The (slightly delayed) Welcome lecture

Challenge #1:

Most biologists don't know much about computational science.

- Among many biologists, there is a general fear or skepticism of computers.

- This leads to shallow thinking about computational science.

Challenge #2:

Most computational scientists don't know much about biology.

- Extant computational solutions may not use appropriate heuristics, or default parameters.

- "It works on my data…", but their data != yours!

- Solutions/programs may not be couched in the right terms for the biology, or with proper appreciation for biological complexity.

Challenge #3:

Both biology and computational science are deep, complex fields of study, inhabited by extremely smart people!

- None of this is easy, on any side of things.
- If it were easy, they wouldn’t need people as smart as all of us to do it, right??
- A two week course can’t possible teach you everything.

Challenge #4:

Sequencing technology is changing very fast.

- We don't understand its limitations or biases very well.

- The software and compute infrastructure lags behind volume of data, type of data.

None of this is the #1 problem you will face with bioinformatics.

Here is the #1 problem:

How do you know if your computational answer is right or wrong?

None of this is the #1 problem you will face with bioinformatics.

Here is the #1 problem:

How do you know if your computational answer is right or wrong?

If you can't answer this question, then what's the point of doing the computation?

Controls

- Just as with experiments, you can put negative and positive controls in your bioinformatics.

- e.g. with BLAST,
  - Do you see expected matches with the parameters and database you're using?

- Positive controls are often easier than negative, in "discovery-driven" science…

Internal controls

- Molecules and sequences for which you have expectations.

- “I know this gene comes up, based on qPCR. I expect to see it in my mRNAseq.”

External controls

- Does the whole process work?

- “I can reproduce what this other person/lab did, with their data, when I use my own software.”

- This is much more rarely done…

Black box nature of algorithms

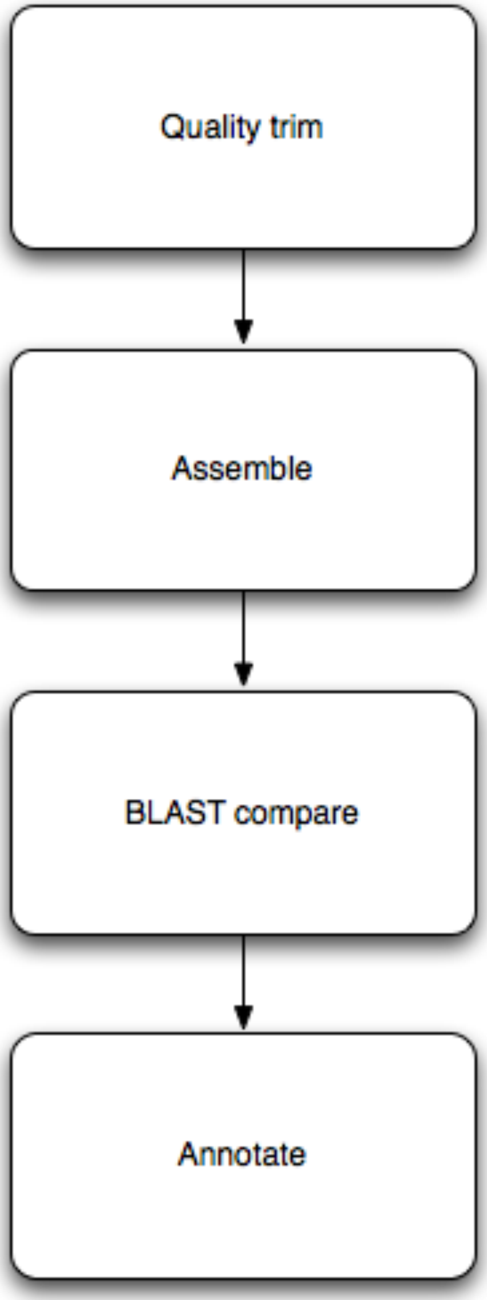
- When you listen to a computational biologist explain their clever algorithm…
- …it's a mistake to think that they necessarily know what's going on.
- Software is full of bugs and unintended consequences.

Pipelines

- Each step can be understood, and tested/ controlled individually.

