# Welcome!
# Lecture 1 / CSE 801
# & Workshop closing 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-day workshop, or one-term course, can't possibly show you everything.

## Challenge #4:

**Data gathering technology is changing very fast.**

- We don't understand its limitations or biases very well (e.g. sequencing tech)
- The software and compute infrastructure lags behind volume of data, type of data.

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

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 computational biology.

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 computing.

- e.g. with BLAST,
  - Do you see expected matches with the parameters and database you're using?
- e.g. with models
  - do you have an alternate route to specific answers?

- 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

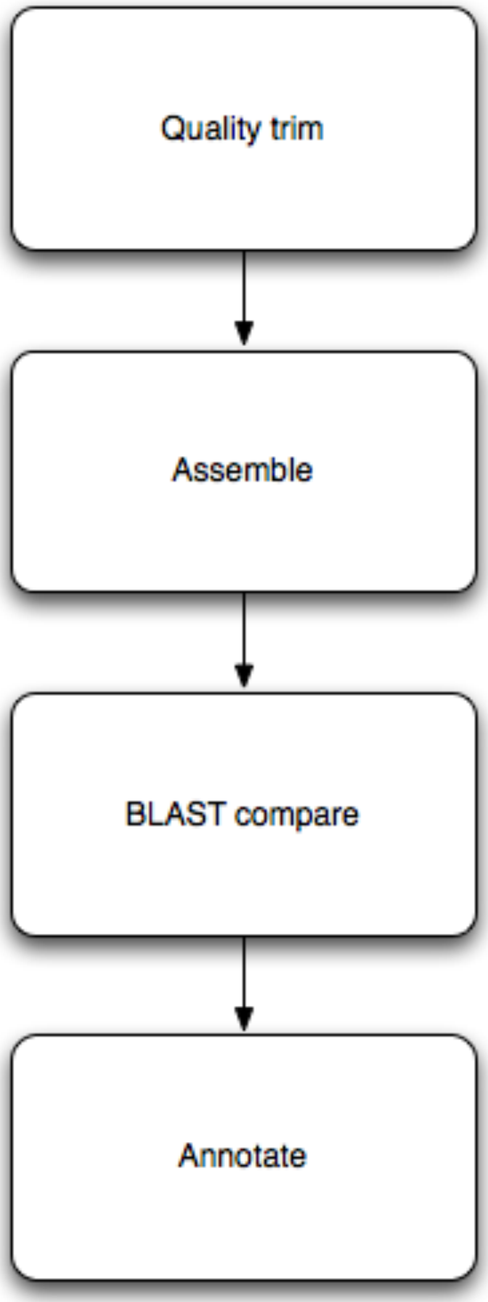
- Does the whole process work?
- “I can reproduce what this other person/lab did, with their data, when I use my own software.”
- This is much more rarely done…

## 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!

## Framing the approach

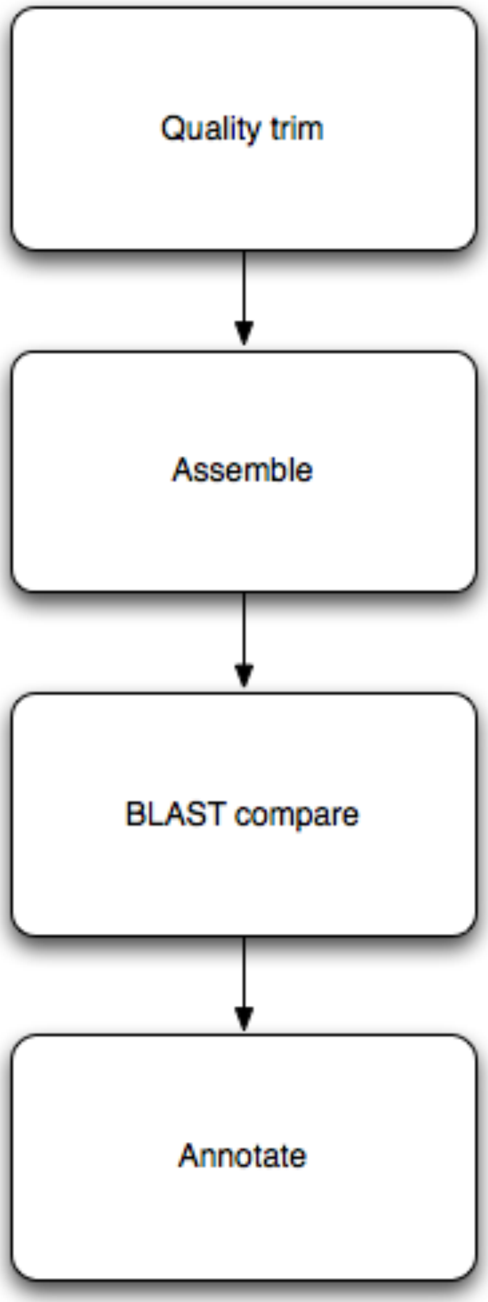
1. How does all this stuff work, generally?

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.

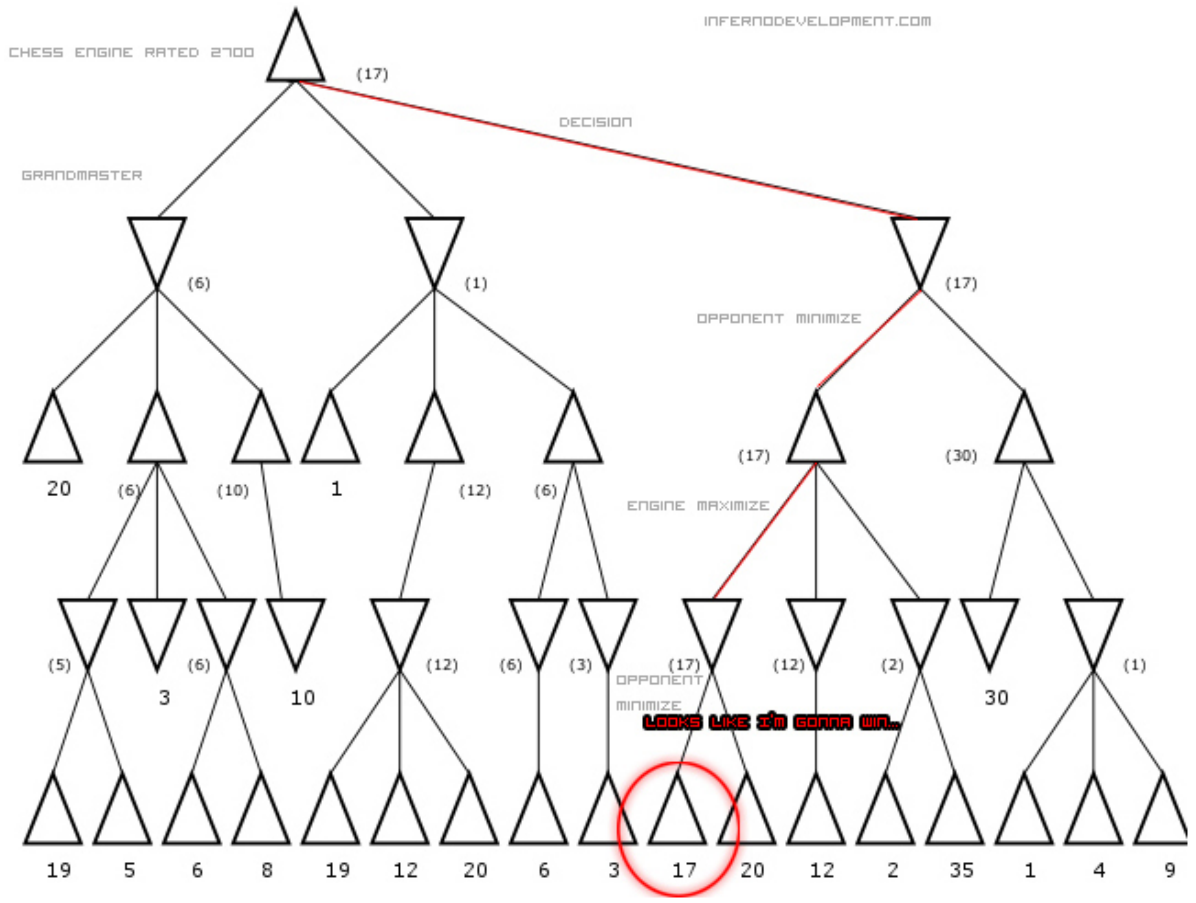
- We'll run into some practical problems:
  - Some programs take a long time to run.
  - Some programs take many different parameters; which are best?
  - Some programs don't finish on "cheap" hardware.

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?

Example: RNAseq for gene expression

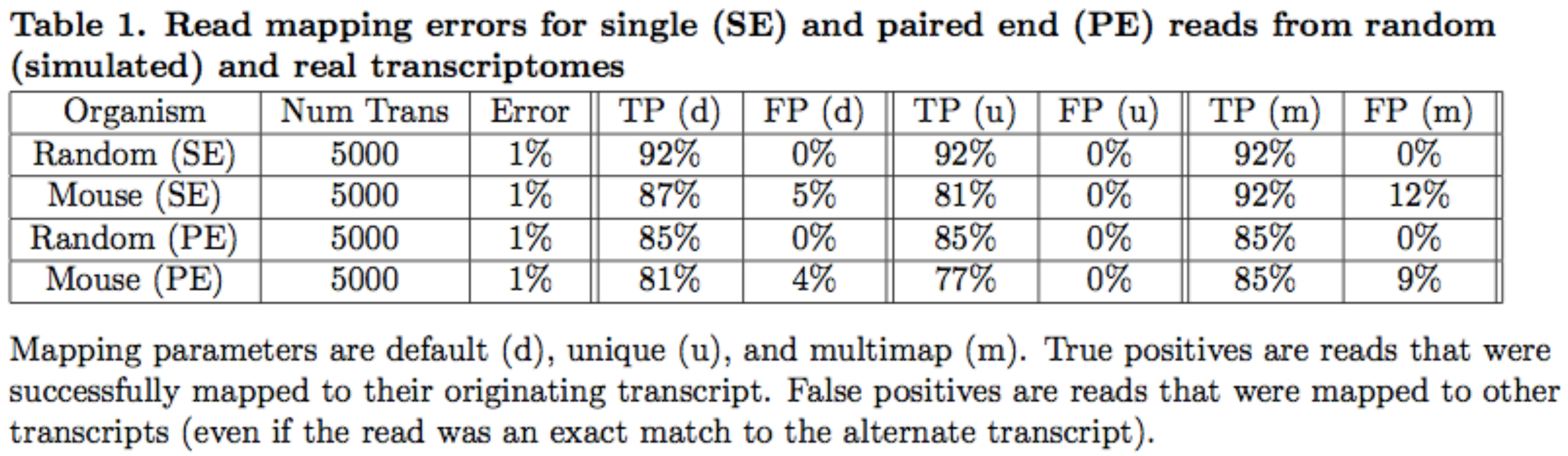


**Table 1. Read mapping errors for single (SE) and paired end (PE) reads from random (simulated) and real transcriptomes**

| 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?

Some details matter; some details don't.

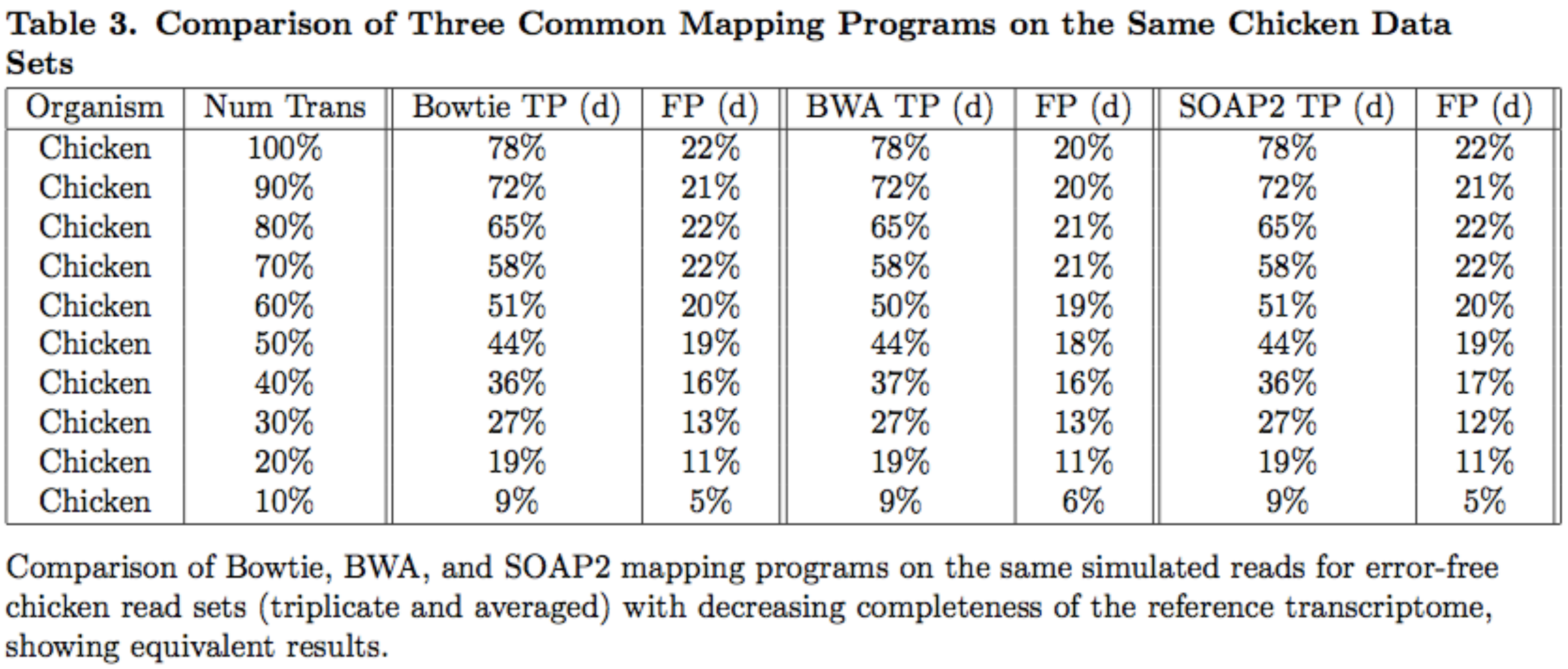


**Table 3. Comparison of Three Common Mapping Programs on the Same Chicken Data Sets**

| 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.

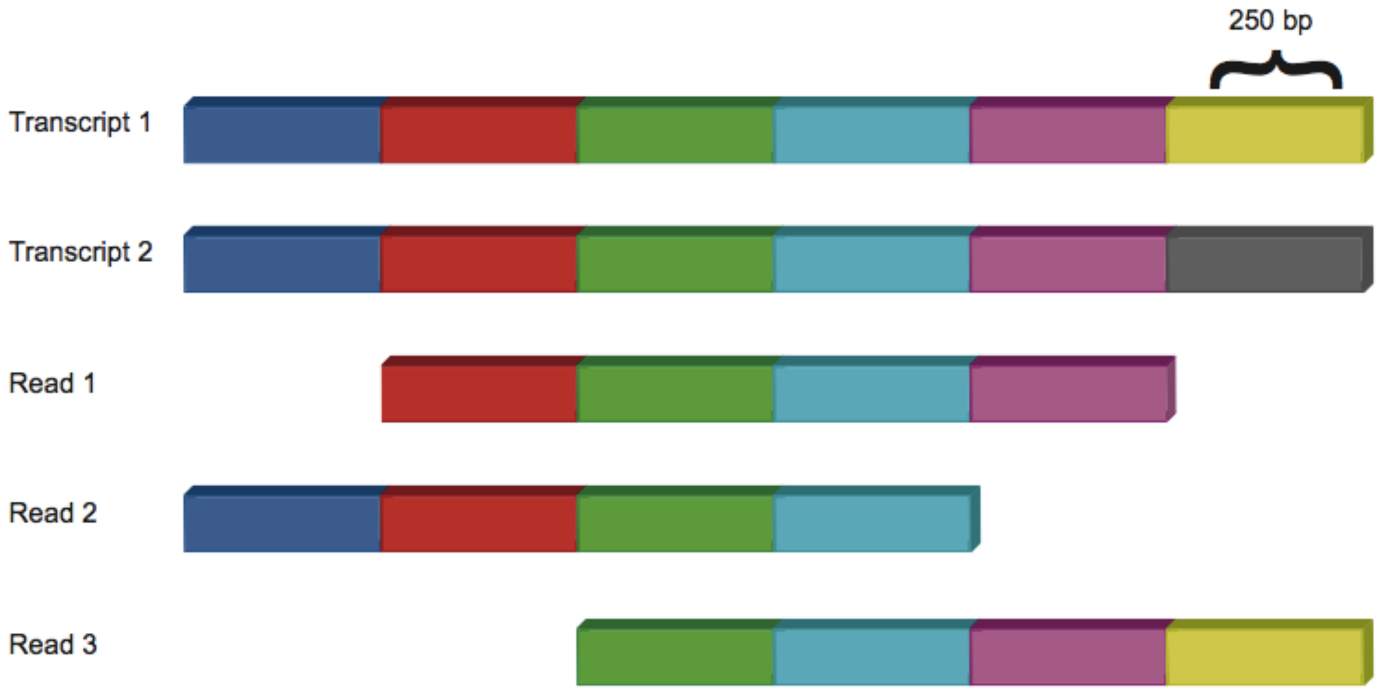


**Figure 6. Hypothetical example of 1 kb multimap reads. Only Read 3 can be uniquely mapped due to the unique exon in Transcript 1.**

Pyrkosz et al., unpub.

## Aka “Science”

- How strong a conclusion *could* we reach from this method?
- Does our analysis *actually* support our conclusions?
- What other analyses can we do, or what other data can we bring in, to better develop/ understand/test our results?

Note! Nowhere specific to “computation”…

## NGS & *de novo* Assembly

## Assembly vs mapping

- No reference needed, for assembly!
  - De novo genomes, transcriptomes…
- But:
  - Scales poorly; need a much bigger computer.
  - Biology gets in the way (repeats!)
  - Need higher coverage
- But but:
  - Often your reference isn't that great, so assembly may actually be the best way to go.

