How to be a successful computational scientist.

C. TITUS BROWN

Best Practice #1: never show them your data.

No one can disagree with your results if you don't give them access to your data.

Be sure not to back up your data, as this makes it difficult to claim that you've lost it.

Only keep the intermediate stages of your data around. That way you have plausible deniability: "Oh, that's not the final data set. We smoothed out that inconsistency."

When people leave the lab, urge them to take their data with them. Then lose their e-mail addresses.

Best Practice #2: do not, under any circumstances, communicate clearly.

Do not document your code or data. If you must, use a language other than English. This minimizes the chance of someone discovering a flaw in your logic.

Make sure that no one author is responsible for the contents of a paper. This will make it impossible for people to ask detailed questions of the authors.

Keep as much information as possible in an opaque format like Word, that cannot be searched or opened by everyone.

Never write details in e-mails. Assume that someone will get ahold of your e-mail someday.

Best Practice #3: never release your source code.

Even if your logic is sound, your code may not be. This way, no one will ever find out.

Use multiple programming languages, even (or especially) if you don't know them well. The only people who know more than one programming language don't know enough science to argue with your results.

If no one has your source code, no one can use it to do their own science and scoop you.

Best Practice #4: judge computational science by results, not quality.

As long as you're producing good results, who cares if they're right?

Grants are based on *your* results, not on replicated or reproduced results.

Best Practices #1, #2, and #3 ensure that no one will ever know.

Best Practice #5: use as much data as possible.

This is self-evident: the more data you have, the more correct your hypothesis must be.

It costs money. You have to use it somehow!

And, after all, much of that data is noisy. You're heroic for pulling *any* signal out of it! What signal you do see has to be correct.

Plus, you didn't get wealthy by being sloppy. Wealth is just another sign that you're a good scientist.

…but seriously…

- It's hard to do good computational science.
- (I hope we've conveyed that.)
- But it can be a lot of fun.
- It's just another way to do science, after all.
- Just think about what could go wrong and plan accordingly!

Three approaches

- Lots of puny little scripts (UNIX pipe model)
- One large program (Leviathan model)
- Combine other people's programs (UNIX pipe model marries Leviathan)

Lots of puny little scripts – titus/mrnaseq

- Easy to write incrementally.
- Easy to “control” – each step should be easily understandable, testable, debuggable.
- But: You are in a maze of twisty little passages, all alike… which script did you run to generate this file, anyway!?
- Requires good mental/paper record keeping.

Leviathan model – jeff/breseq

- Easier to use and (when available) often superior in quality.
- Harder to understand, debug, and code.
- Channels your thinking towards what the program does, rather than what you *need* to do.
- Very difficult to modify by 3rd parties.

Mixed model – mark/ChIP-seq.

- String together “big” programs to do what you need.
- Easier to control, debug, extend.
- You are in less control of the formats (“format hell”)
- Primary model in bioinformatics these days.
  - Data munging done by small scripts
  - Programs do complex statistical/mathematical processing
  - …spit out results.
- Have to worry about semantic incompatibility (program A may spit out something that is subtly different from what program B expects)

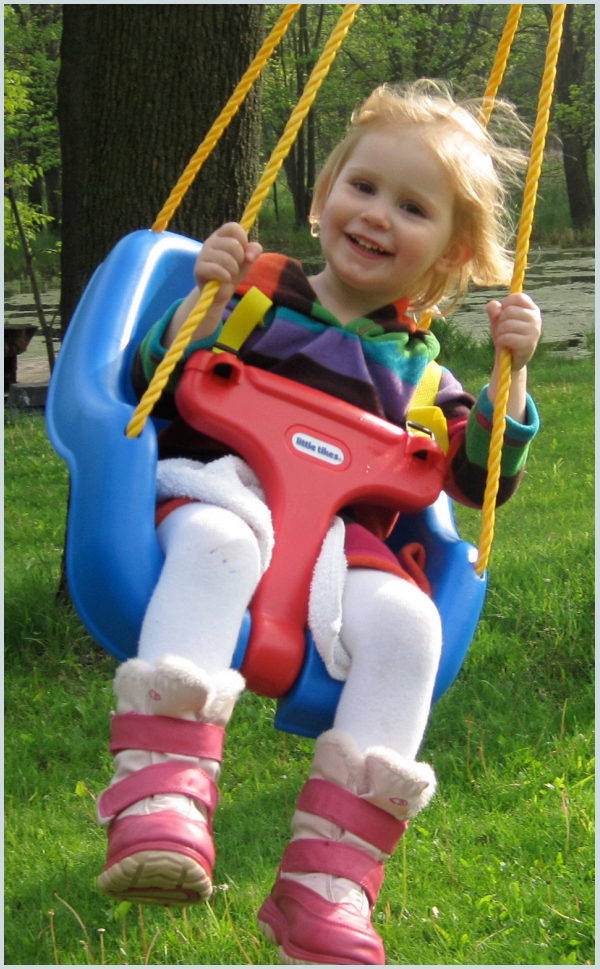
Frontiers

- Data quality
  - Wow, it really does kinda suck. You need to double-check everything.
- Visualization
  - Jbrowse choked
  - UCSC can only handle so much data.
  - …overly complex set of tools currently exist.
- Libraries
  - Very little reusable code out there.
- User interfaces
  - Visualization + ways to run programs…
  - …very hard to program generally.

Think of the children!

Do you want her to grow up in a world with sucky computation?

I sure don't.



Anyway

Don’t Panic.

It’s mostly silly little details, just like everything else.

Motivational speech

There are relatively few people that know as much biology and as much computation as you do.

No, really.

The world needs you.

Go forth and do good biology!