- Each step is re-usable! Just need to figure out input/output formats.

- Automate, automate, automate.



The opportunity:

- The sequence is here.

- “In the land of the blind, the one eyed is king.” -- those prepared to *think* about how to use sequencing technology to answer their question will have a substantial leg up.

- Who knows? Some of you might even like this mix!

Our goals

- Provide a “safe place” to experiment.
- Lots and lots of help (in the form of TAs)
- Provide lots of data sets, tools, scripts.
- Research specific help?

Our requirements of you

- Nothing.

- This is a requirements free zone.

- You can safely skip the entire course…

Our expectations

- Questions!
- Ask for help when you need it!
- A certain amount of tolerance may be needed, by you of us…

Our hopes

- Enthusiasm!
- Engagement!

Daily schedule (tentative)

- 9am – lecture
- 10:45am – tutorial 1
- 12-1pm - lunch
- 1:15pm – tutorial 2
- 3pm – free time!
- 5-6:30 - dinner
- 7pm – tutorial/lecture/etc

Weekly schedule – tentative wk1

- Tuesday – BLAST, cloud computing, scripting
- Wed – mapping
- Thursday – assembly
- Friday & Saturday – tools, stats, and pipelines

Dramatis personae

- Titus Brown (that's me)
- Ian Dworkin -- co-instructor
- Istvan Albert – co-instructor

- Adina Howe – Titus' body double
- Likit Preeyanon – TA
- Tristan De Buysscher– TA
- Amanda Charbonneau –TA
- Elijah Lowe – TA
- Michael Crusoe – TA and go-fer

- Other course faculty: Rayan Chaikhi, Chris Chandler, and Erich Schwarz.

Written rules

- No night-swimming without a buddy.

Unwritten rules

Framing the approach

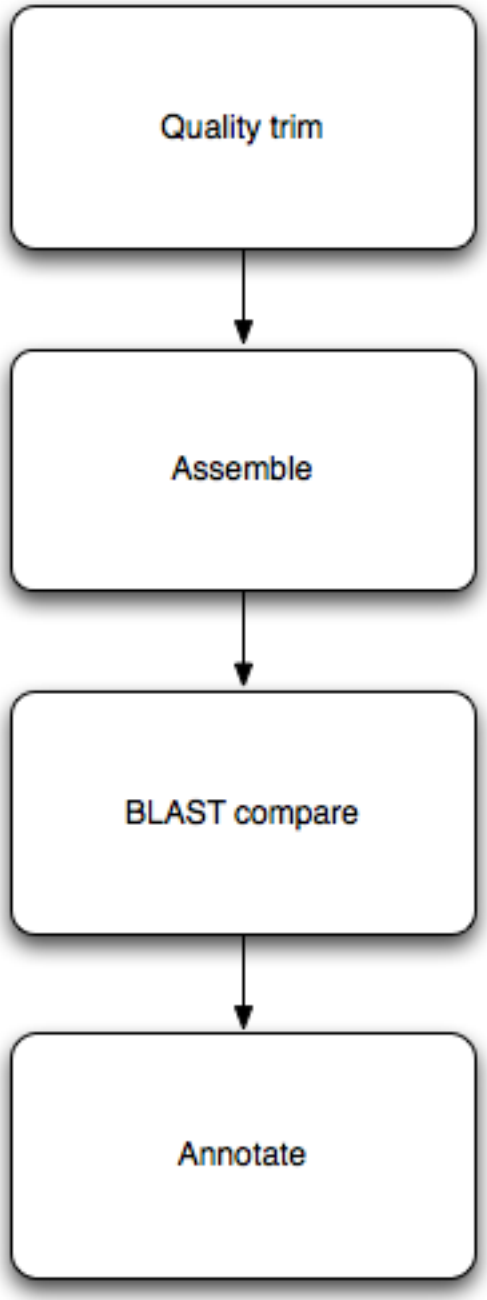
1. How does all this stuff work, generally?

2. Can we automate things and/or do them more efficiently?

How does this stuff work?

- Typically, you need to run multiple different programs in sequence.
- Each program takes in data, in files; and outputs data, in files.