## Assembly

It was the best of times, it was the wor

, it was the worst of times, it was the

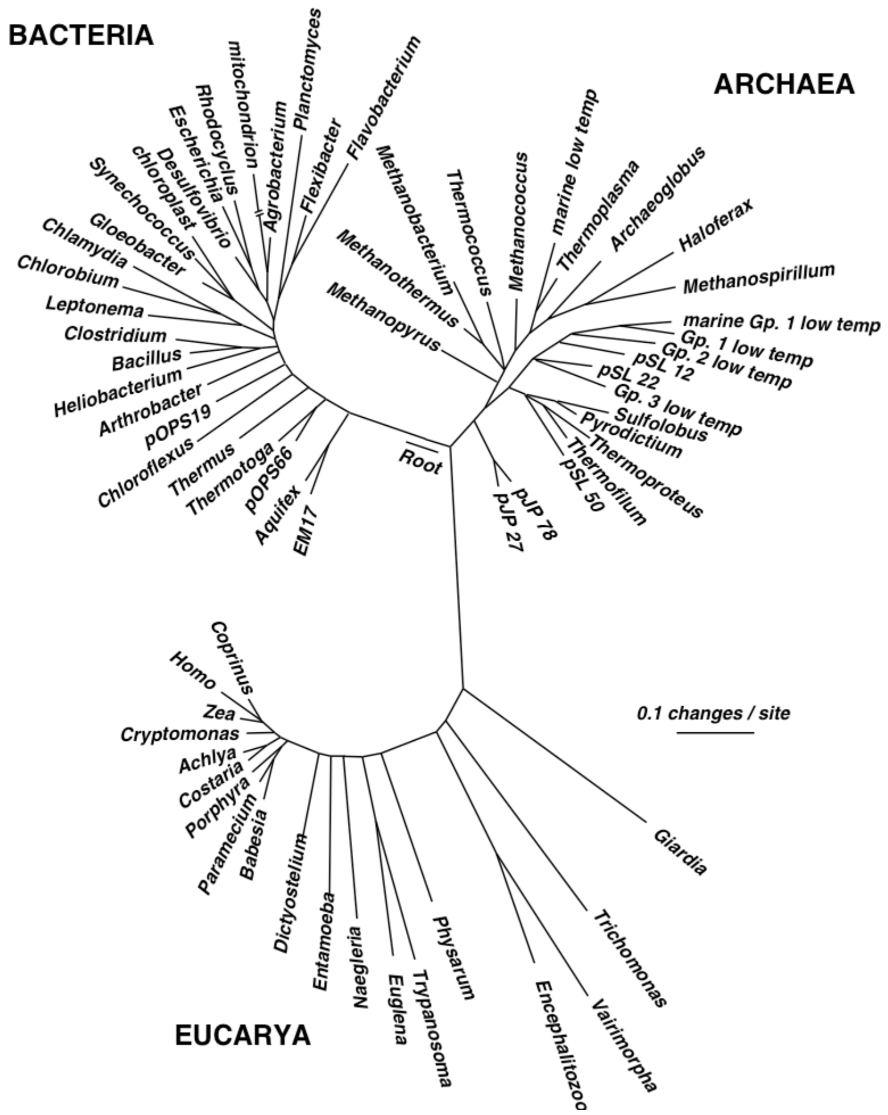
isdom, it was the age of foolishness

mes, it was the age of wisdom, it was th

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness

…but for lots and lots of fragments!

## “But don’t we have all the genomes we need?”



## Assemble based on word overlaps:

**the quick brown fox jumped**

**jumped over the lazy dog**

**the quick brown fox jumped over the lazy dog**

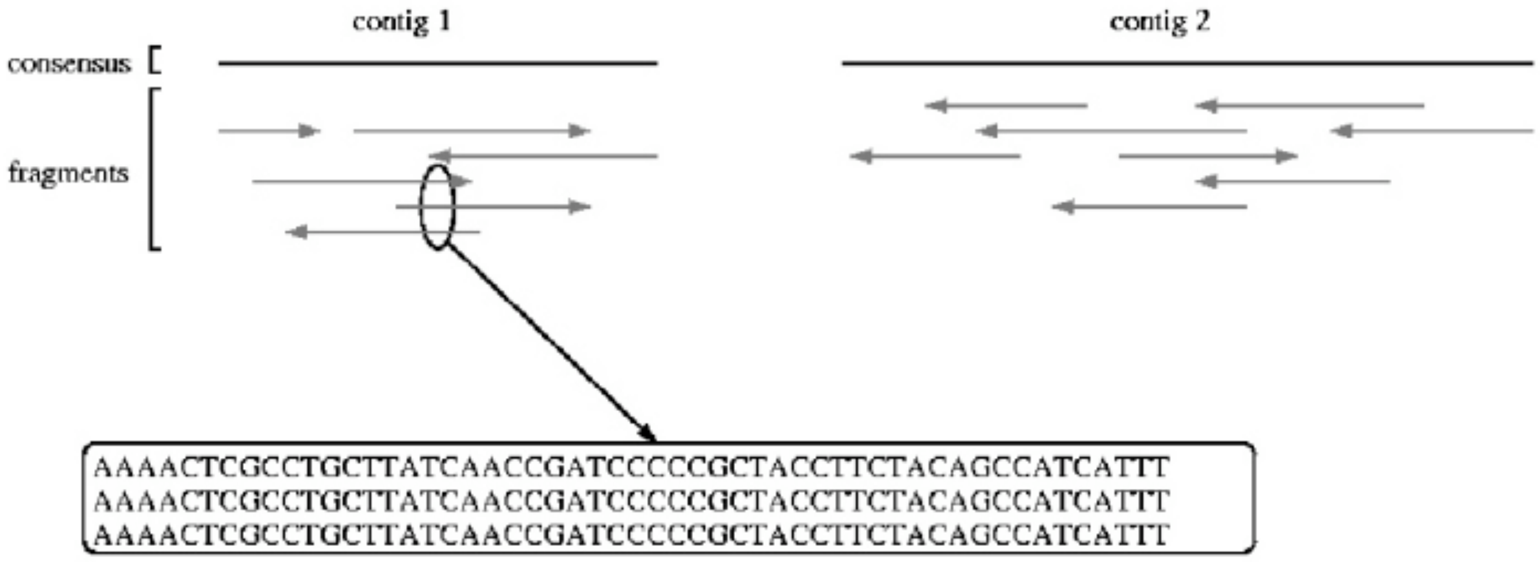
## Repeats do cause problems:

**my chemical romance: na na na**

**na na na, batman!**

# Shotgun sequencing & assembly

Randomly fragment & sequence from DNA; reassemble computationally.

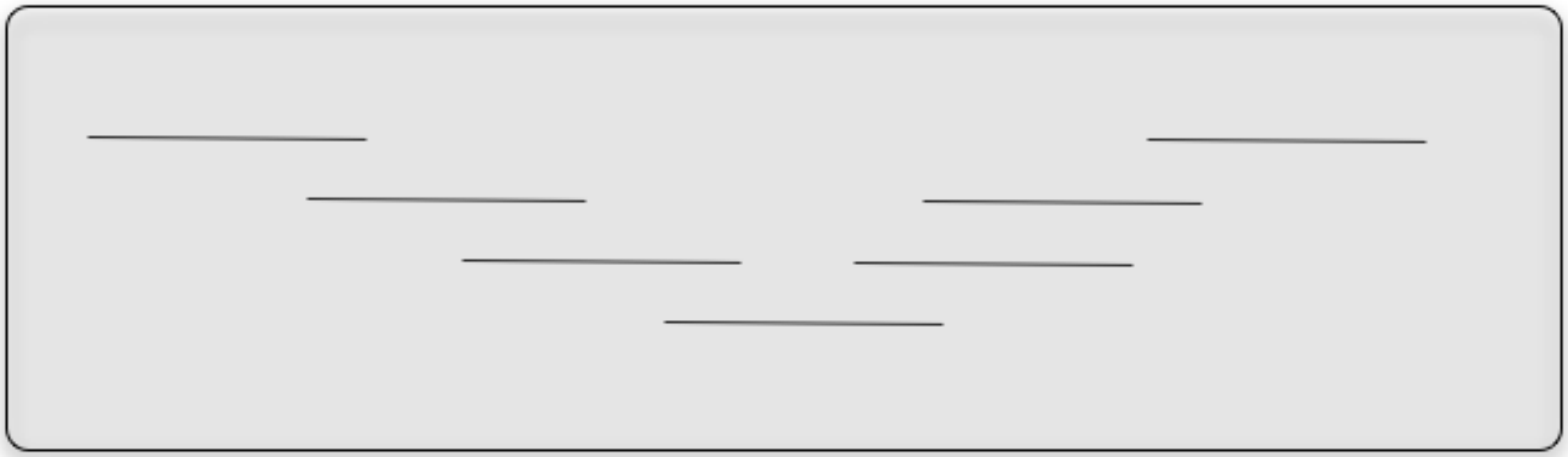


contig 1
contig 2
consensus
fragments
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT

UMD assembly primer (cbcb.umd.edu)

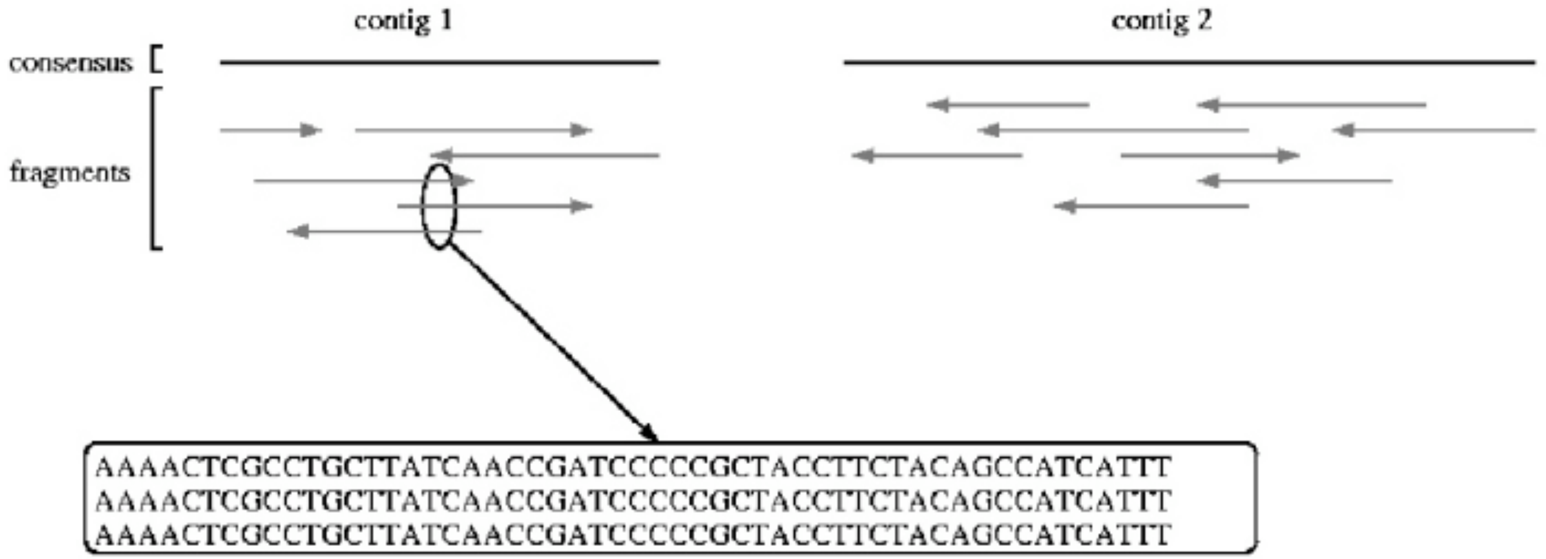
## Assembly – no subdivision!

Assembly is inherently an *all by all* process. There is no good way to subdivide the reads without potentially missing a key connection



## Short-read assembly

- Short-read assembly is problematic
- Relies on very deep coverage, ruthless read trimming, paired ends.



contig 1
contig 2
consensus
fragments
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT

UMD assembly primer (cbcb.umd.edu)

## Short read lengths are hard.

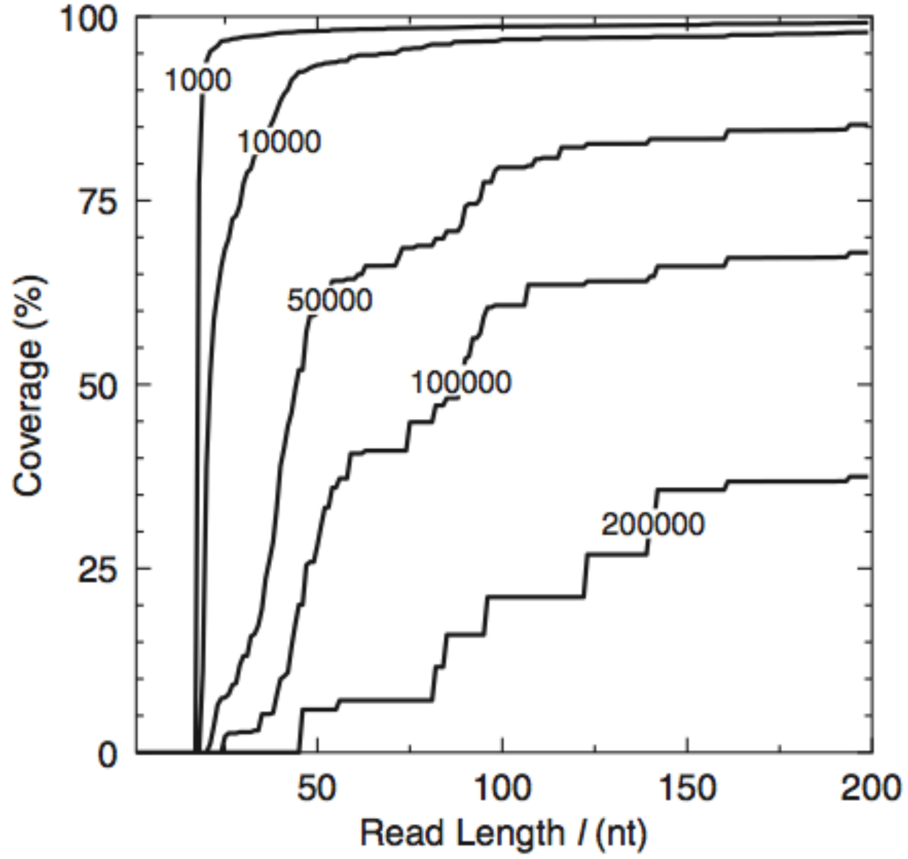


**Figure 3.** Percentage of the *E.coli* genome covered by contigs greater than a threshold length as a function of read length.

Whiteford et al., Nuc. Acid Res, 2005

# Short read lengths are hard.

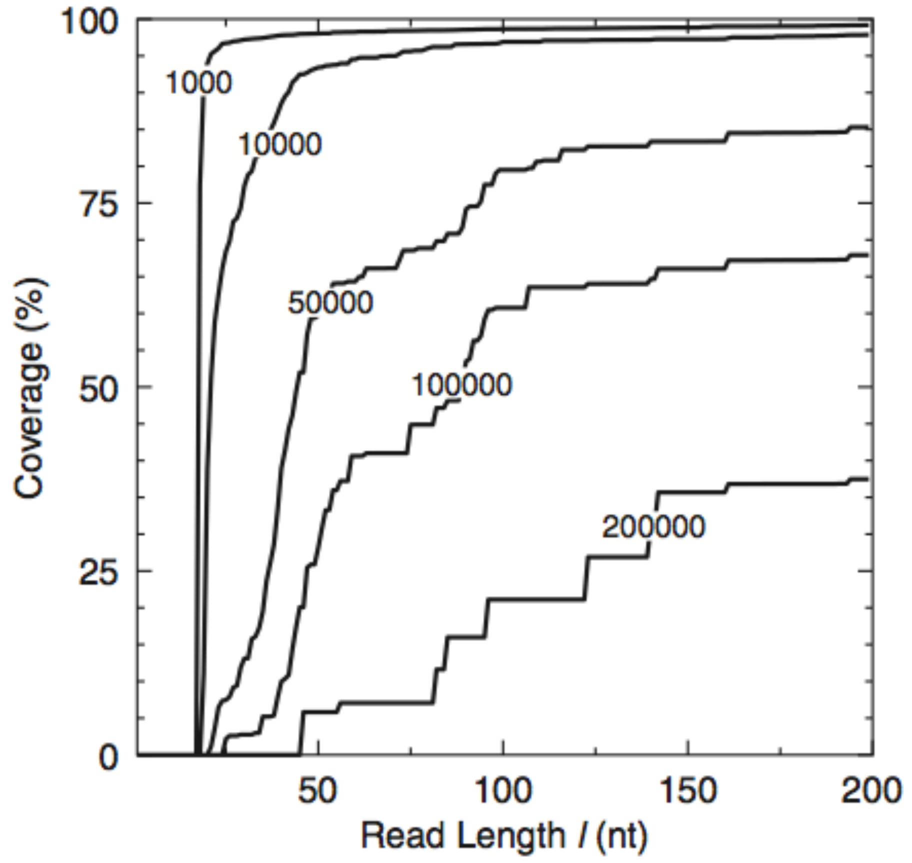


**Figure 3.** Percentage of the *E.coli* genome covered by contigs greater than a threshold length as a function of read length.

Conclusion: even with a read length of 200, the E. coli genome cannot be assembled completely.

Why?

