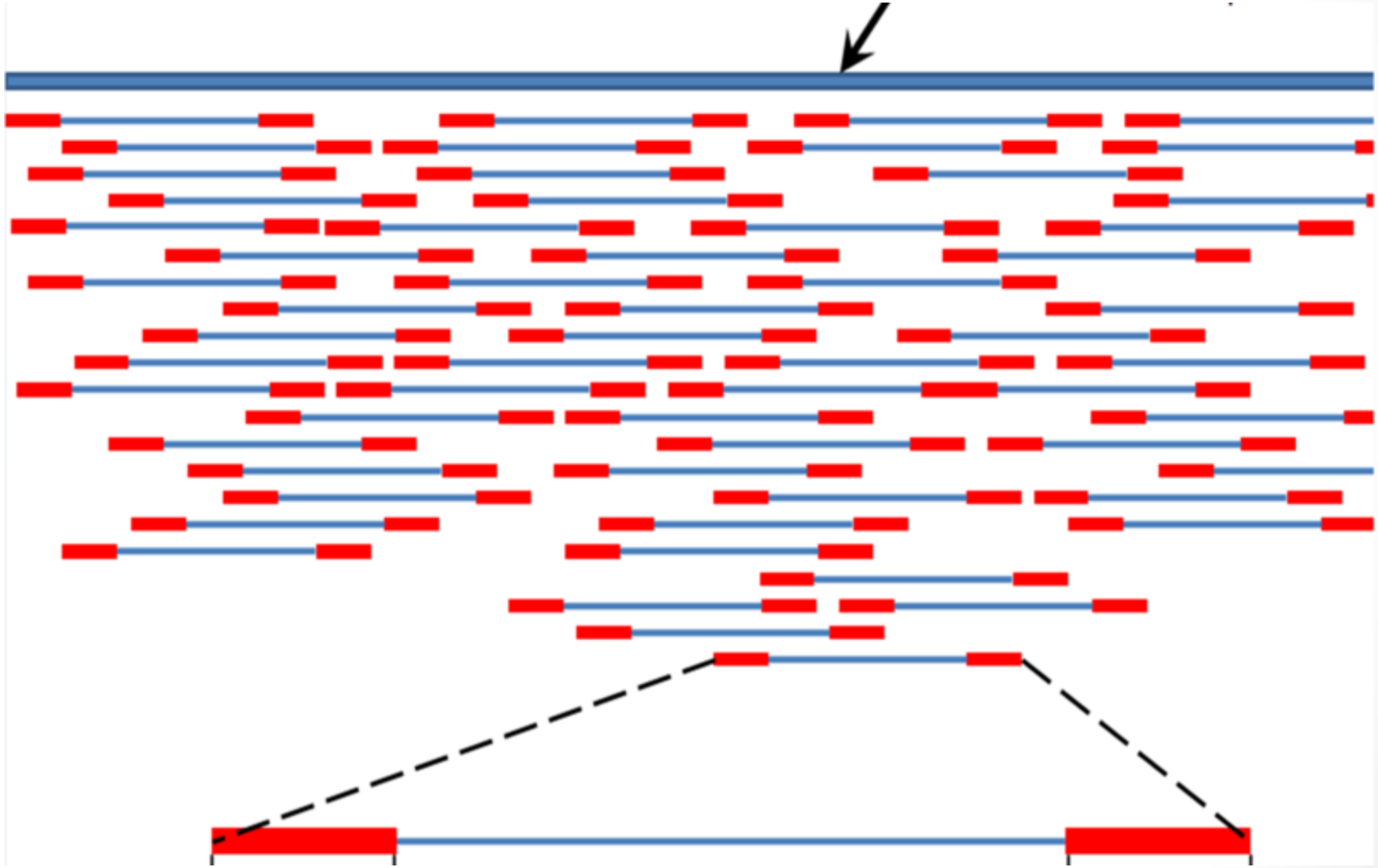
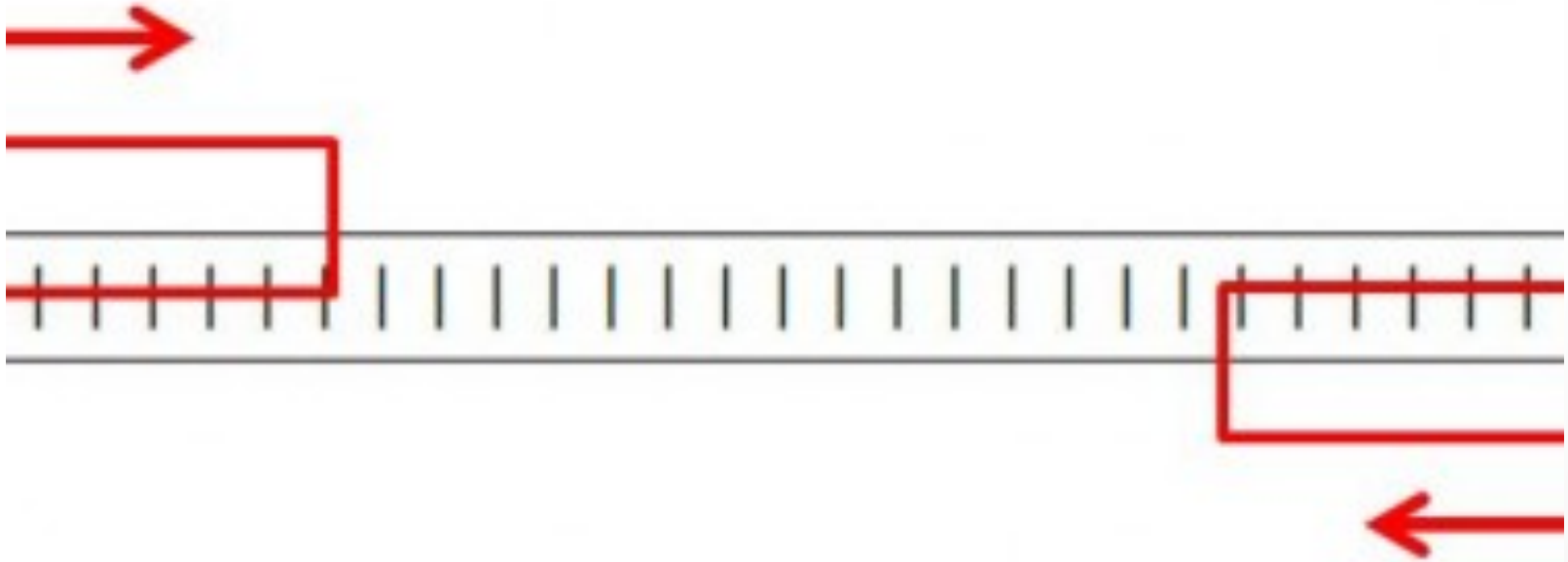
Mapping short reads