- (Some programs also produce pretty pictures via the Web.)



Why automation & computational efficiency matter

- You'll learn to run lots of different programs here.

- We'll run into some practical problems:
  - Some programs take a long time to run.
  - Some programs take many different parameters; which are best?
  - Some programs don't finish on "cheap" hardware.

How do we run many long-running programs? How do we remember what we did? How do we get our programs to finish?

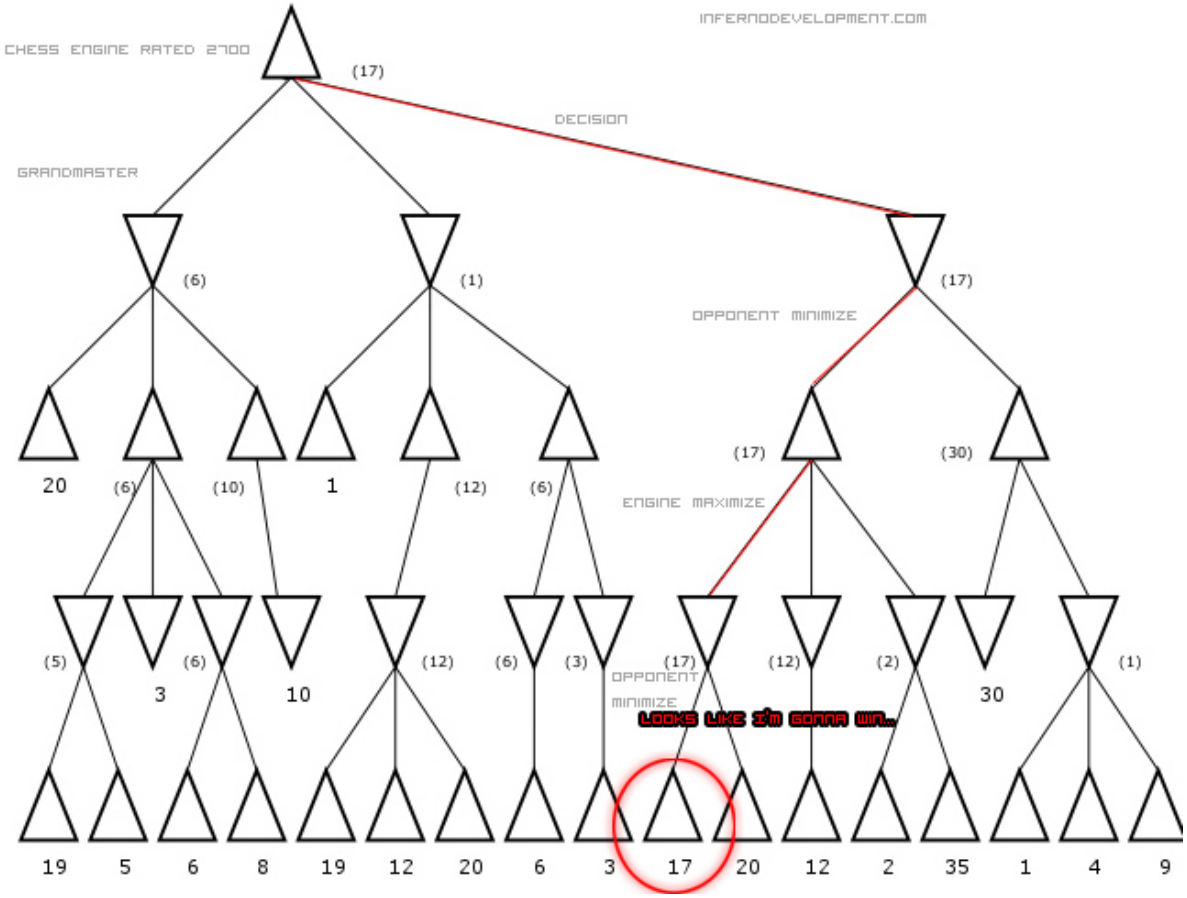
“Heuristics”

- What do computers do when the answer is either really, really hard to compute exactly, or actually impossible?