Whiteford et al., Nuc. Acid Res, 2005

## Short read lengths are hard.

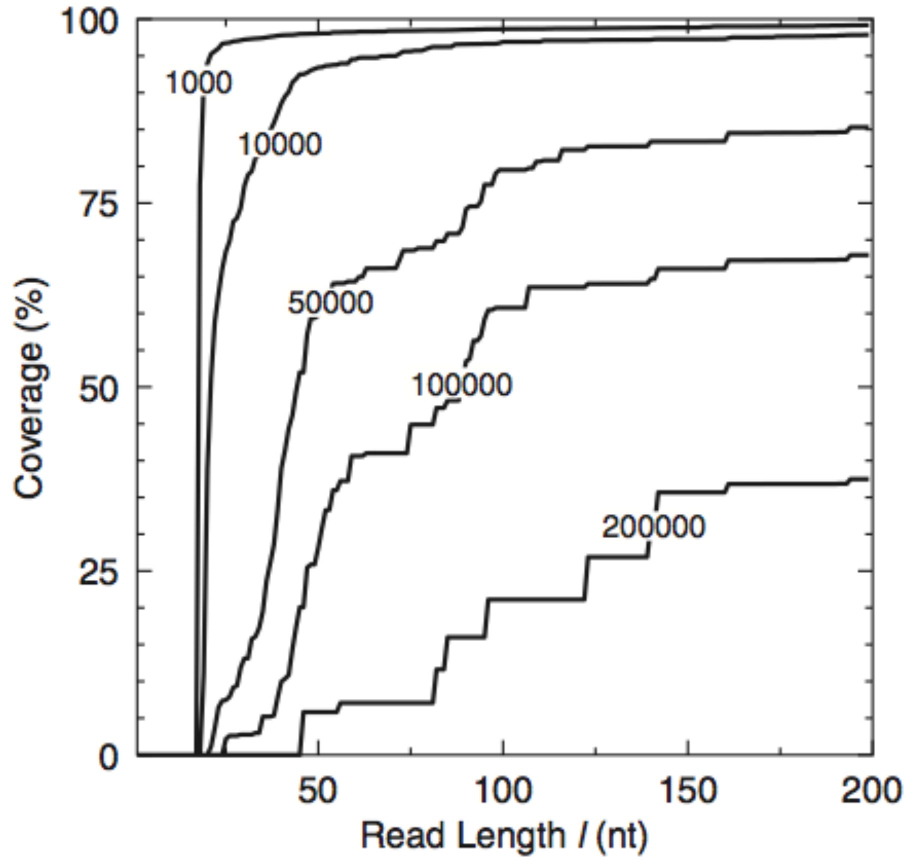


**Figure 3.** Percentage of the *E.coli* genome covered by contigs greater than a threshold length as a function of read length.

Conclusion: even with a read length of 200, the E. coli genome cannot be assembled completely.

Why? **REPEATS.**

This is why paired-end sequencing is so important for assembly.

Whiteford et al., Nuc. Acid Res, 2005

## Four main challenges for *de novo* sequencing.

- Repeats.
- Low coverage.
- Errors

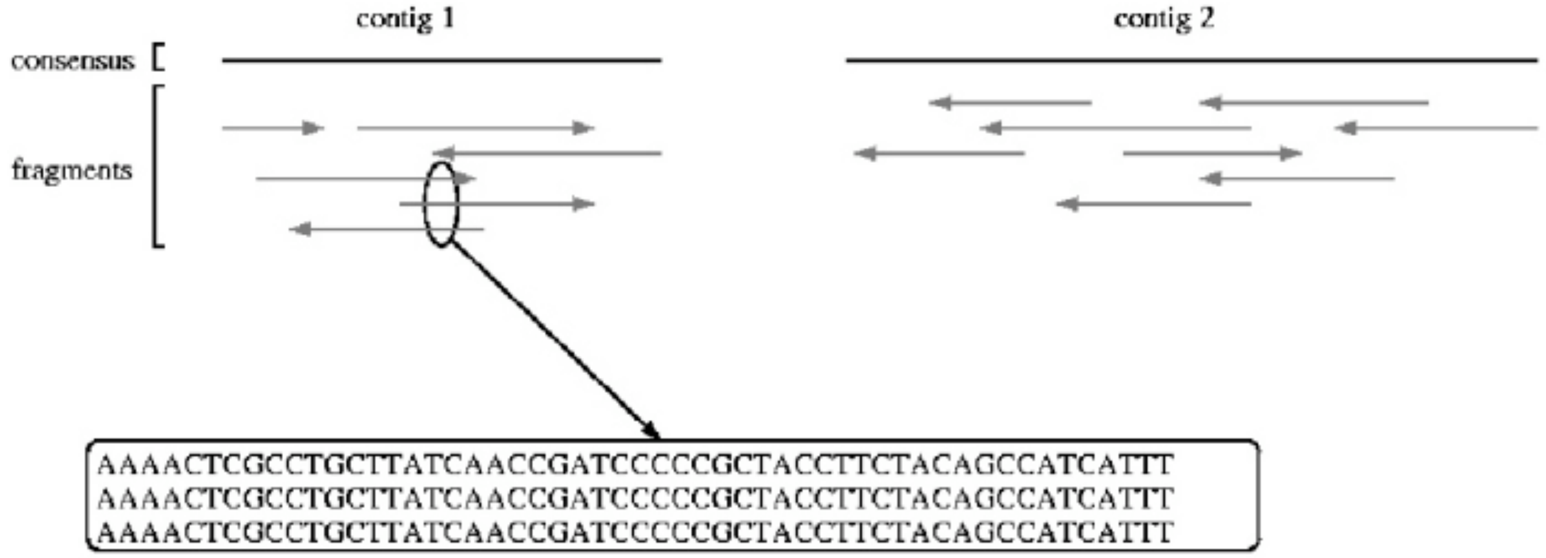
These introduce breaks in the construction of contigs.

- *Variation* in coverage – transcriptomes and metagenomes, as well as amplified genomic.

This challenges the assembler to distinguish between erroneous connections (e.g. repeats) and real connections.

# Repeats

- Overlaps don't place sequences uniquely when there are repeats present.


contig 1
contig 2
consensus
fragments
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT

UMD assembly primer (cbcb.umd.edu)

## Coverage

Easy calculation:

(# reads x avg read length) / genome size

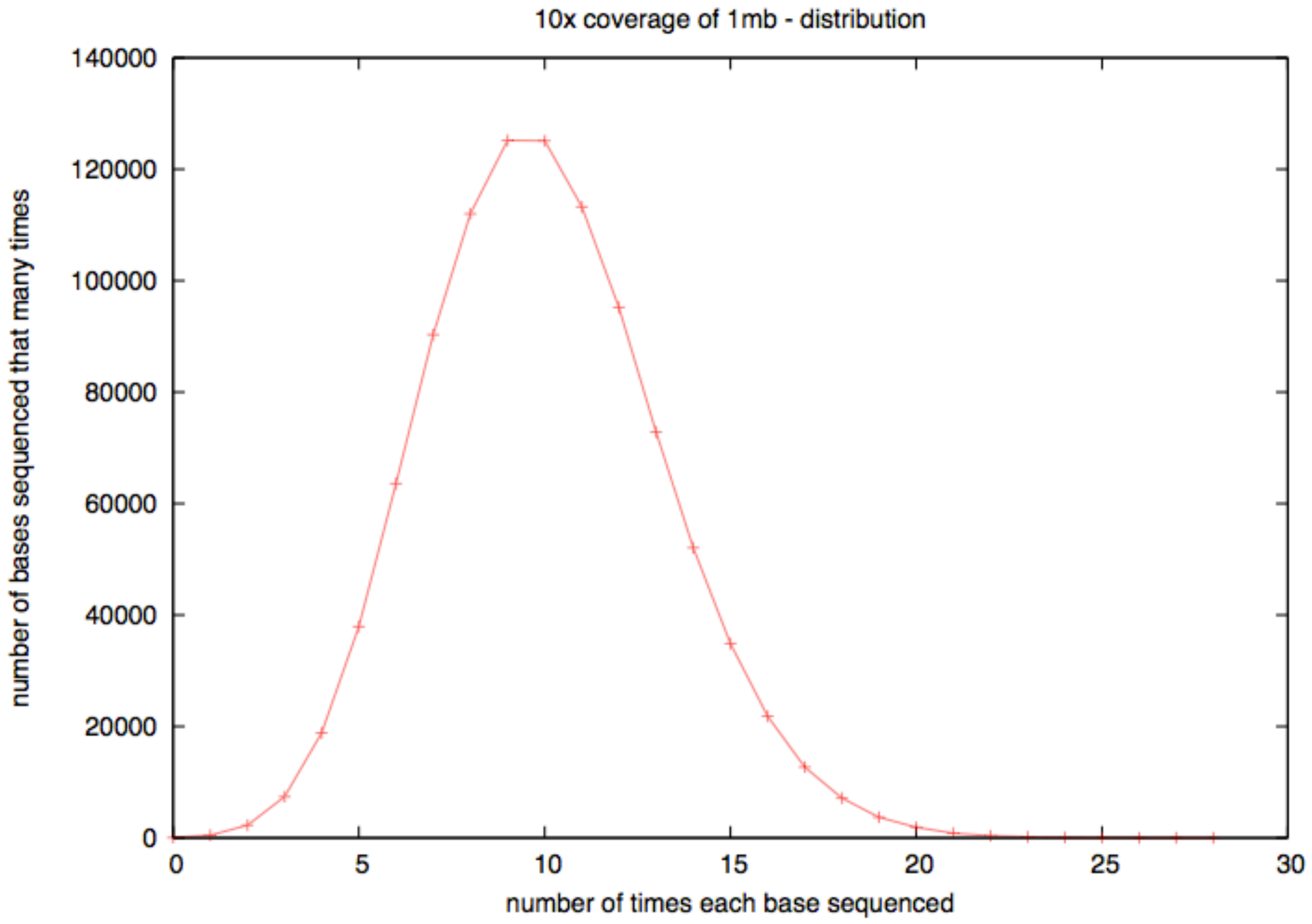
So, for haploid human genome:

30m reads x 100 bp = 3 bn

## Coverage

- “1x” doesn’t mean every DNA sequence is read once.
- It means that, if sampling were *systematic,* it would be.
- Sampling isn’t systematic, it’s random!

# Actual coverage varies widely from the average, for low avg coverage



## Two basic assembly approaches

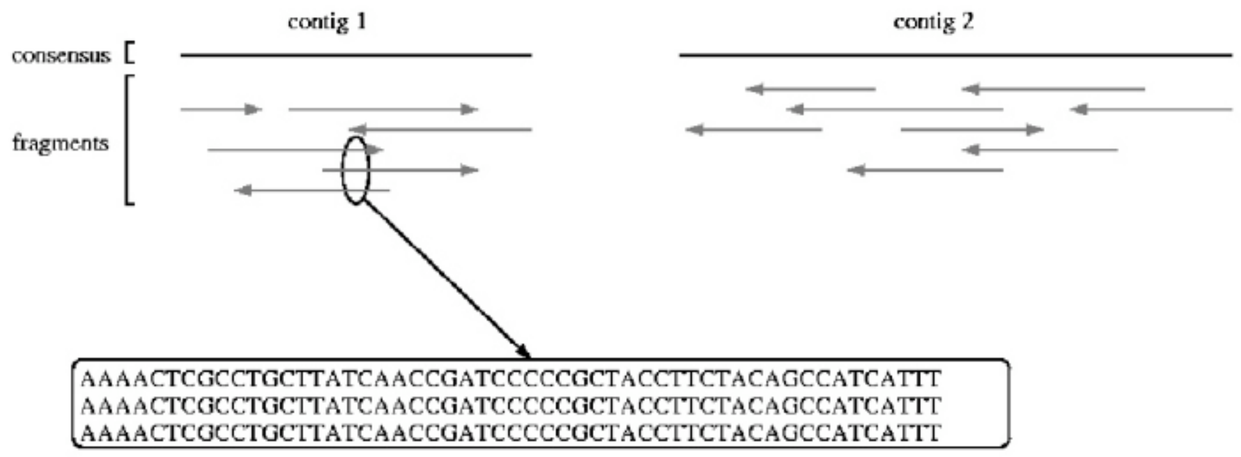
- Overlap/layout/consensus
- De Bruijn k-mer graphs

The former is used for long reads, esp all Sanger-based assemblies. The latter is used because of memory efficiency.

# Overlap/layout/consensus

Essentially,

1. Calculate all overlaps
2. Cluster based on overlap.
3. Do a multiple sequence alignment


contig 1
contig 2
consensus
fragments
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT
AAAACTCGCCTGCTTATCAACCGATCCCCCGCTACCTTCTACAGCCATCATTT

UMD assembly primer (cbcb.umd.edu)

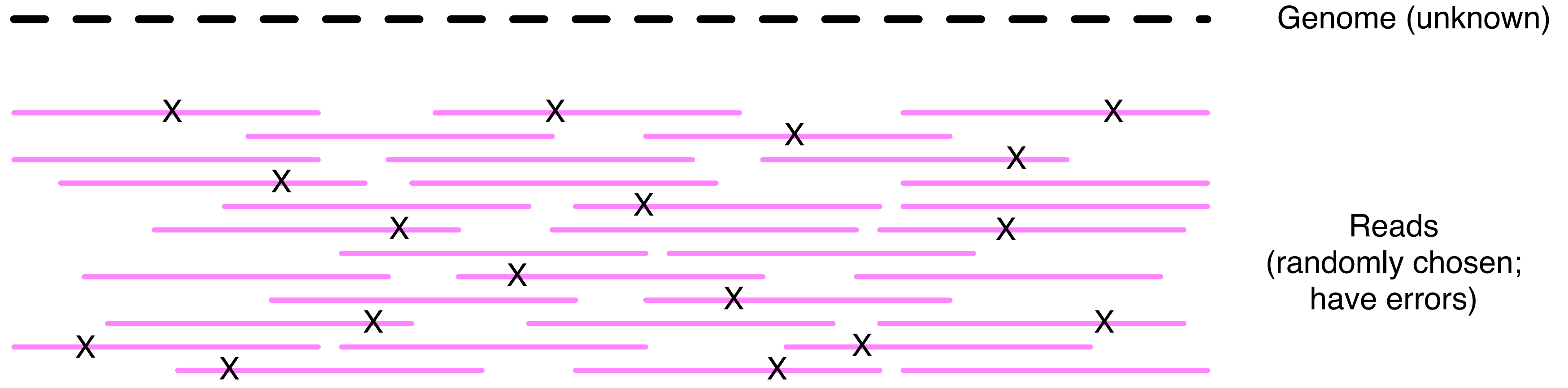
## K-mers

Break reads (of any length) down into multiple overlapping words of fixed length *k.*

```
ATGGACCAGATGACAC (k=12) =>

ATGGACCAGATG
 TGGACCAGATGA
  GGACCAGATGAC
   GACCAGATGACA
    ACCAGATGACAC
```

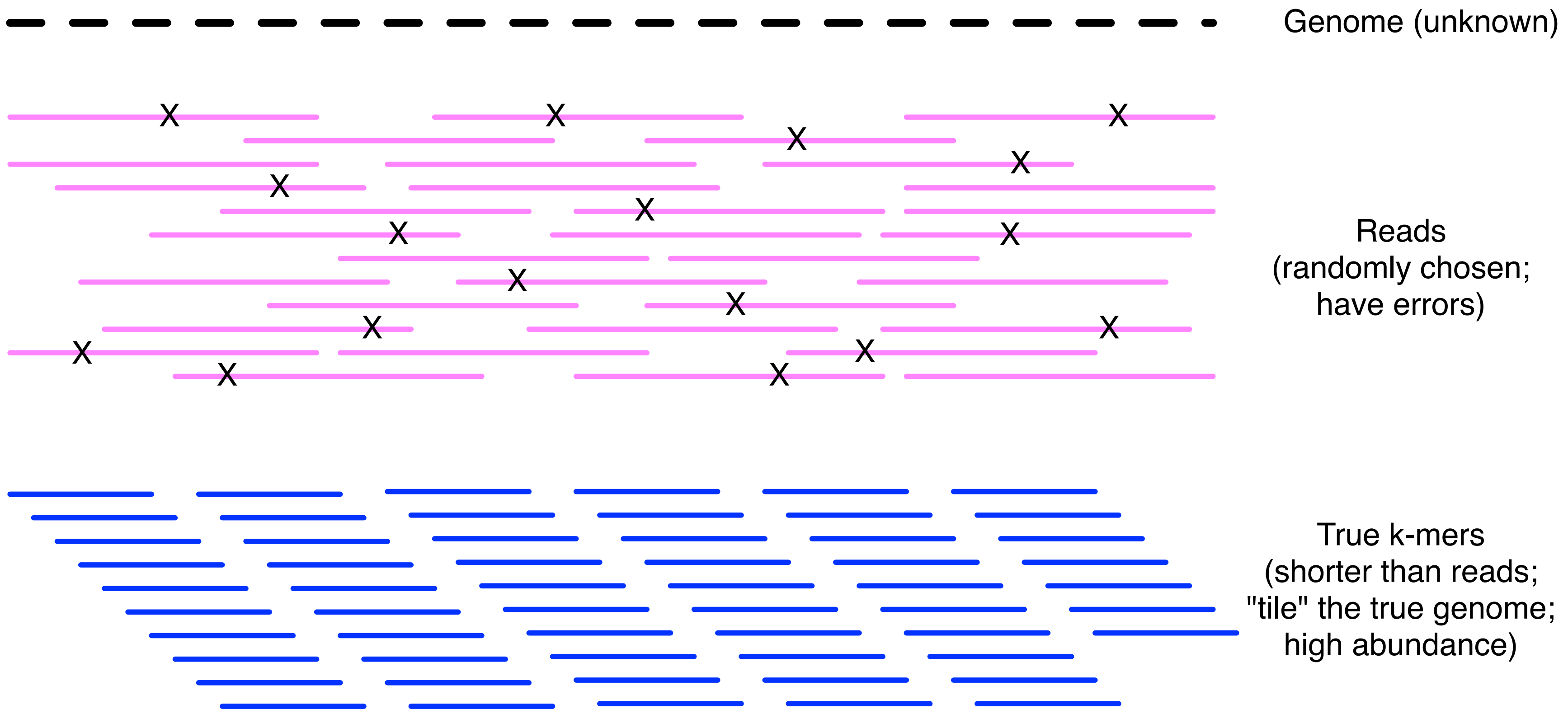
# Shotgun sequencing & k-mers



"Coverage" is simply the average number of reads that overlap each true base in genome.

