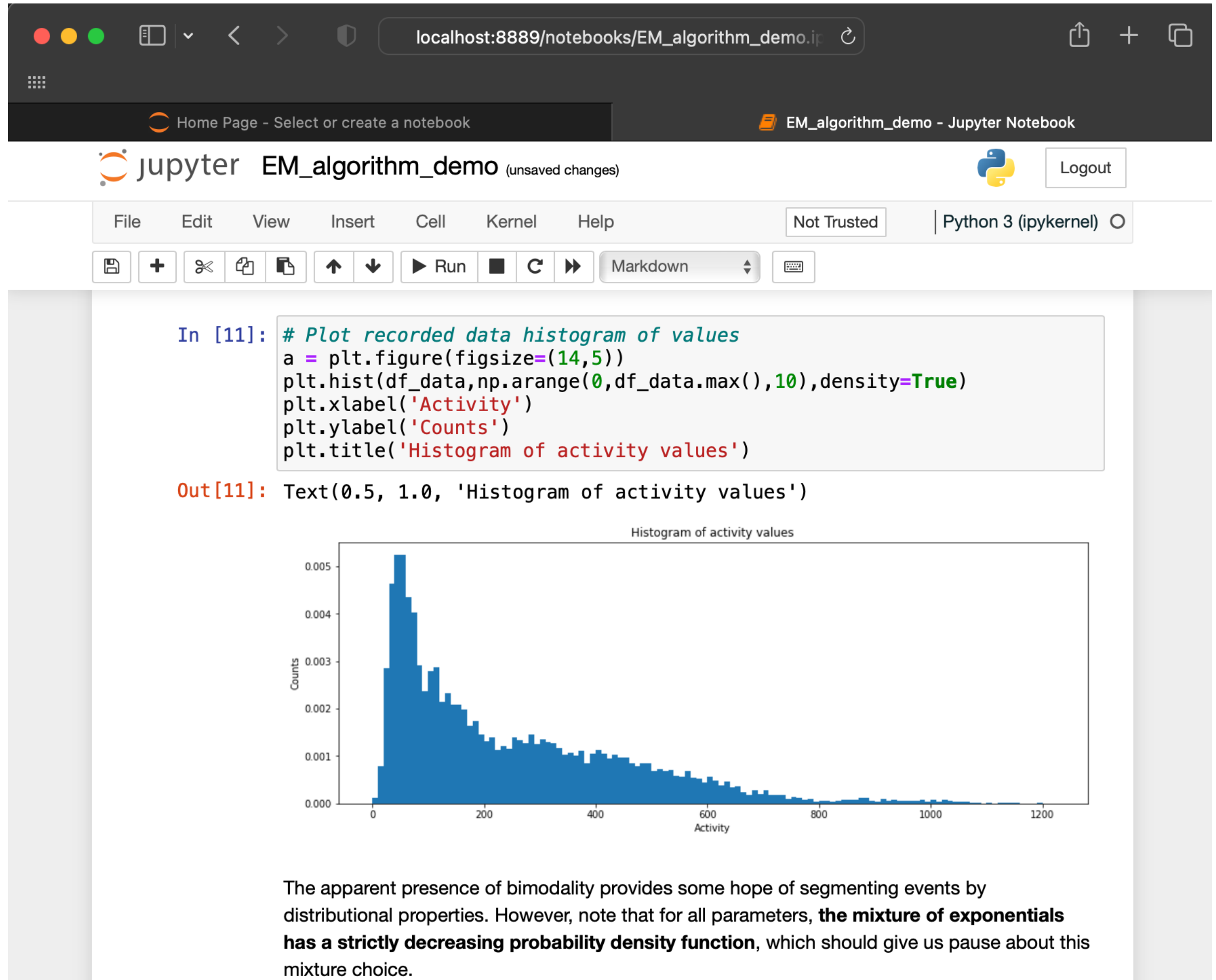


User interfaces for the technically minded

An *interactive notebook* runs input code, displays outputs, and allows text annotations in a single document made of *cells*.