- They approximate! Or guess!

- The term “heuristic” refers to a guess, or shortcut procedure, that usually returns a pretty good answer.

Often explicit or implicit tradeoffs between compute “amount” and quality of result



http://www.infernodevelopment.com/how-computer-chess-engines-think-minimax-tree

This kind of issue (heuristics; time to compute => quality) comes up a lot.

- Mapping.
- Assembly.
- Statistics (Monte Carlo and resampling methods).
- Simulations.

- More generally, most “interesting” algorithms involve approximations and shortcuts. When are they (in)appropriate for your task?

What are the limits of data + compute?

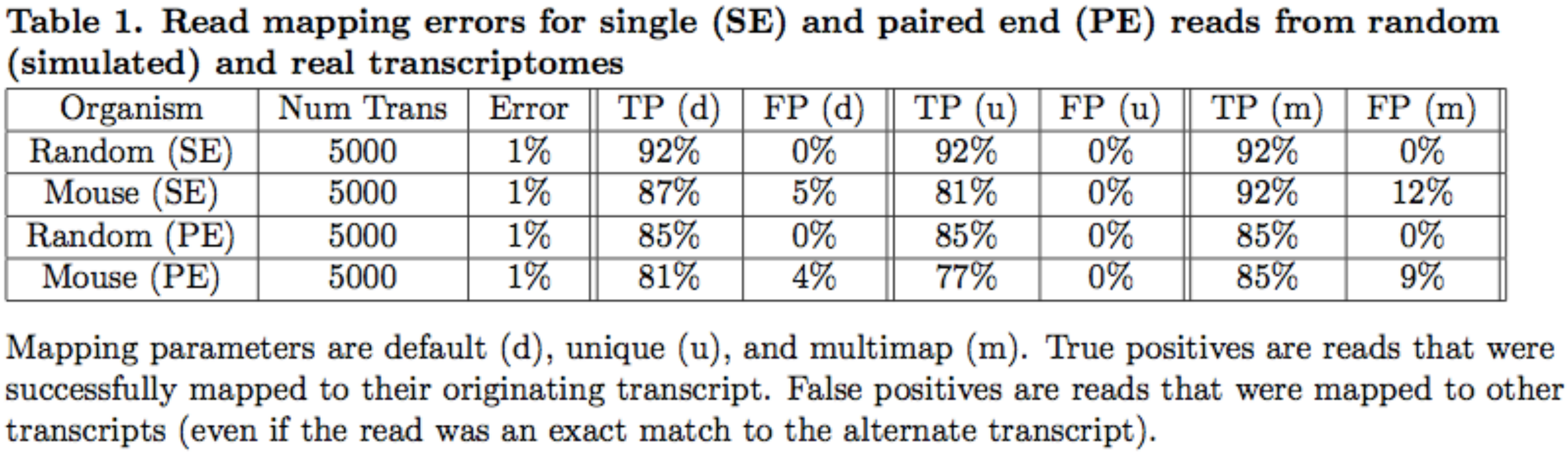


Table 1. Read mapping errors for single (SE) and paired end (PE) reads from random (simulated) and real transcriptomes

Organism	Num Trans	Error	TP (d)	FP (d)	TP (u)	FP (u)	TP (m)	FP (m)
Random (SE)	5000	1%	92%	0%	92%	0%	92%	0%
Mouse (SE)	5000	1%	87%	5%	81%	0%	92%	12%
Random (PE)	5000	1%	85%	0%	85%	0%	85%	0%
Mouse (PE)	5000	1%	81%	4%	77%	0%	85%	9%

Mapping parameters are default (d), unique (u), and multimap (m). True positives are reads that were successfully mapped to their originating transcript. False positives are reads that were mapped to other transcripts (even if the read was an exact match to the alternate transcript).

Mappers will ignore some fraction of reads due to errors.

Pyrkosz et al., unpub.

Does choice of mapper matter?