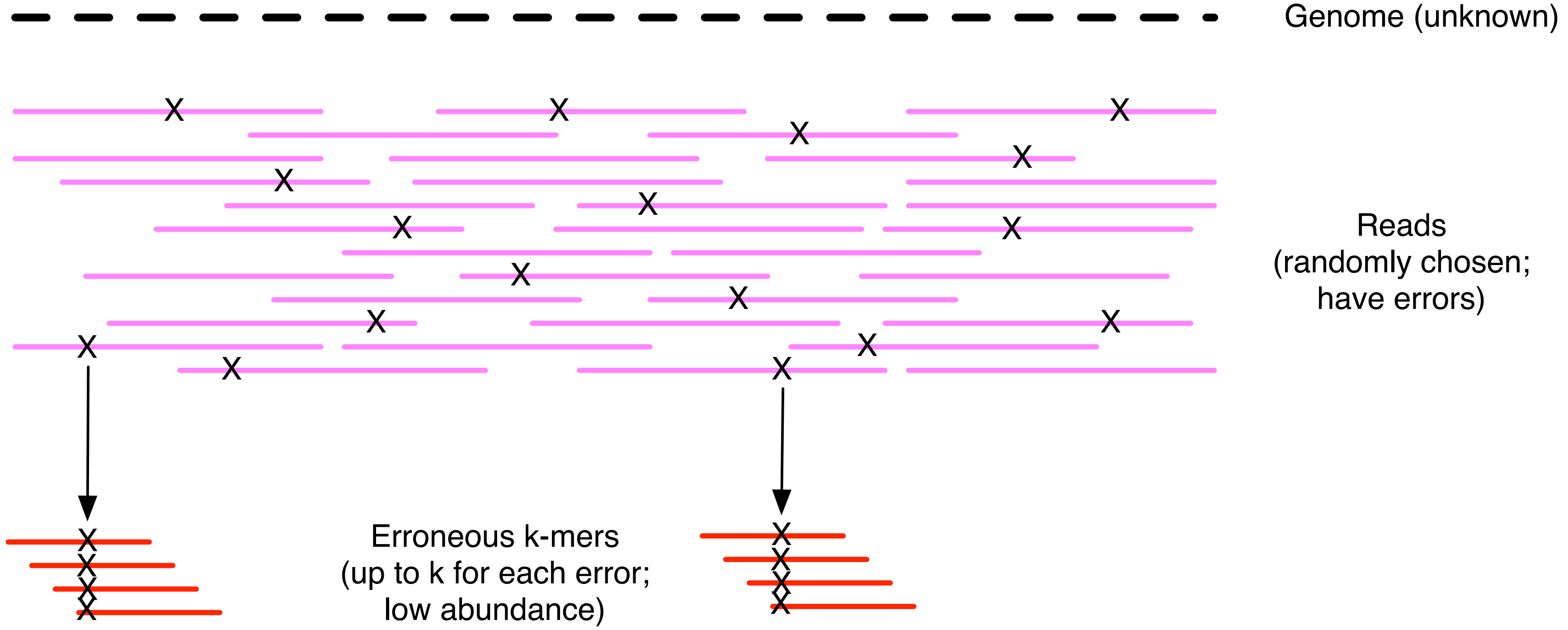
Here, the coverage is ~10 – just draw a line straight down from the top through all of the reads.

# Reducing to k-mers ⇔ overlaps


Genome (unknown)
Reads (randomly chosen; have errors)
True k-mers (shorter than reads; "tile" the true genome; high abundance)

Note that k-mer abundance is not properly represented here! Each blue k-mer will be present around 10 times.

# Errors create *new* k-mers



Each single base error generates ~k new k-mers.
Generally, erroneous k-mers show up only once – errors are *random.*

# K-mers – what size k to use?

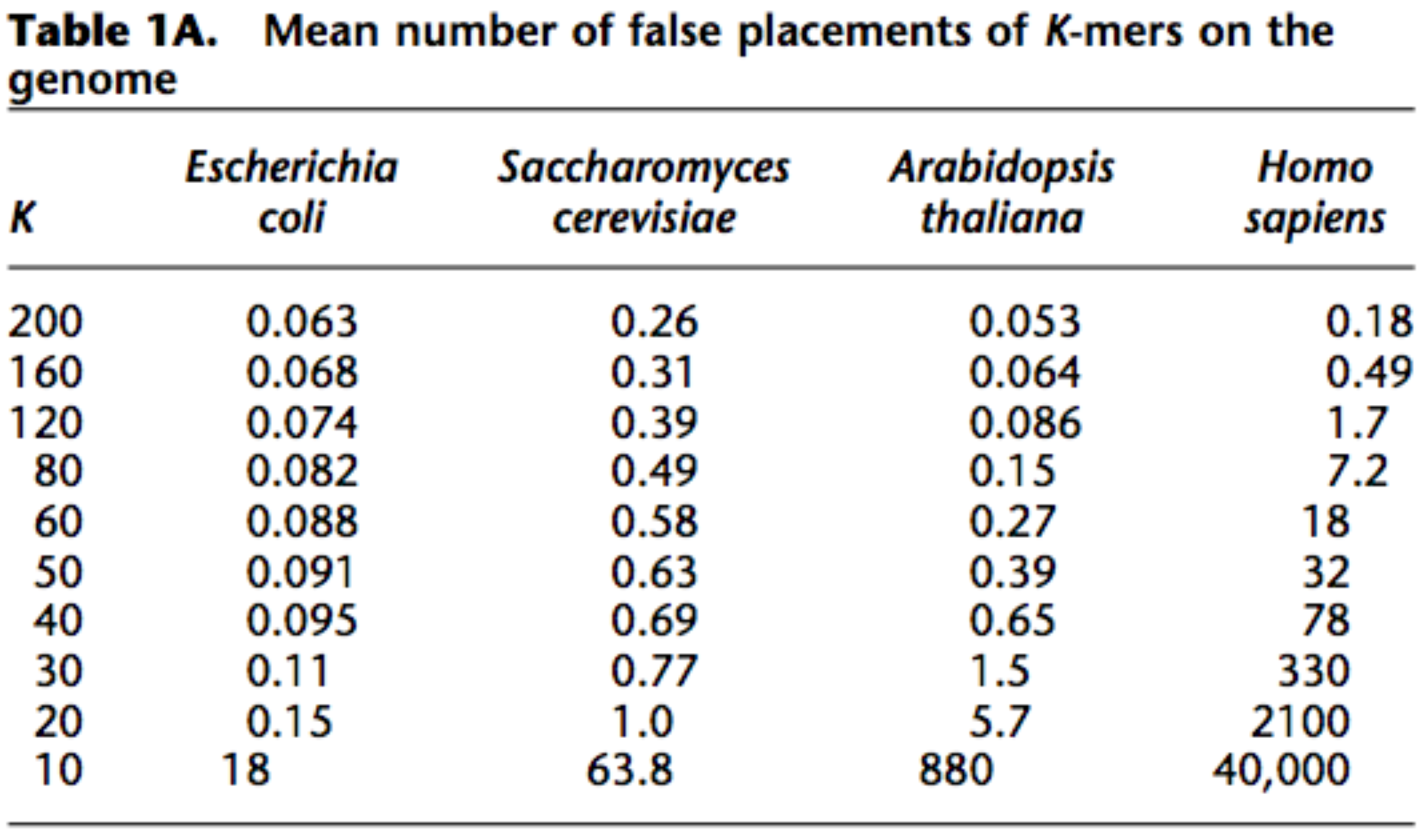
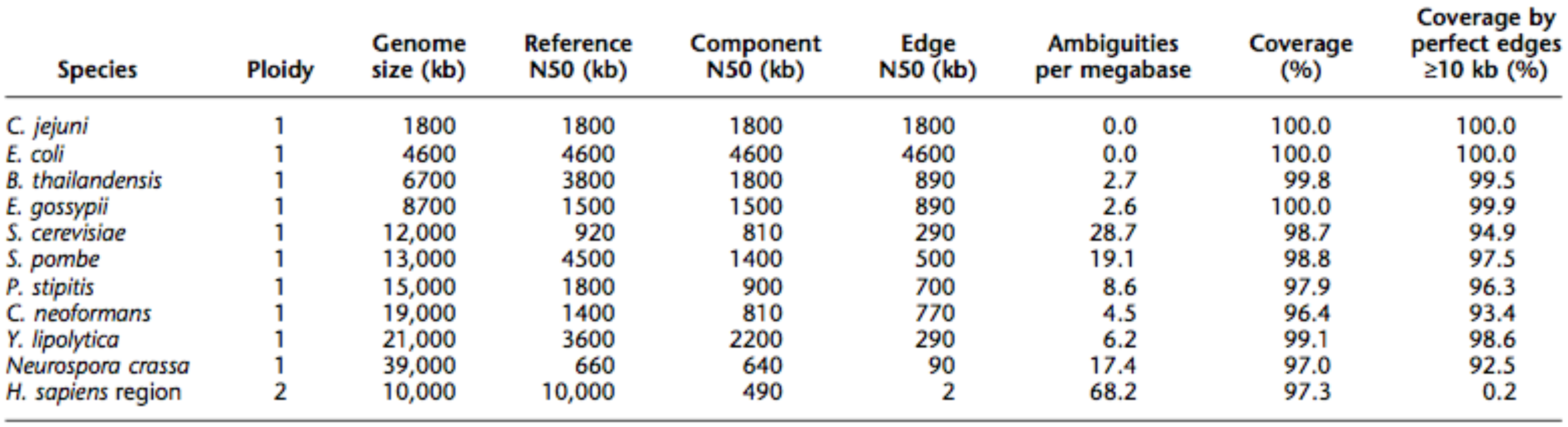


**Table 1A.** Mean number of false placements of *K*-mers on the genome

| *K* | *Escherichia coli* | *Saccharomyces cerevisiae* | *Arabidopsis thaliana* | *Homo sapiens* |
|---|---|---|---|---|
| 200 | 0.063 | 0.26 | 0.053 | 0.18 |
| 160 | 0.068 | 0.31 | 0.064 | 0.49 |
| 120 | 0.074 | 0.39 | 0.086 | 1.7 |
| 80 | 0.082 | 0.49 | 0.15 | 7.2 |
| 60 | 0.088 | 0.58 | 0.27 | 18 |
| 50 | 0.091 | 0.63 | 0.39 | 32 |
| 40 | 0.095 | 0.69 | 0.65 | 78 |
| 30 | 0.11 | 0.77 | 1.5 | 330 |
| 20 | 0.15 | 1.0 | 5.7 | 2100 |
| 10 | 18 | 63.8 | 880 | 40,000 |

Butler et al., Genome Res, 2009

# Big genomes are problematic

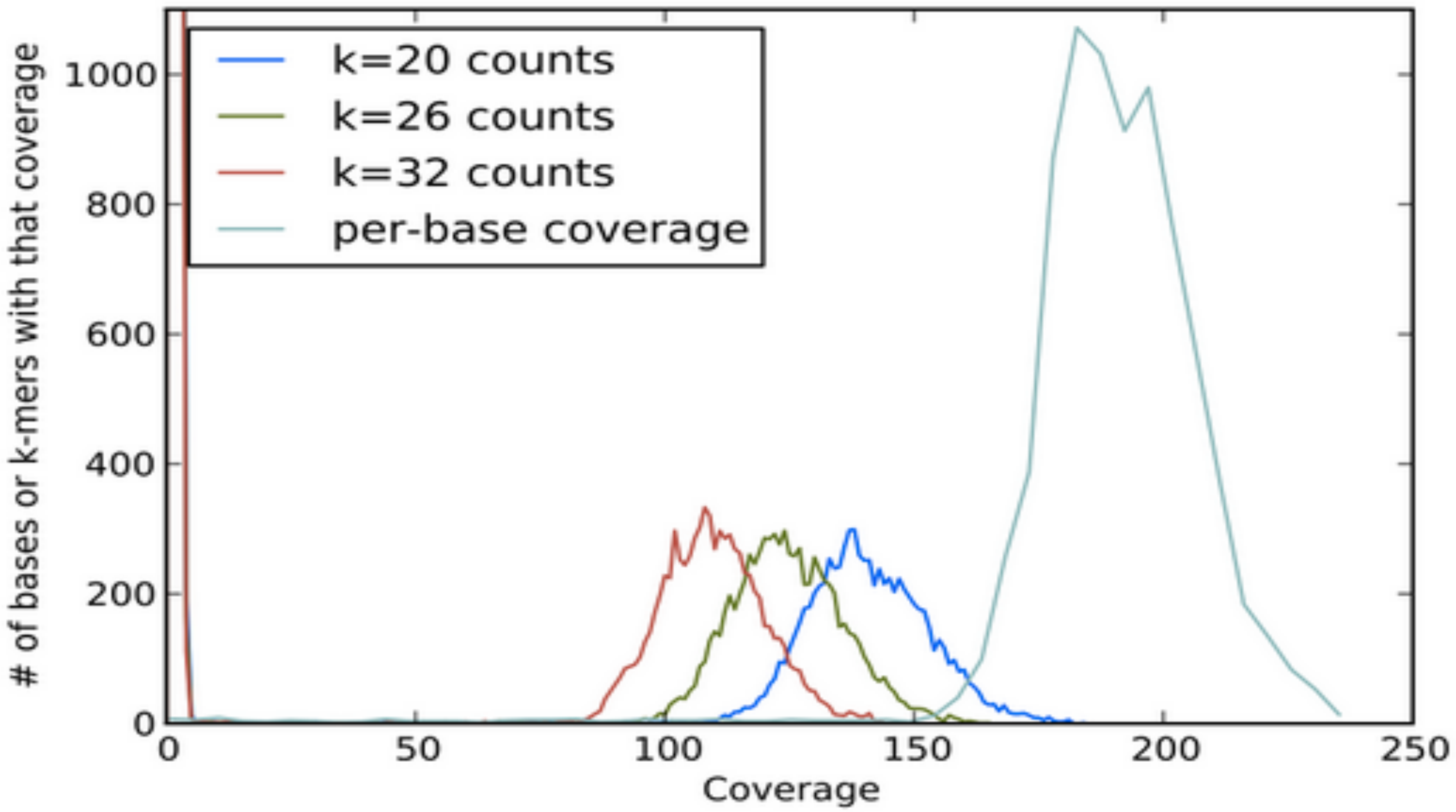


| Species | Ploidy | Genome size (kb) | Reference N50 (kb) | Component N50 (kb) | Edge N50 (kb) | Ambiguities per megabase | Coverage (%) | Coverage by perfect edges ≥10 kb (%) |
|---|---|---|---|---|---|---|---|---|
| *C. jejuni* | 1 | 1800 | 1800 | 1800 | 1800 | 0.0 | 100.0 | 100.0 |
| *E. coli* | 1 | 4600 | 4600 | 4600 | 4600 | 0.0 | 100.0 | 100.0 |
| *B. thailandensis* | 1 | 6700 | 3800 | 1800 | 890 | 2.7 | 99.8 | 99.5 |
| *E. gossypii* | 1 | 8700 | 1500 | 1500 | 890 | 2.6 | 100.0 | 99.9 |
| *S. cerevisiae* | 1 | 12,000 | 920 | 810 | 290 | 28.7 | 98.7 | 94.9 |
| *S. pombe* | 1 | 13,000 | 4500 | 1400 | 500 | 19.1 | 98.8 | 97.5 |
| *P. stipitis* | 1 | 15,000 | 1800 | 900 | 700 | 8.6 | 97.9 | 96.3 |
| *C. neoformans* | 1 | 19,000 | 1400 | 810 | 770 | 4.5 | 96.4 | 93.4 |
| *Y. lipolytica* | 1 | 21,000 | 3600 | 2200 | 290 | 6.2 | 99.1 | 98.6 |
| *Neurospora crassa* | 1 | 39,000 | 660 | 640 | 90 | 17.4 | 97.0 | 92.5 |
| *H. sapiens* region | 2 | 10,000 | 10,000 | 490 | 2 | 68.2 | 97.3 | 0.2 |

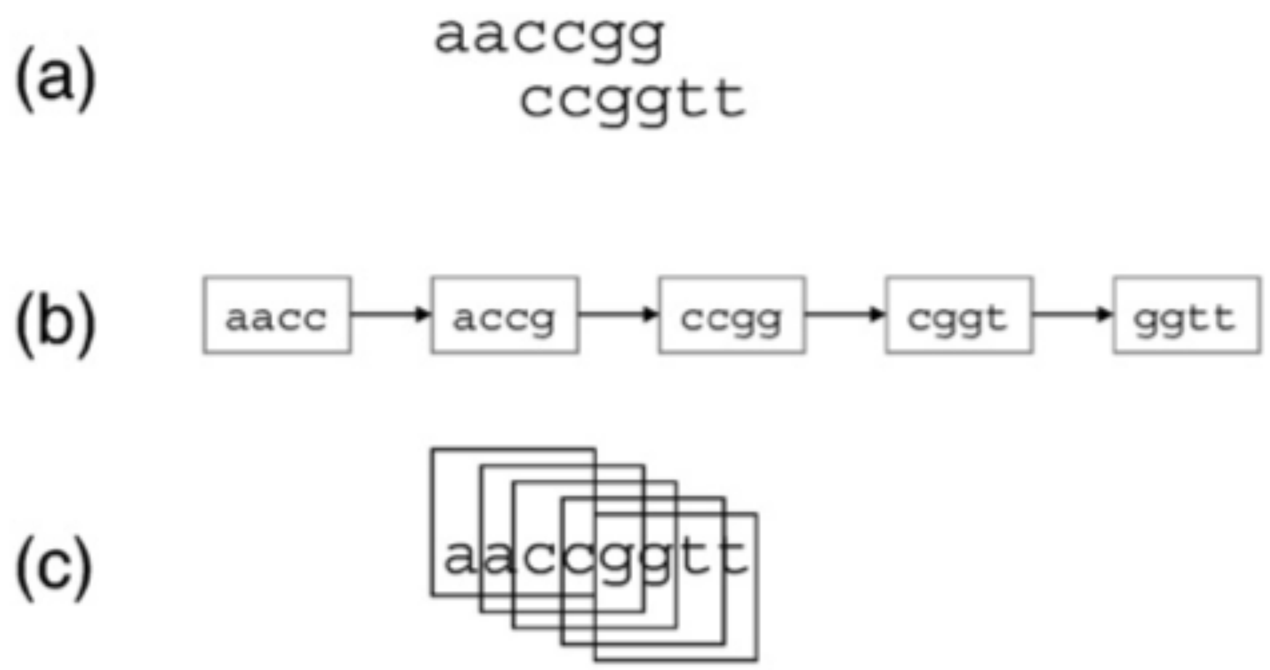
Butler et al., Genome Res, 2009

# Choice of k affects apparent coverage

Two reasons: read length; errors and variants.