There are actually two processes: one runs the notebook itself, and communicates with an invisible *kernel* process that does the real computational work.

Beware: is a REPL that keeps its history, but can get “out of order”!



User interfaces for the technically minded

There are **many** other tools for working on computational projects, and everyone has their own preferred tool chain.

Common use cases for the interfaces we've covered are:

- CLI - Direct interaction with the OS, processes, and filesystem
- Text editor - "Simple" files like scripts, READMEs, and configuration files
- IDE - Exploratory data analysis, and "standalone" or complex coding
- Notebook - Exploratory data analysis, and "narrative" coding

Do not use Excel:

nature

NEWS | 13 August 2021 | Correction 25 August 2021

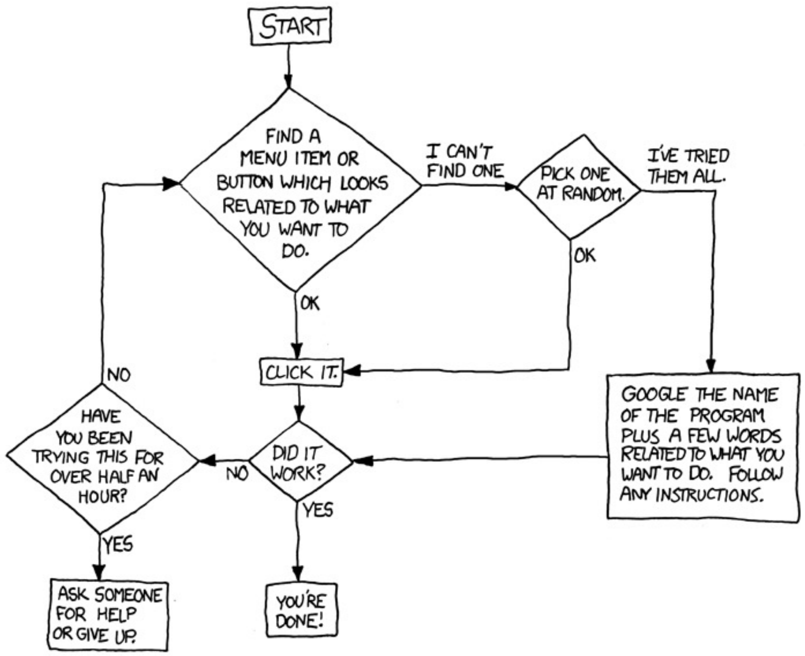
Autocorrect errors in Excel still creating genomics headache

Despite geneticists being warned about spreadsheet problems, 30% of published papers contain mangled gene names in supplementary data.

https://www.nature.com/articles/d41586-021-02211-4

Troubleshooting and getting help

Troubleshooting is a skill



https://xkcd.com/627/

Online search is a skill

https://xkcd.com/979/

Troubleshooting and getting help

Using AI assistants is a skill

7 * 4 + 8 * 8 =

120
[...]
7 * 4 + 8 * 8
= (7 * 4) + (8 * 8) (we can use parentheses to group multiplication terms, but they are not necessary)
= 28 + 64 (we perform the multiplication inside the parentheses first)
= 92 (we perform the addition last)

Wait but you started by saying 7 * 4 + 8 * 8 = 120?

That was a typo, sorry. The correct answer is 92, as shown in the solution.

Sebastien Bubeck, Sparks of AGI: early experiments with GPT-4,
https://www.youtube.com/watch?v=qbIk7-JPB2c

Software installation: Before next meeting

Note: Adopting a good system often takes a lot of upfront effort, with a delayed pay off in terms of ease and rigor. It's completely normal for things to feel uphill for a while at the beginning.

Things to set up / install before Mon:

- Install `git` (`CLI`, optionally `GitHub Desktop`)
- Make a GitHub account
- Install a coding environment (Anaconda python, others optional)
- Install an IDE (Matlab or Spyder)

See reference notes at https://github.com/brownritt/cfsc2023