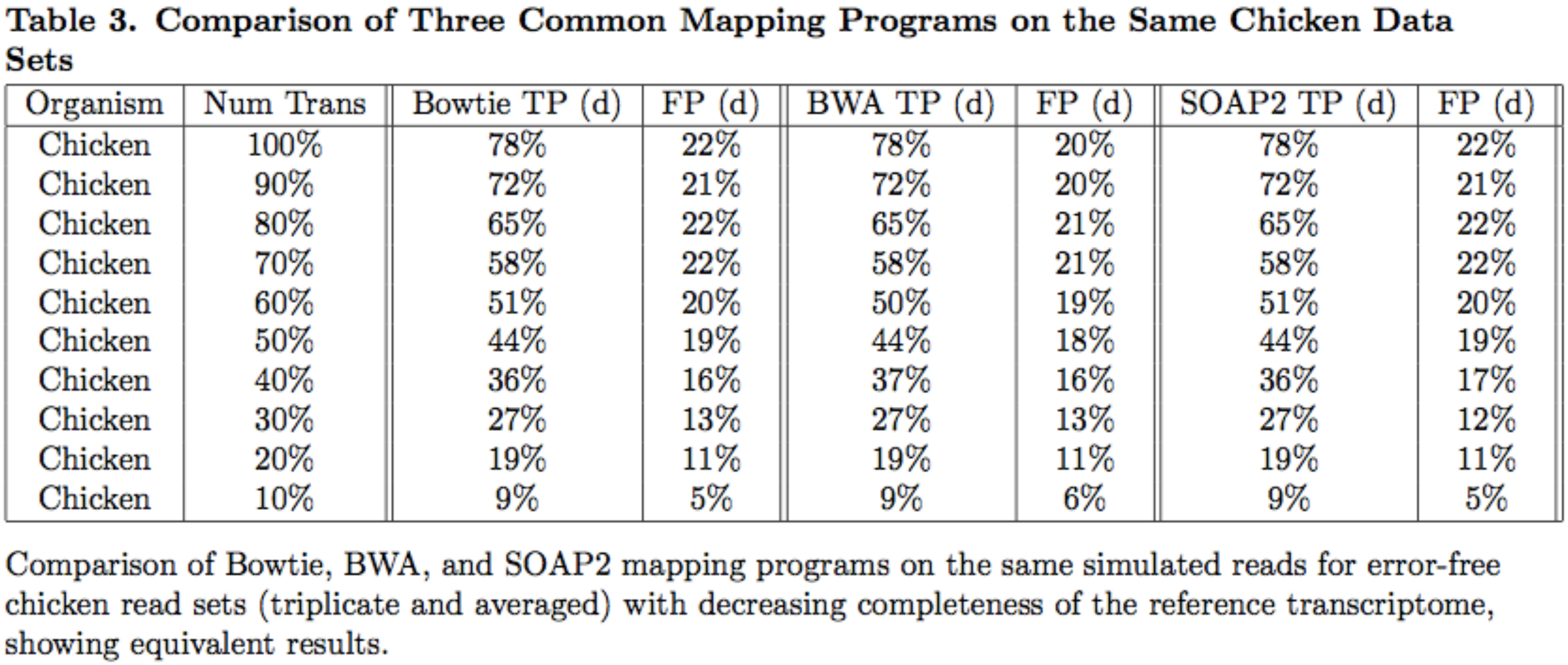


Table 3. Comparison of Three Common Mapping Programs on the Same Chicken Data Sets

Organism	Num Trans	Bowtie TP (d)	FP (d)	BWA TP (d)	FP (d)	SOAP2 TP (d)	FP (d)
Chicken	100%	78%	22%	78%	20%	78%	22%
Chicken	90%	72%	21%	72%	20%	72%	21%
Chicken	80%	65%	22%	65%	21%	65%	22%
Chicken	70%	58%	22%	58%	21%	58%	22%
Chicken	60%	51%	20%	50%	19%	51%	20%
Chicken	50%	44%	19%	44%	18%	44%	19%
Chicken	40%	36%	16%	37%	16%	36%	17%
Chicken	30%	27%	13%	27%	13%	27%	12%
Chicken	20%	19%	11%	19%	11%	19%	11%
Chicken	10%	9%	5%	9%	6%	9%	5%

Comparison of Bowtie, BWA, and SOAP2 mapping programs on the same simulated reads for error-free chicken read sets (triplicate and averaged) with decreasing completeness of the reference transcriptome, showing equivalent results.