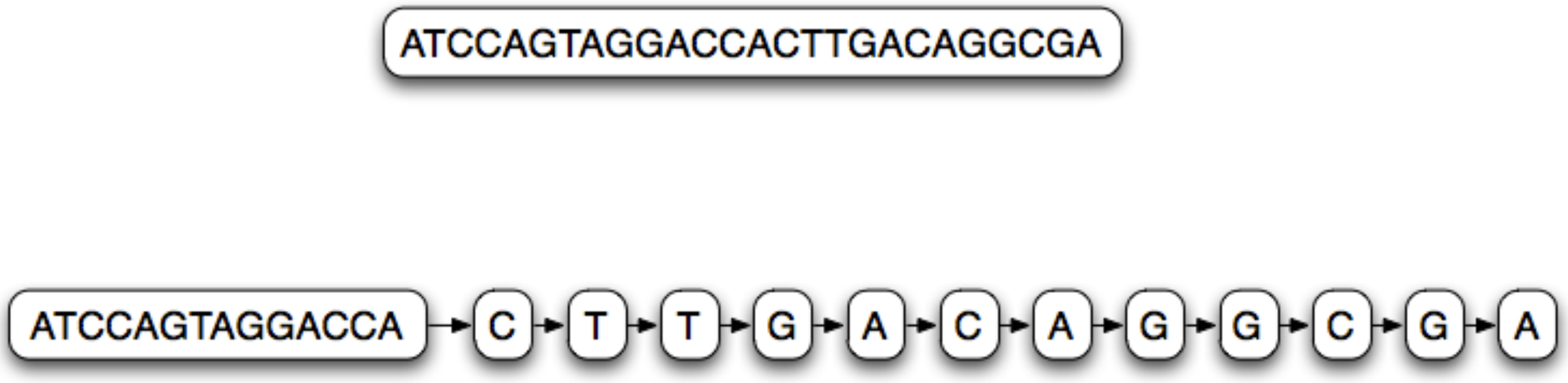


# K-mer graphs - overlaps



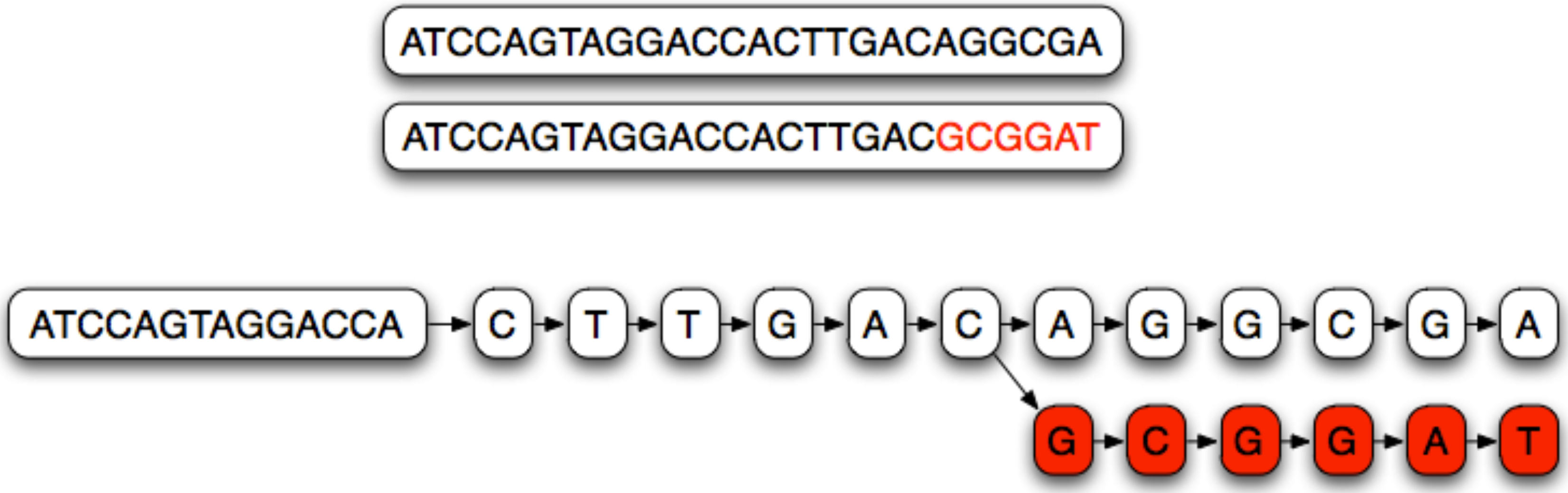
J.R. Miller et al. / Genomics (2010)

# K-mer graph (k=14)



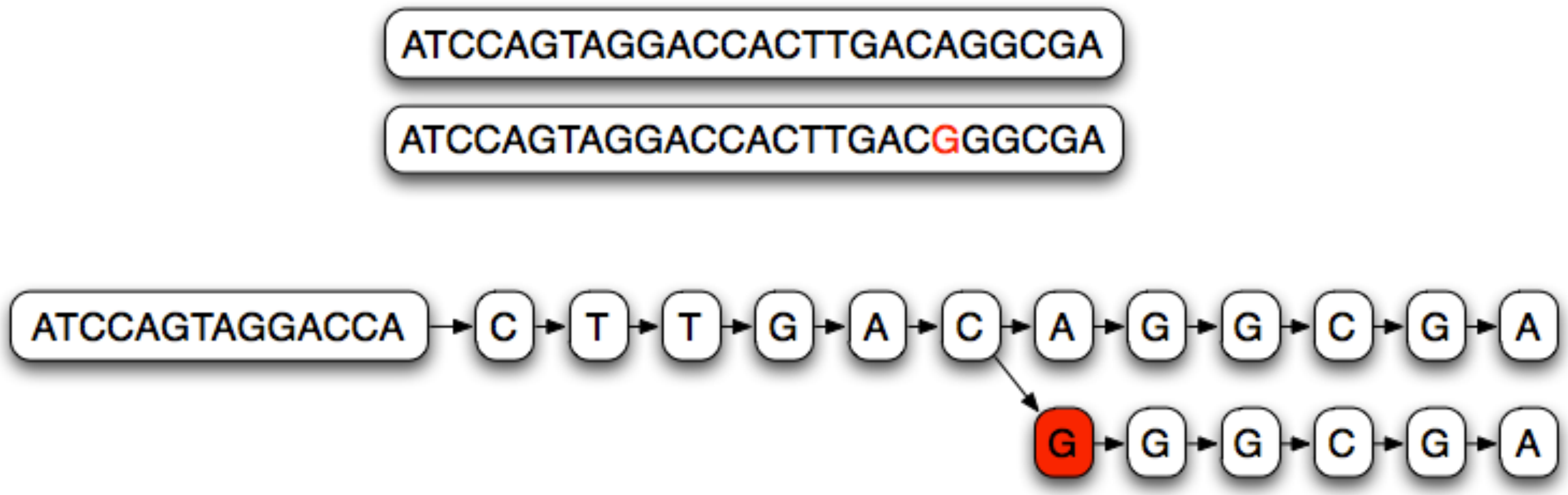
Each node represents a 14-mer;
Links between each node are 13-mer overlaps

## K-mer graph (k=14)



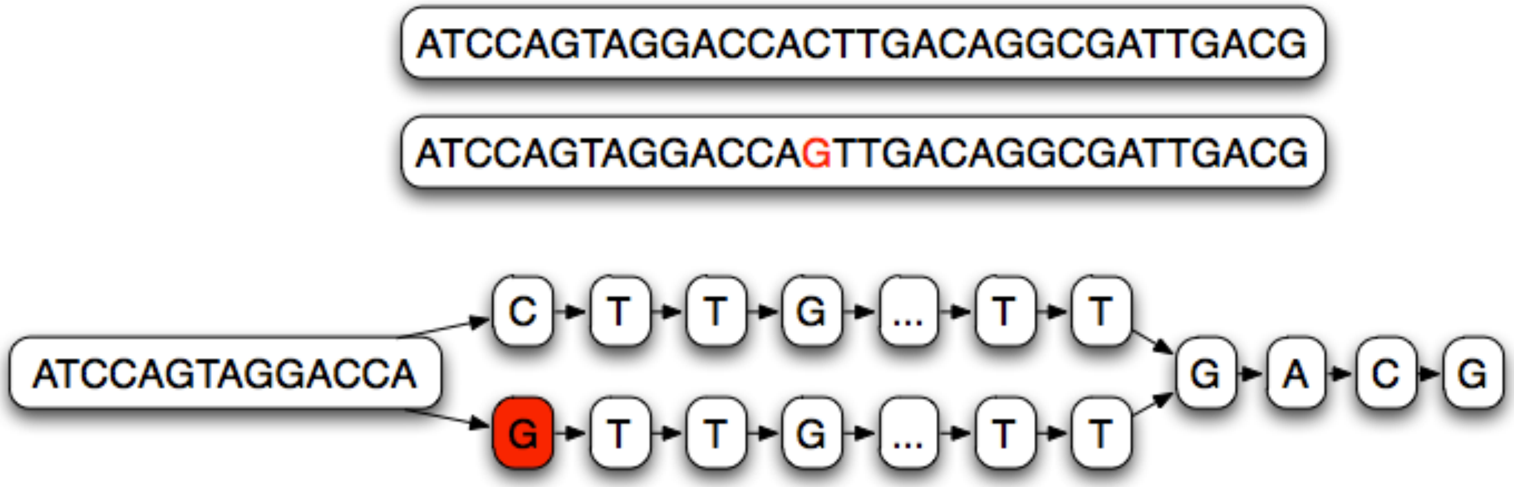
Branches in the graph represent partially overlapping sequences.

# K-mer graph (k=14)



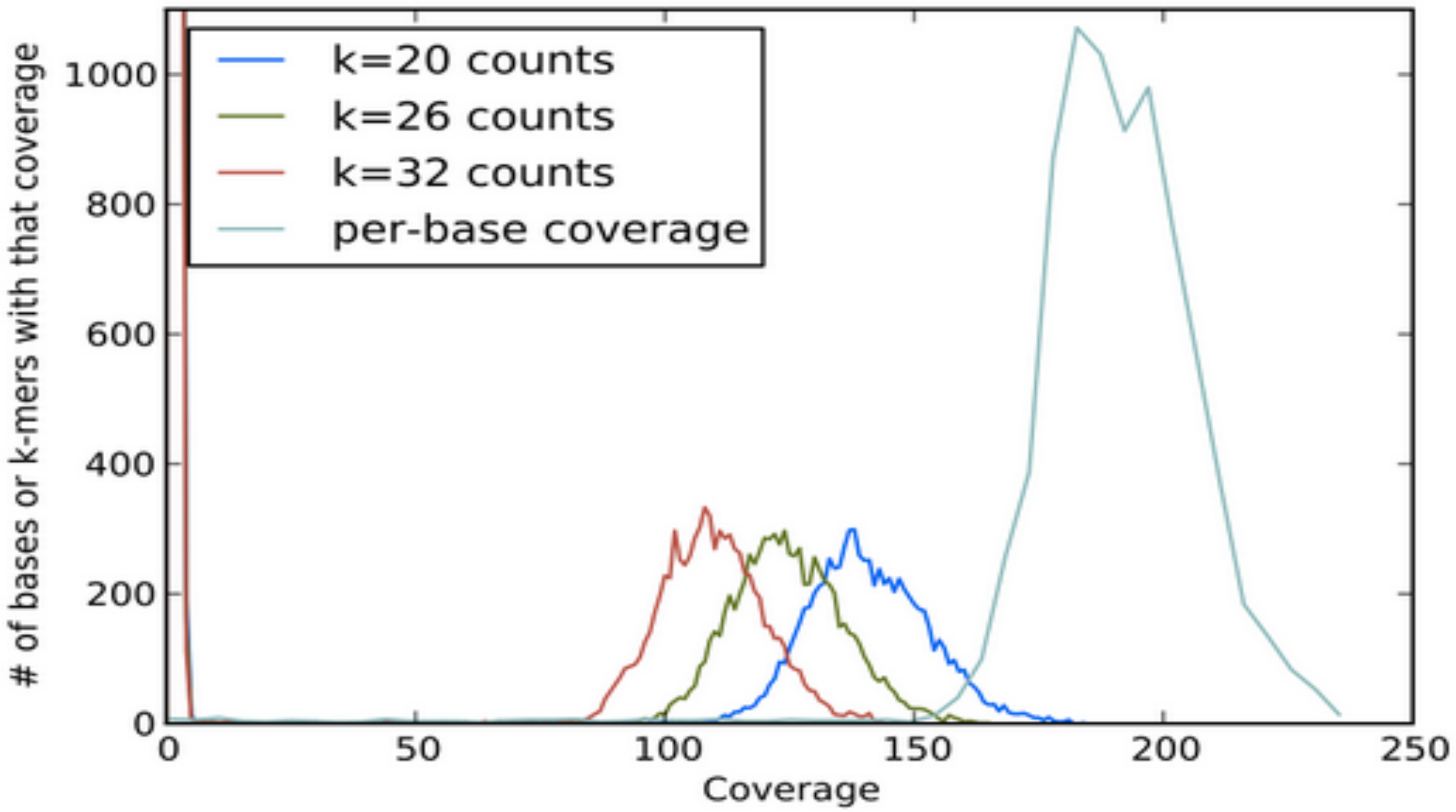
Single nucleotide variations cause long branches

# K-mer graph (k=14)

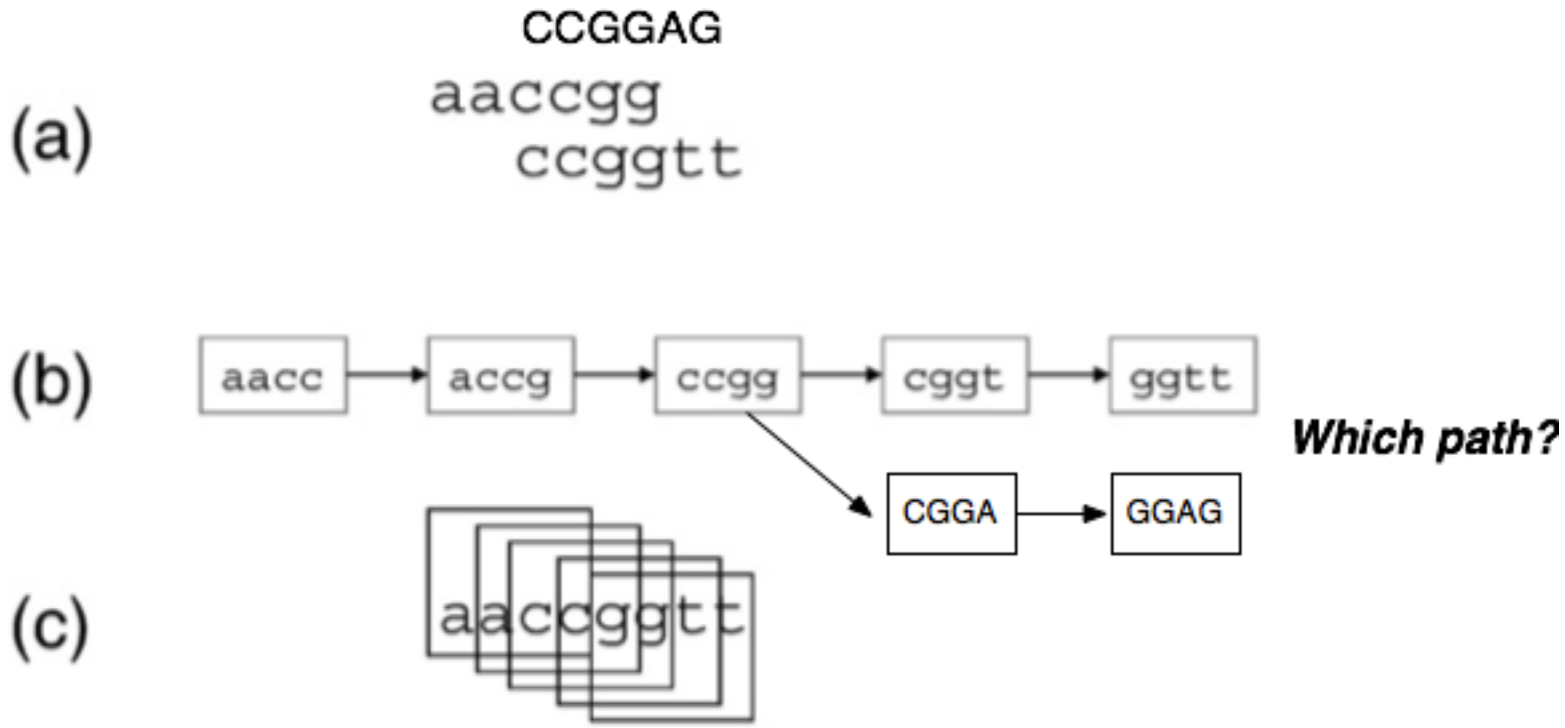

ATCCAGTAGGACCACTTGACAGGCGATTGACG
ATCCAGTAGGACCAGTTGACAGGCGATTGACG
ATCCAGTAGGACCA
C
T
T
G
...
T
T
G
T
T
G
...
T
T
G
A
C
G

Single nucleotide variations cause long branches;
They don't rejoin quickly.

## Choice of k affects apparent coverage



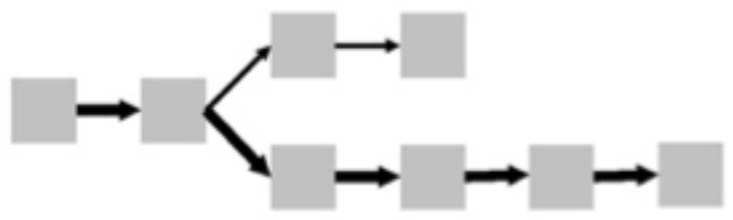
# K-mer graphs - branching



For decisions about which paths etc, biology-based heuristics come into play as well.

## K-mer graph complexity - spur

(a)



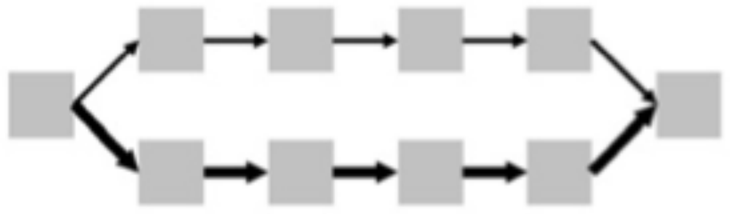
(Short) dead-end in graph.