Locate reads in ref genome



http://en.wikipedia.org/wiki/File:Mapping_Reads.png



http://www.cureffi.org/2012/12/19/forward-and-reverse-reads-in-paired-end-sequencing/

SNP calling – which variants are "real"?

http://www.kenkraaijeveld.nl/genomics/bioinformatics/

Note: long v short

- Mapping *long* reads is a different problem from mapping short reads.

- This is for two reasons, both of them pragmatic/practical:
  - The volume of data has traditionally been much less: 1m 454 reads vs 200m Illumina
  - Long reads are much more likely to have insertions or deletions in them

Long reads: BLAST vs 'blat'

- BLAST is not the right tool.
  - BLAST requires that a query sequence contains the same 11-mer as a database sequence before it attempts further alignment.
  - Any given 11-mer occurs only once in 2m sequences, so this filters out many database sequences quickly.
  - You can also store the list of all possible 11-mers in memory easily (~2mb), making it possible to keep track of everything quickly.
- 'blat' does the same thing as BLAST, but is faster because it uses longer k-mers.

How *alignment* works, and why indels are the devil

There are many alignment strategies, but most work like this:

```
GCGGAGatggac        GCGGAGatggac
||||||......  =>    ||||||x.....
GCGGAGgcggac        GCGGAGgcggac
```

At each base, try extending alignment; is total score still above threshold?

How *alignment* works, and why indels are the devil

There are many alignment strategies, but most work like this:

```
GCGGAGatggac        GCGGAGatggac
||||||......  =>    ||||||xx....
GCGGAGgcggac        GCGGAGgcggac
```

Each mismatch *costs*.

How *alignment* works, and why indels are the devil

Insertions/deletions introduce *lots* more ambiguity:

```
GCGGAGagaccaacc          GCGGAGag-accaacc
||||||              =>  |||||||
GCGGAGggaaccacc          GCGGAGggaacc-acc

GCGGAGagaccaacc          GCGGAGaga-ccaacc
||||||              =>  |||||||
GCGGAGggaaccacc          GCGGAGggaacca-cc
```

Mapping short reads, again

- What's hard about mapping
- Some mapping programs
- Decisions to be made
- Color space issues

Mapping, defined

- Exhibit A: 20m+ reads from genome/transcriptome.
- Exhibit B: related genome/transcriptome, aka “the reference”

- Goal: assign all reads to location(s) within reference.

- Req’d for resequencing, ChIP-seq, and mRNAseq

Want *global*, not *local*, alignment

- You do not want matches *within* the read, like BLAST would produce.

- Do not use BLAST!

Mapping is "pleasantly parallel"

- Goal is to assign each individual read to location(s) within the genome.

- So, you can map each read *separately*.

What makes mapping challenging?