Reference completeness matters more!

Pyrkosz et al., unpub.

Real problem? Our data can’t uniquely specify solution!

Here, there is no direct way to know if last exon is connected to first exon.

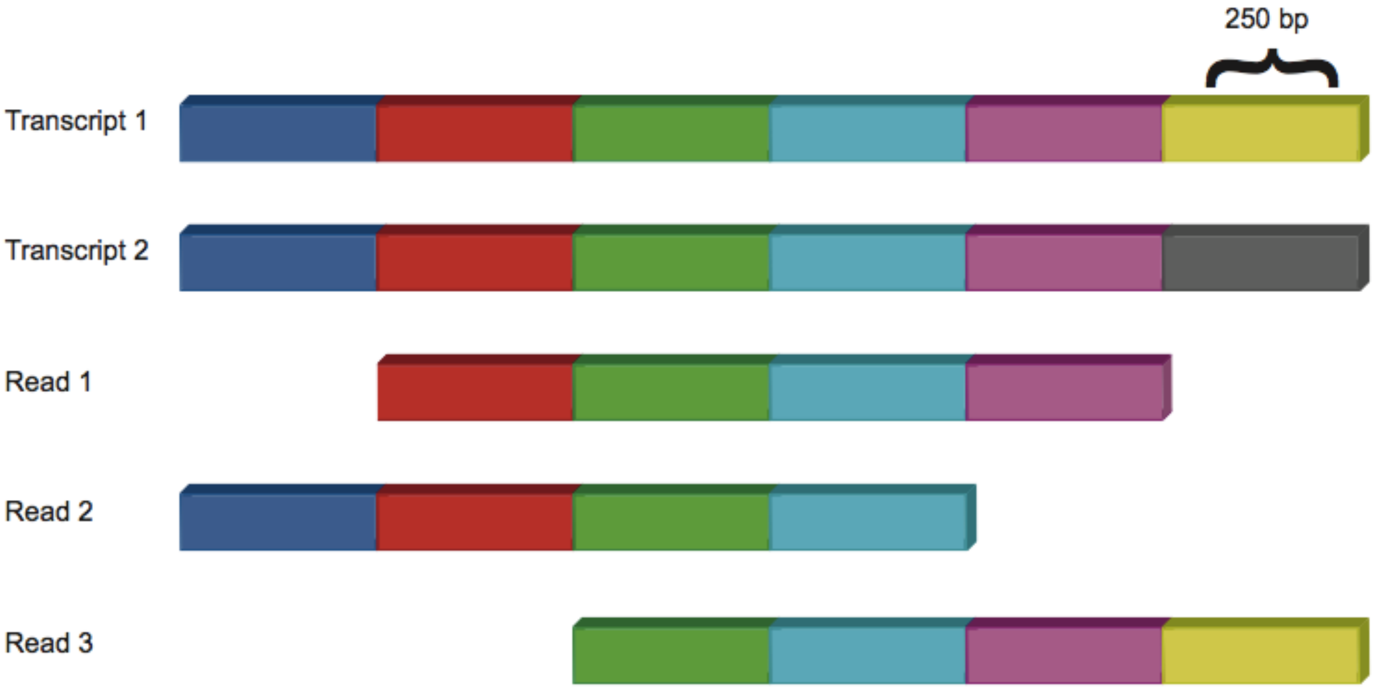


Figure 6. Hypothetical example of 1 kb multimap reads. Only Read 3 can be uniquely mapped due to the unique exon in Transcript 1.

Pyrkosz et al., unpub.

Concluding thoughts

- There's what you can do today, computationally, with existing programs. This is often limited by our time, experience, etc.
- There's what you could, in theory, do with the data you had. This is the upper limit on your accuracy.
- Figuring these out is one of the main reasons you're here :)

Any questions or comments?

What you should do now:

- Take a bit of a break.
- Start up an EC2 instance (m1.large will be plenty big enough).
- Log in via ssh, as you did yesterday.