Can be caused by error at the end of some overlapping reads, or low coverage

J.R. Miller et al. / Genomics (2010)

## K-mer graph complexity - bubble

(b)



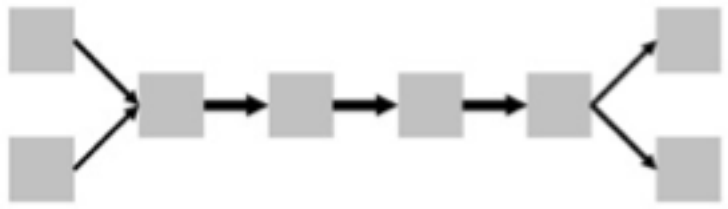
Multiple parallel paths that diverge and join.

Caused by sequencing error and true polymorphism / polyploidy in sample.

J.R. Miller et al. / Genomics (2010)

## K-mer graph complexity – “frayed rope”

(c)



Converging, then diverging paths.

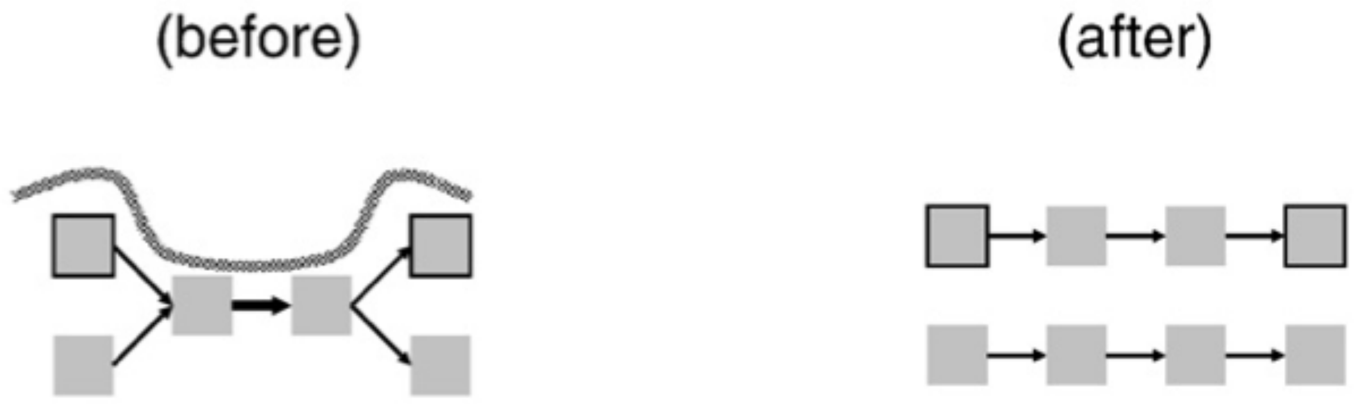
Caused by repetitive sequences.

J.R. Miller et al. / Genomics (2010)

## Resolving graph complexity

- Primarily heuristic (approximate) approaches.
- Detecting complex graph structures can generally not be done efficiently.
- Much of the divergence in functionality of new assemblers comes from this.
- Three examples:

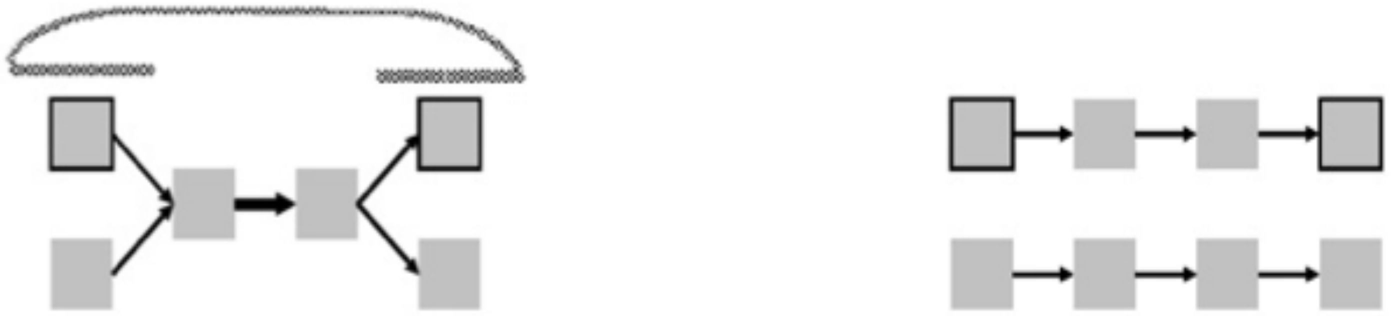
# Read threading



Single read spans k-mer graph => extract the single-read path.

J.R. Miller et al. / Genomics (2010)

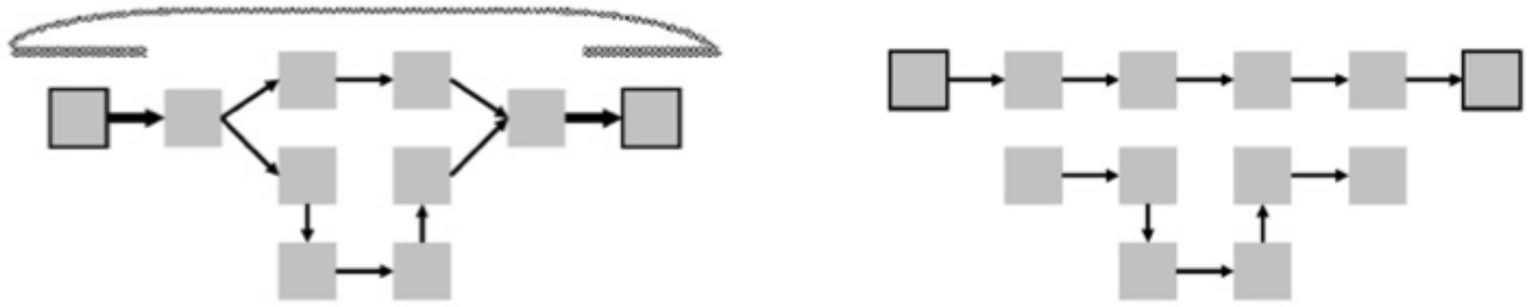
# Mate threading



Resolve "frayed-rope" pattern caused by repeats, by separating paths based on mate-pair reads.

J.R. Miller et al. / Genomics (2010)

# Path following



Reject inconsistent paths based on mate-pair reads and insert size.

J.R. Miller et al. / Genomics (2010)

## More assembly issues

- Many parameters to optimize!
- RNAseq has variation in copy number; naïve assemblers can treat this as repetitive and eliminate it.
- Some assemblers require gobs of memory (4 lanes, 60m reads => ~ 150gb RAM)
- How do we evaluate assemblies?
  - What's the best assembler?

## K-mer based assemblers scale poorly

Why do big data sets require big machines??

Memory usage ~ “real” variation + number of errors

Number of errors ~ size of data set

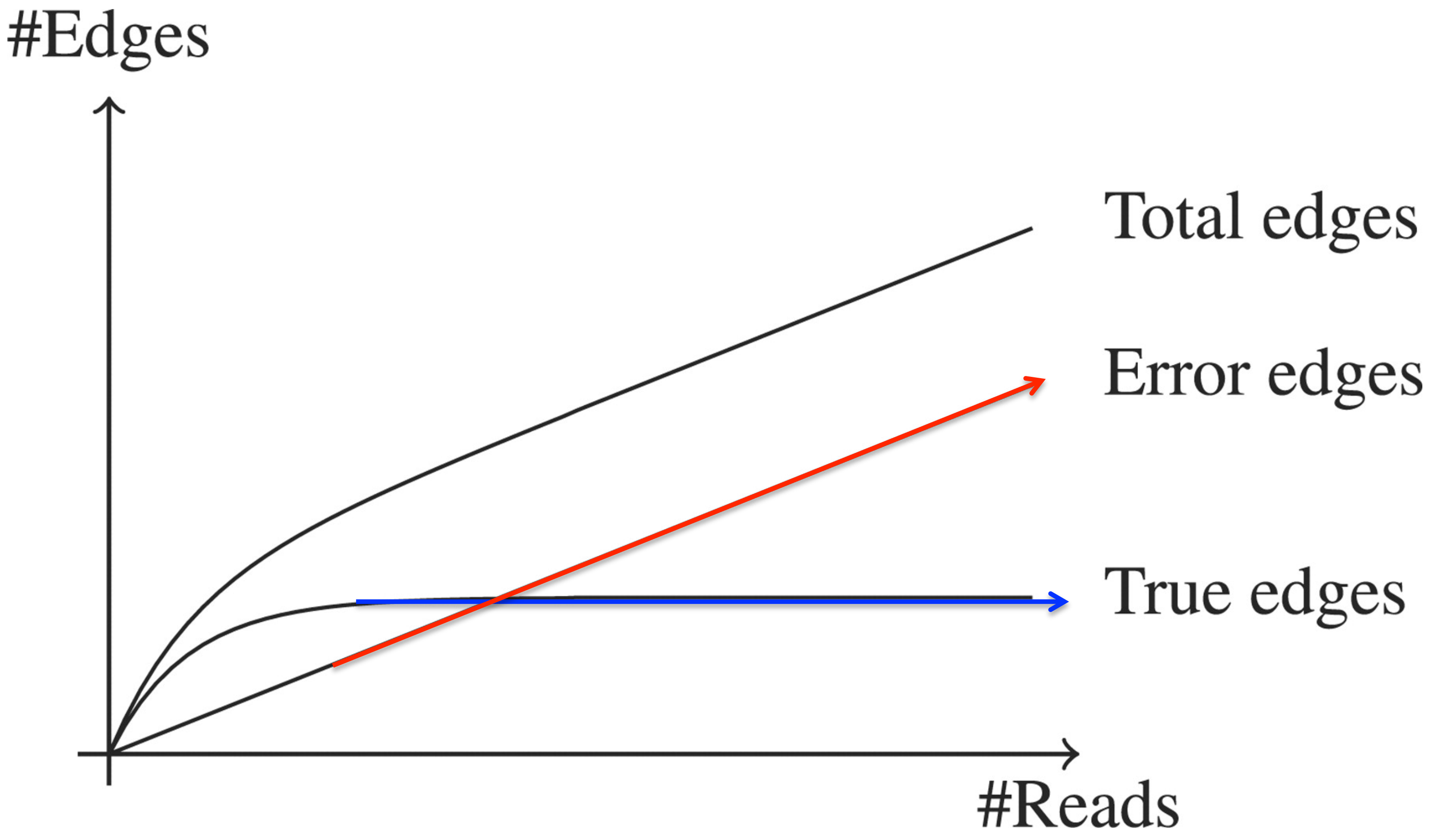
GCGTCAGGTAG**C**AGACCACCGCCATGGCGACGATG

GCGTCAGGTAGGAGACCACCG**T**CATGGCGACGATG

GCGT**T**AGGTAGGAGACCACCGCCATGGCGACGATG

GCGTCAGGTAGGAGACC**G**CCGCCATGGCGACGATG

## De Bruijn graphs scale poorly with erroneous data

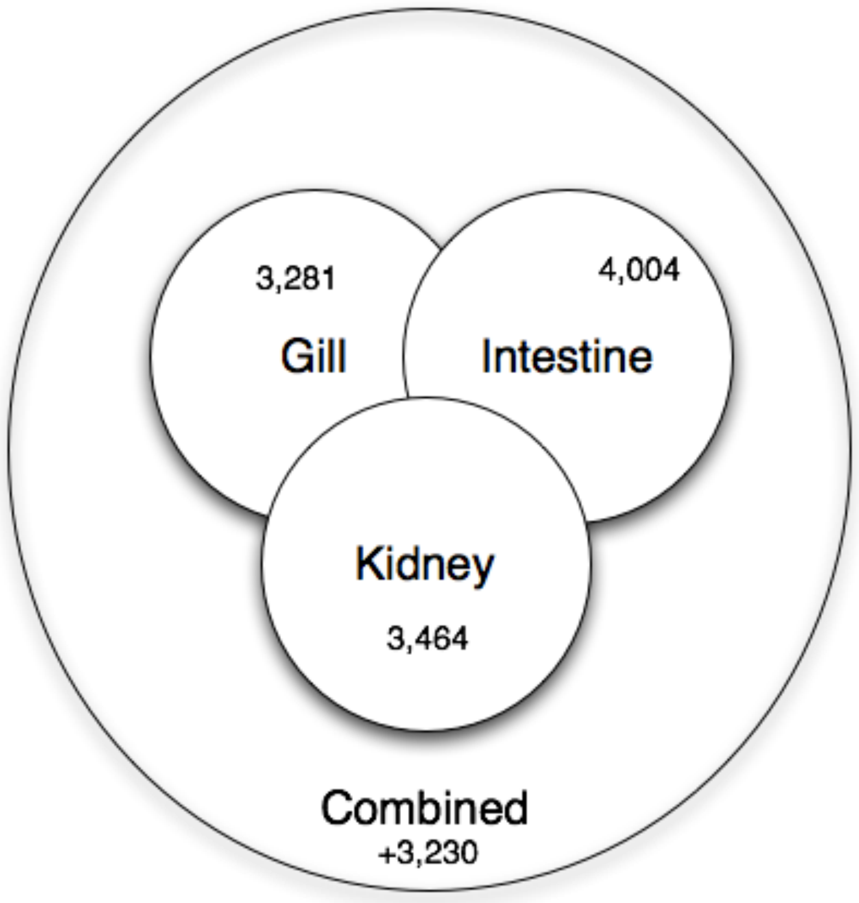


**Conway T C , Bromage A J Bioinformatics 2011;27:479-486**

Bioinformatics

## Co-assembly is important for sensitivity

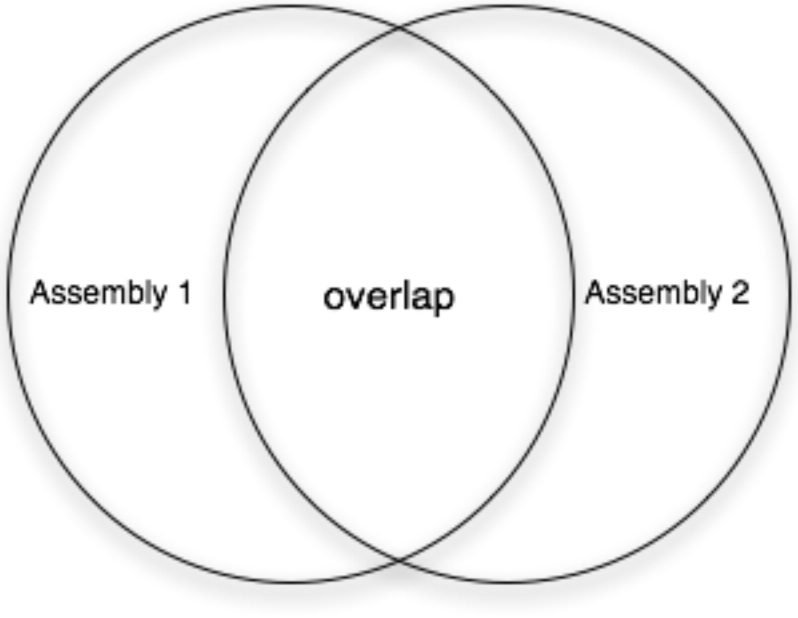
Shared low-level transcripts may not reach the threshold for assembly.



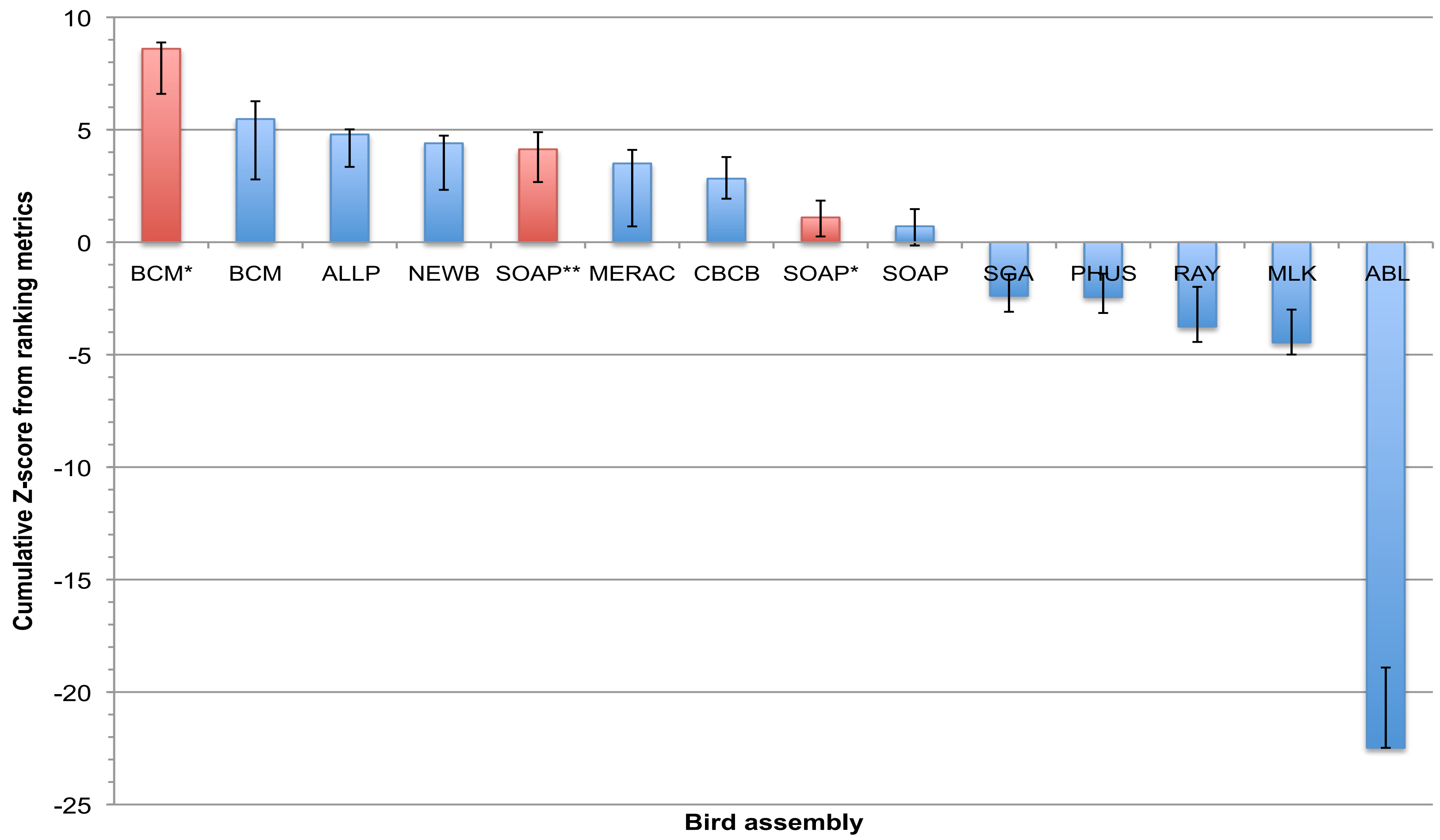
## Is your assembly good?