- Volume of data
- Garbage reads
- Errors in reads, and quality scores
- Repeat elements and multicopy sequence
- SNPs/SNVs
- Indels
- Splicing (transcriptome)

Volume of data

- Size of reference genome is not a problem: you can load essentially any genome into memory (~3 gb).

- However, doing *any* complicated process 20m times is generally going to require optimization!

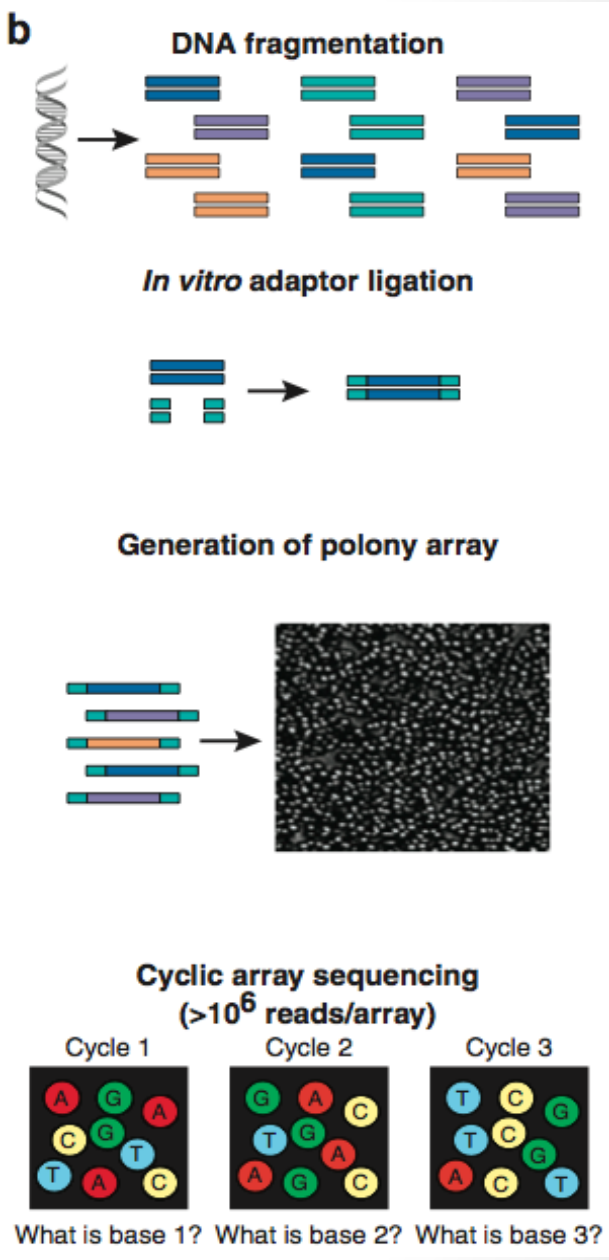
Garbage reads

Overlapping polonies result in mixed signals.

These reads will not map to anything!

Used to be ~40% of data.

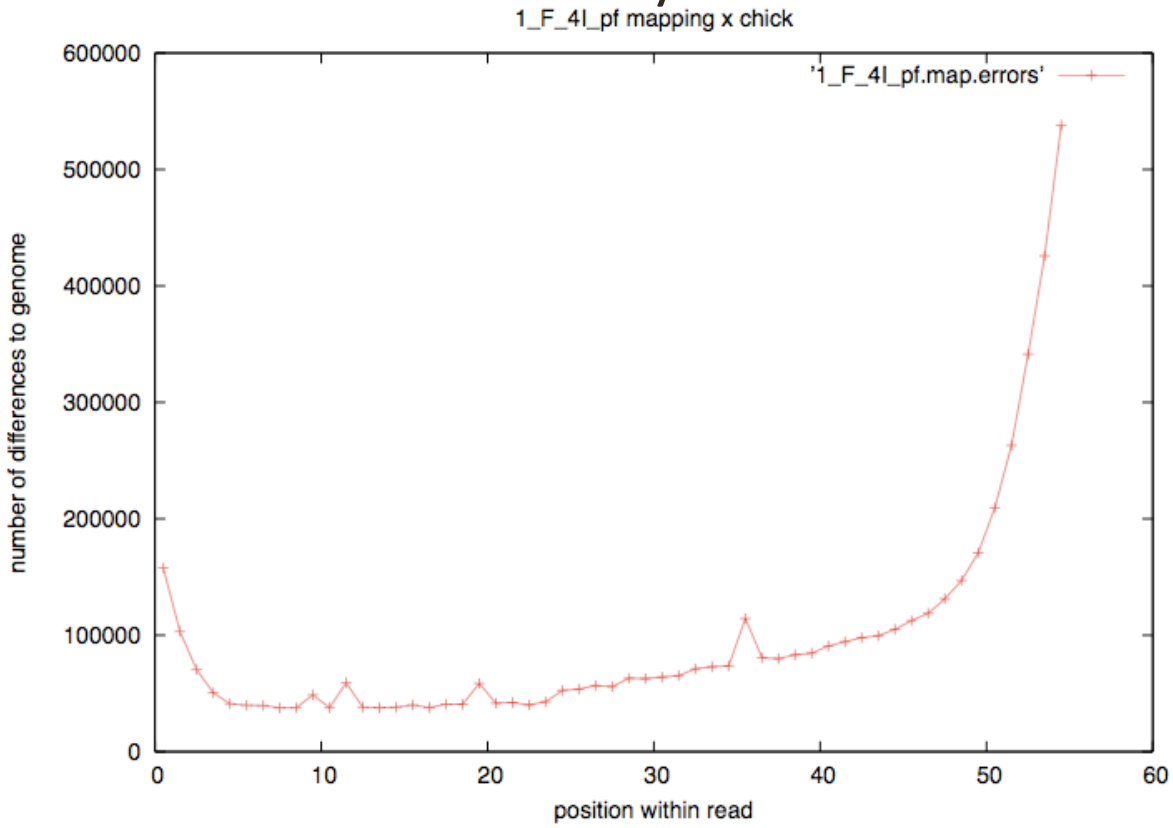
Increasingly, filtered out by sequencing software.