- For genomes, N50 is an OK measure:
  - “50% or more of the genome is in contigs > this number”
- That assumes your contigs are correct…!
- What about mRNA and metagenomes??
- **Truly reference-free assembly is hard to evaluate.**

## How do you compare assemblies?

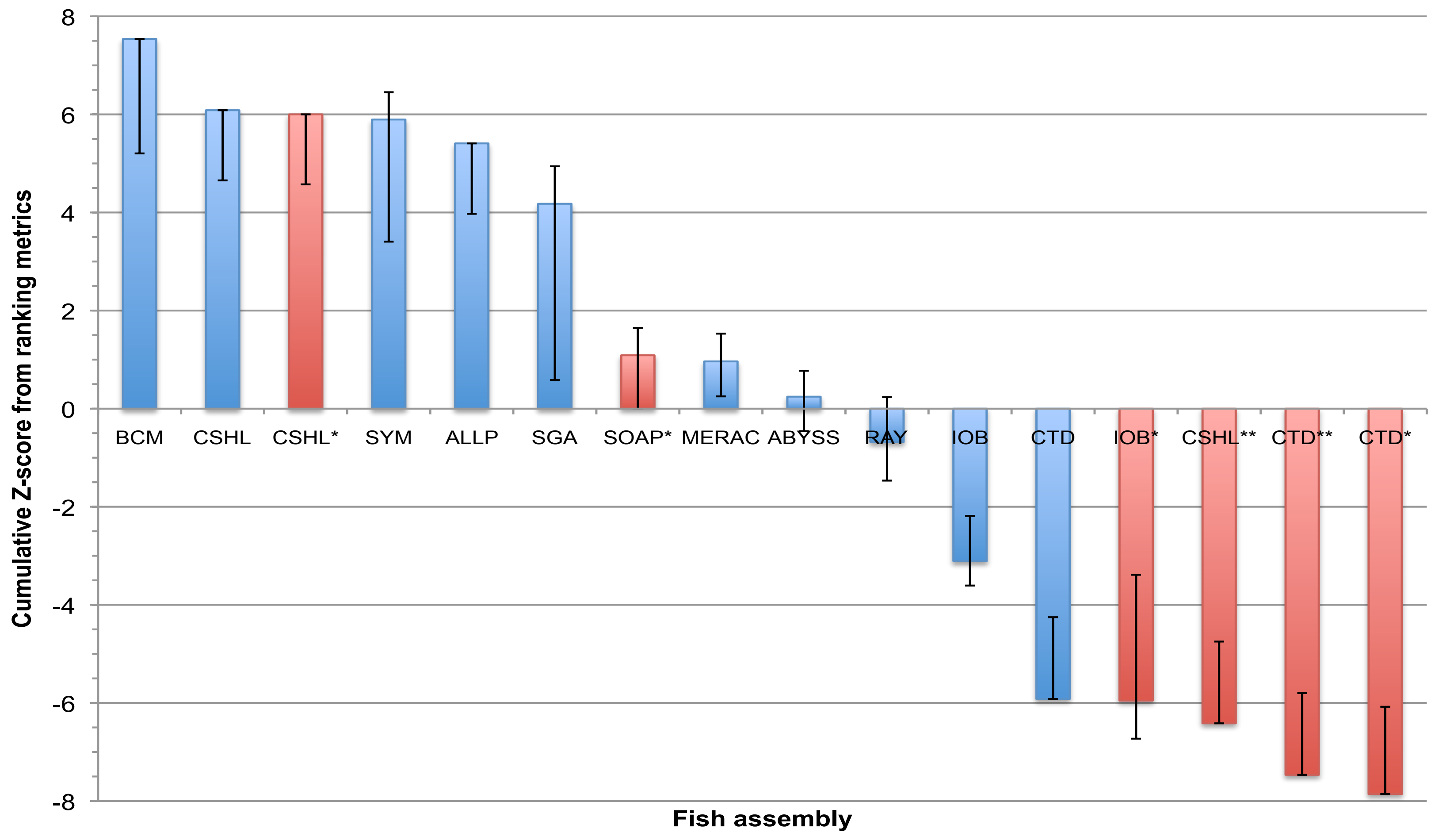


# What’s the best assembler?



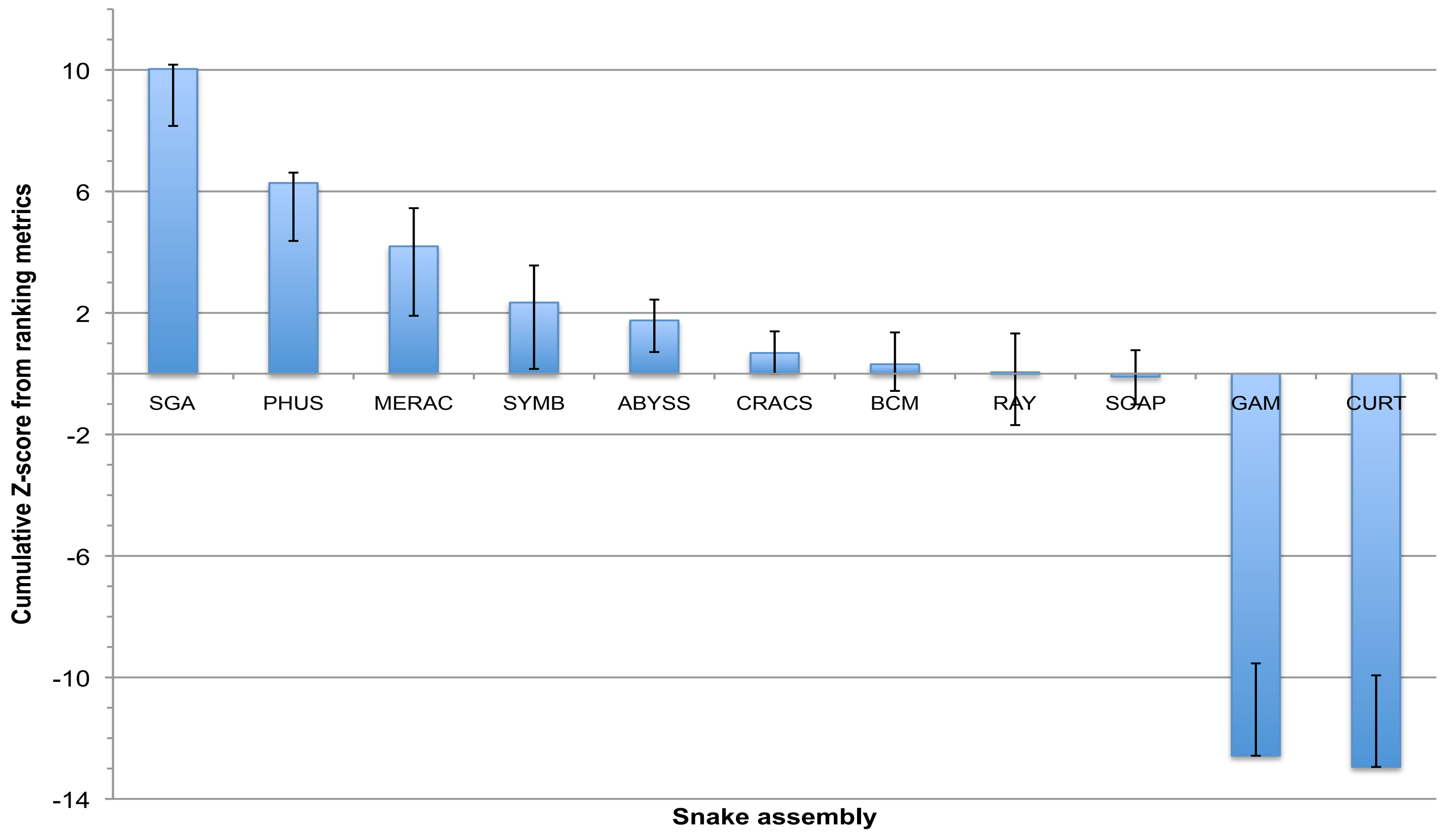
Bradnam et al., Assemblathon 2:
http://arxiv.org/pdf/1301.5406v1.pdf

# What's the best assembler?



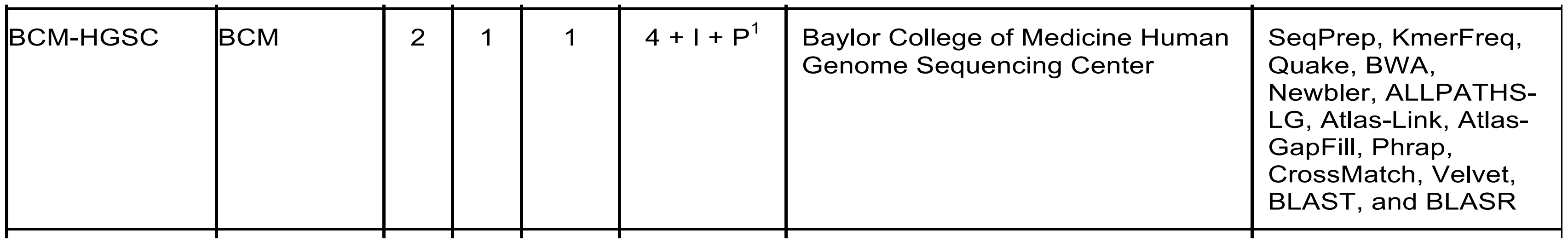
Bradnam et al., Assemblathon 2:
http://arxiv.org/pdf/1301.5406v1.pdf

# What's the best assembler?



Bradnam et al., Assemblathon 2:
http://arxiv.org/pdf/1301.5406v1.pdf

## Note: the teams mostly used *multiple* software packages

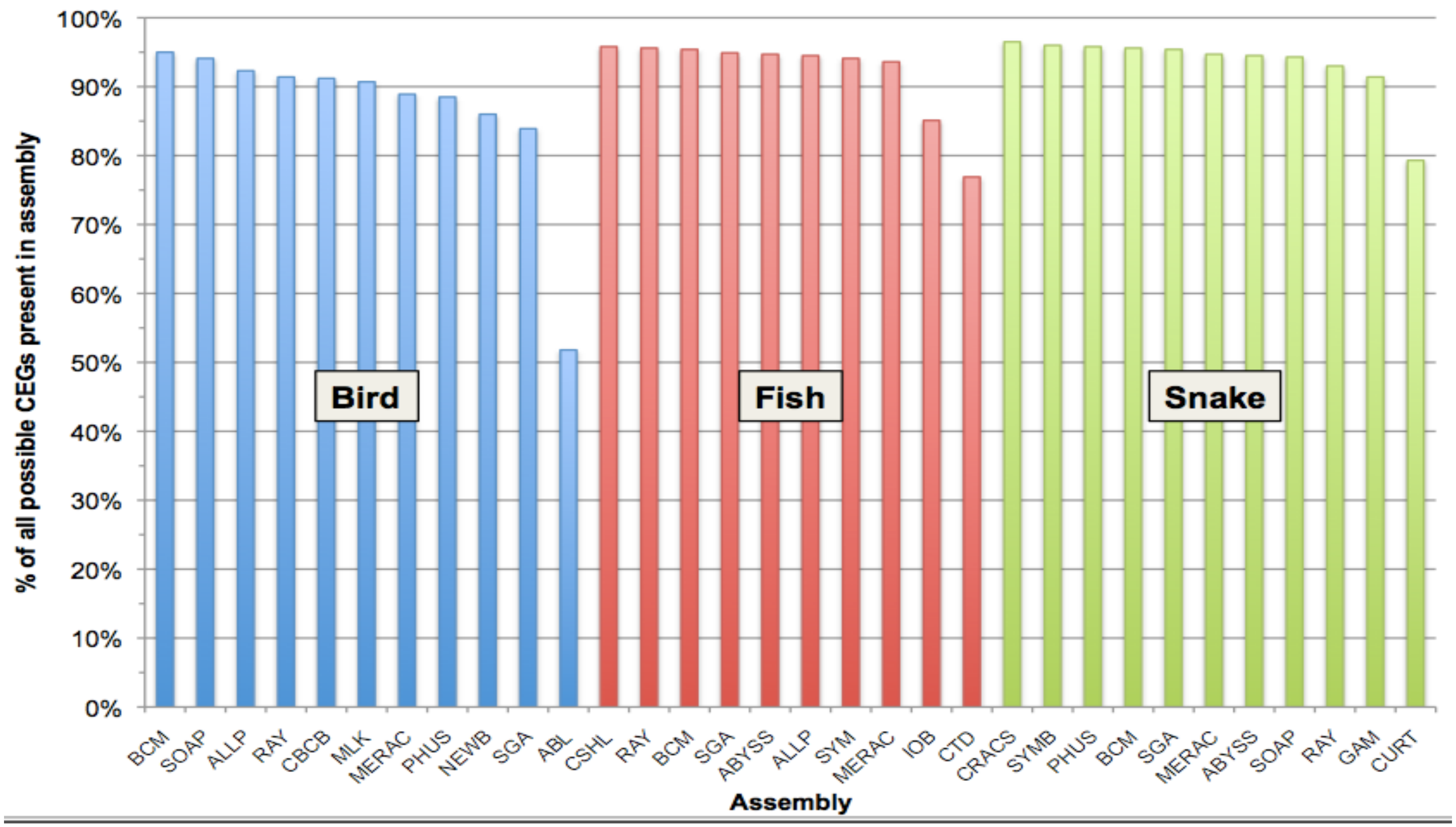


| | | | | | | | |
|---|---|---|---|---|---|---|---|
| BCM-HGSC | BCM | 2 | 1 | 1 | 4 + I + P[1] | Baylor College of Medicine Human Genome Sequencing Center | SeqPrep, KmerFreq, Quake, BWA, Newbler, ALLPATHS-LG, Atlas-Link, Atlas-GapFill, Phrap, CrossMatch, Velvet, BLAST, and BLASR |

## Answer: it depends

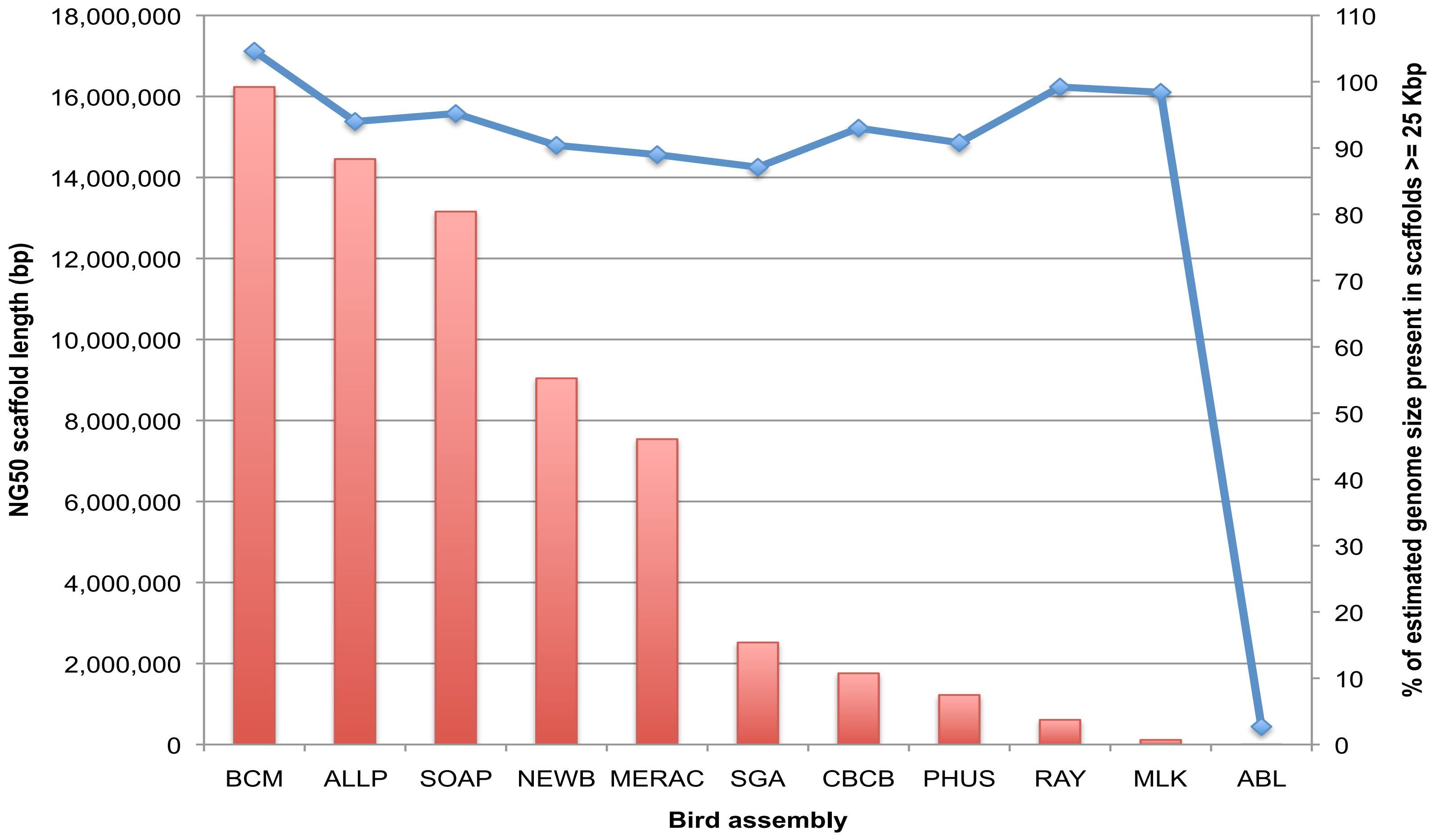
- Different assemblers perform differently, depending on
  - Repeat content
  - Heterozygosity
- Generally the results are very good (est completeness, etc.) but *different* between different assemblers (!)
- There Is No One Answer.

# Estimated completeness: CEGMA



Each assembler lost *different* ~5% CEGs

# Tradeoffs in N 50 and % incl.



## Practical issues

- Do you have enough memory?
- Trim vs use quality scores?
- When is your assembly as good as it gets?
- Paired-end vs longer reads?

- More data is not *necessarily* better, if it introduces more errors.

## Practical issues

- Many bacterial genomes can be completely assembled with a combination of PacBio and Illumina.

  (see: http://arxiv.org/abs/1304.3752)

- As soon as repeats, heterozygosity, and GC variation enter the picture, all bets are off (eukaryotes are trouble!)

## Mapping & assembly

- Assembly and mapping (and variations thereof) are the two basic approaches used to deal with next-gen sequencing data.

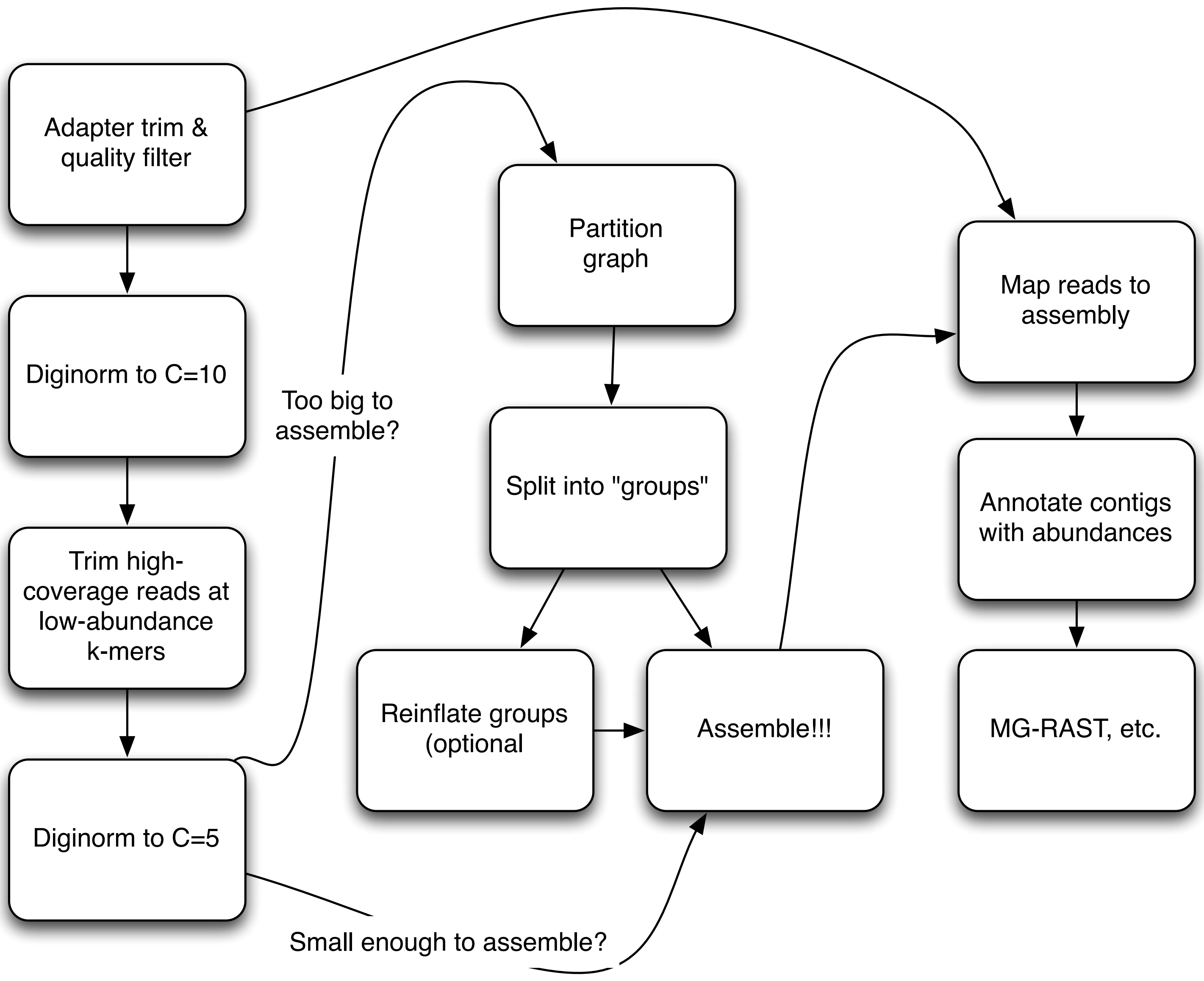
- I’ll be showing both in the next ~5 weeks.

## Protocols

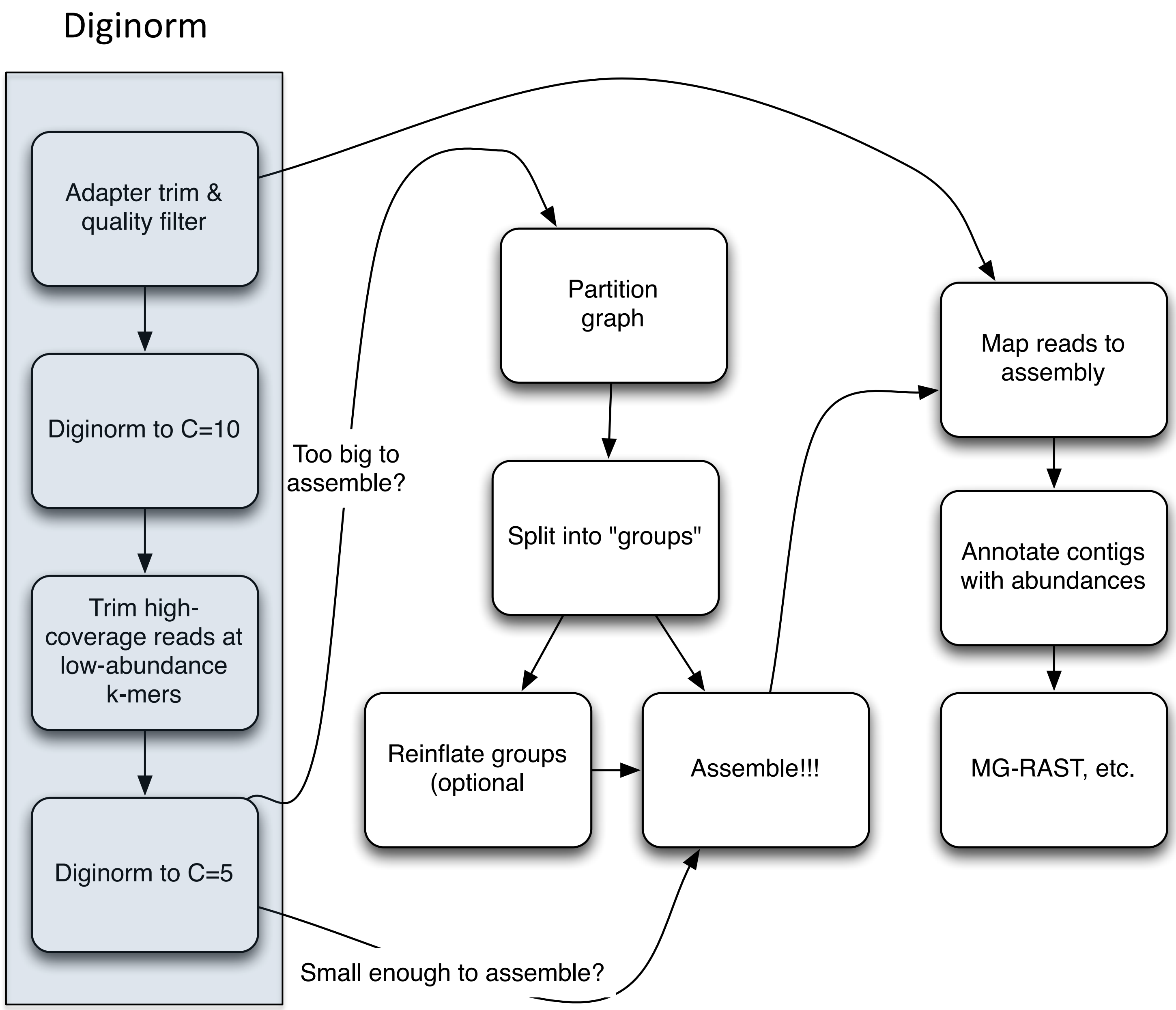
https://khmer-protocols.readthedocs.org/

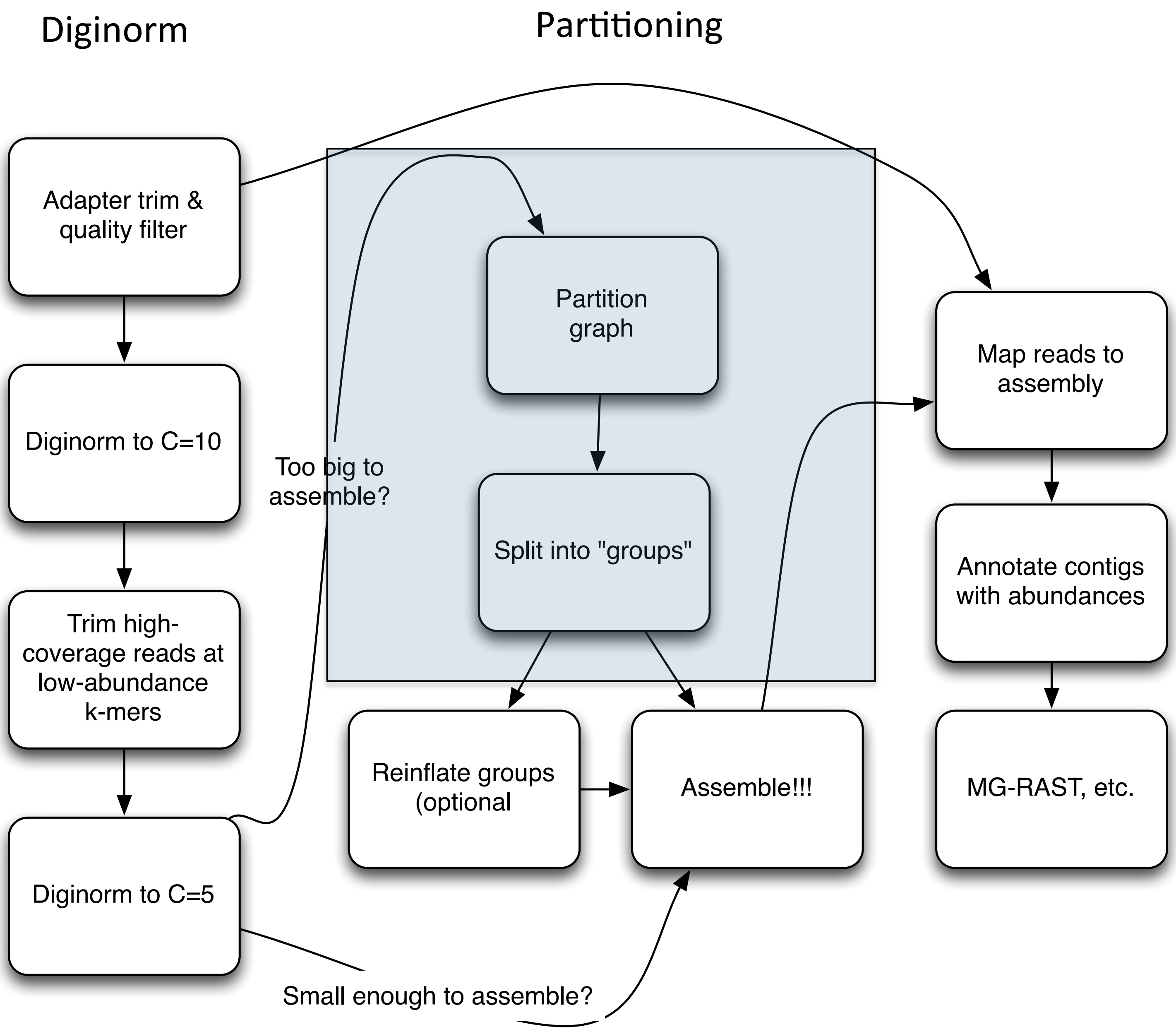
Effort to make things transparent & reproducible.

# The Kalamazoo Metagenome Assembly Pipeline



Adapter trim & quality filter
Diginorm to C=10
Trim high-coverage reads at low-abundance k-mers
Diginorm to C=5
Too big to assemble?
Partition graph
Split into "groups"
Reinflate groups (optional
Assemble!!!
Small enough to assemble?
Map reads to assembly
Annotate contigs with abundances
MG-RAST, etc.


Diginorm
Adapter trim & quality filter
Diginorm to C=10
Trim high-coverage reads at low-abundance k-mers
Diginorm to C=5
Too big to assemble?
Small enough to assemble?
Partition graph
Split into "groups"
Reinflate groups (optional
Assemble!!!
Map reads to assembly
Annotate contigs with abundances
MG-RAST, etc.

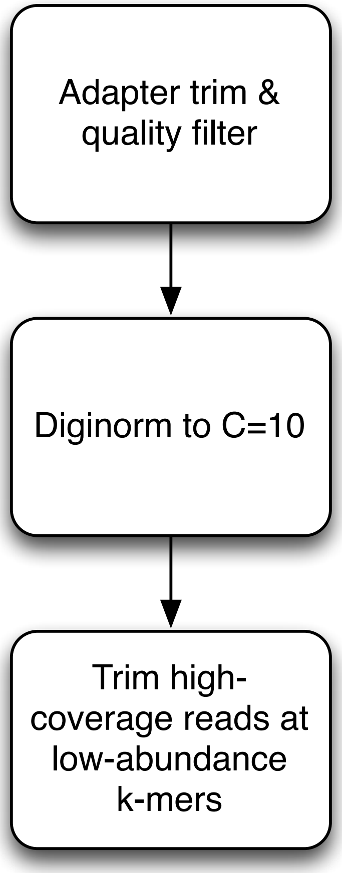

Diginorm
Partitioning
Adapter trim & quality filter
Diginorm to C=10
Trim high-coverage reads at low-abundance k-mers
Diginorm to C=5
Too big to assemble?
Small enough to assemble?
Partition graph
Split into "groups"
Reinflate groups (optional
Assemble!!!
Map reads to assembly
Annotate contigs with abundances
MG-RAST, etc.

## Deep Carbon data set

- Name: DCO_TCO_MM5
- Masimong Gold Mine; microbial cells filtered from fracture water from within a 1.9km borehole. (32,000 year old water?!)
- M.C.Y. Lau, C. Magnabosco, S. Grim, G. Lacrampe Couloume, K. Wilkie, B. Sherwood Lollar, D.N. Simkus, G.F. Slater, S. Hendrickson, M. Pullin, T.L. Kieft, O. Kuloyo, B. Linage, G. Borgonie, E. van Heerden, J. Ackerman, C. van Jaarsveld, and T.C. Onstott

## DCO_TCO_MM5

20m reads / 2.1 Gbp



Adapter trim & quality filter
Diginorm to C=10
Trim high-coverage reads at low-abundance k-mers

5.6m reads / 601.3 Mbp

"Could you take a look at this? MG-RAST is telling us we have a lot of artificially duplicated reads, i.e. the data is bad."

Entire process took ~4 hours of computation, or so.

(Minimum genome size est: 60.1 Mbp)

DCO_TCO_MM5

# Assembly stats:

| | All contigs | | Contigs > 1kb | | |
|---|---|---|---|---|---|
| *k* | *N contigs* | *Sum BP* | *N contigs* | *Sum BP* | *Max contig* |
| 21 | 343263 | 63217837 | 6271 | 10537601 | 9987 |
| 23 | 302613 | 63025311 | 7183 | 13867885 | 21348 |
| 25 | 276261 | 62874727 | 7375 | 15303646 | 34272 |
| 27 | 258073 | 62500739 | 7424 | 16078145 | 48742 |
| 29 | 242552 | 62001315 | 7349 | 16426147 | 48746 |
| 31 | 228043 | 61445912 | 7307 | 16864293 | 48750 |
| 33 | 214559 | 60744478 | 7241 | 17133827 | 48768 |
| 35 | 203292 | 60039871 | 7129 | 17249351 | 45446 |
| 37 | 189948 | 58899828 | 7088 | 17527450 | 59437 |
| 39 | 180754 | 58146806 | 7027 | 17610071 | 54112 |
| 41 | 172209 | 57126650 | 6914 | 17551789 | 65207 |
| 43 | 165563 | 56440648 | 6925 | 17654067 | 73231 |

(Minimum genome size est: 60.1 Mbp)

DCO_TCO_MM5

## Chose two:

- A: k=43 (“long contigs”)
- 165563 contigs
- 56.4 Mbp
- longest contig: **73231** bp

How to evaluate??

- B: k=43 (“high recall”)
- 343263 contigs
- **63.2** Mbp
- longest contig is 9987 bp

# How many reads map back?

Mapped 3.8m paired-end reads (one subsample):

- high-recall: 41% of pairs map
- **longer-contigs: 70% of pairs map**

+ 150k single-end reads:

- high-recall: 49% of sequences map
- **longer-contigs: 79% of sequences map**

DCO_TCO_MM5

## Conclusion

- **This is a pretty good metagenome assembly – > 80% of reads map!**

- Surprised that the larger dataset (6.32 Mbp, “high recall”) accounts for a smaller percentage of the reads – 49% vs 79% for the 56.4 Mbp “long contigs” data set.

- Are different parameters recovering different portions of the data set?

## CSE 801

- First 5 weeks: NGS assembly, mapping, etc.
- Second 5 weeks: Modeling & simulations; agent-based stuff.
- All 10 weeks: intro programming in Python.