Shendure and Ji, Nat Biotech, 2008

Errors in reads

When mapping, a mismatch is not necessarily “real”.



Rule of thumb: anything that varies by position within read is NOT REAL!

Errors in reads

- Quality scores are based on Sanger sequencing-style quality scores: per base.

- But 454 & Ion/Proton data are subject to different biases than Illumina, and the biases are not necessarily base-by-base (think: homopolymer runs)

Repeat/multi-copy elements

- Multi-copy sequence makes it impossible to map all reads uniquely.

- Repeats are particularly bad, because there are (a) lots of them and (b) they vary in sequence. They therefore may "attract" reads depending on what optimizations/heuristics you use.

SNP/SNVs

- Genuine mismatches between reference and sequence *do* exist, of course.
  - Polymorphism
  - Diploidy
  - Population studies
- You want to map these reads!
- Fast heuristic approaches exist, based on fuzzy matching.
- However, they are still biased towards mapping exact matches.
  - This can be a problem for allelotyping and population studies.
  - Likit will discuss next week.

Indels

- Remember, they are the devil:
- Complicate mapping heuristics
- Complicate *statistics*

Indels: ambiguity & decisions…

```
TGACGATATGGCGATGGACTGGACG
|x|||||||||||x||x|x||||||
TcACGATATGGCGgTGaA-TGGACG

TGACGATATGGCGATGGACTGGACG
|x|||||||||||x|x||x||||||
TcACGATATGGCGgT-GAaTGGACG
```

Splice sites

- If you are mapping transcriptome reads to the genome, your reference sequence is different from your source sequence!
- This is a problem if you don't have a really good annotation!

- Main technique: try to map across splice sites, build new exon models.
- Another technique: assembly.

Two specific mapping programs

- Bowtie
- BWA

Both open source.

BWA is widely used now, so we'll use that for examples.

(There are many more, too.)

Bowtie1

- Not indel-capable.

- Designed for:
  - Many reads have one good, valid alignment
  - Many reads are high quality
  - Small number of alignments/read

a.k.a. “sweet spot” :)

BWA

- Uses similar strategy to Bowtie, but does gapped alignment.
- Newest, hottest tool.
- Written by the Mapping God, Heng Li
  (Istvan Albert's scientific crush)

Decisions to be made by you

- How many mismatches to allow?
  - Vary depending on biology & completeness of reference genome
- Report how many matches?
  - Are you interested in multiple matches?
- Require best match, or first/any that fit criteria?
  - It can be much faster to find *first* match that fits your criteria.

All of these decisions affect your results and how you treat your data.

Mapping best done on *entire reference*

- May be tempted to optimize by doing mapping to one chr, etc. “just to see what happens”
- Don’t.
- Simple reason: if you allow mismatches, then many of your reads will match erroneously to what’s in the chr you chose.

Look at your mapping

Just like statistics, always look at your “raw data” ☺

We’ll do some of that today.

Two considerations in mapping

- Building an index
  - Prepares your “reference”
  - (Not really a big deal for single microbial genomes)

Indexing – e.g. BLAST

BLASTN filters sequences for exact matches between “words” of length 11:

```
GAGGGTATGACGATATGGCGATGGAC
||x|||||x||||||||||||x|x||x
GAcGGTATcACGATATGGCGgT-Gag
```

What the ‘formatdb’ command does (see Tuesday’s first tutorial) is *build an index* (“index”) sequences by their 11-base word content – a “reverse index” of sorts.

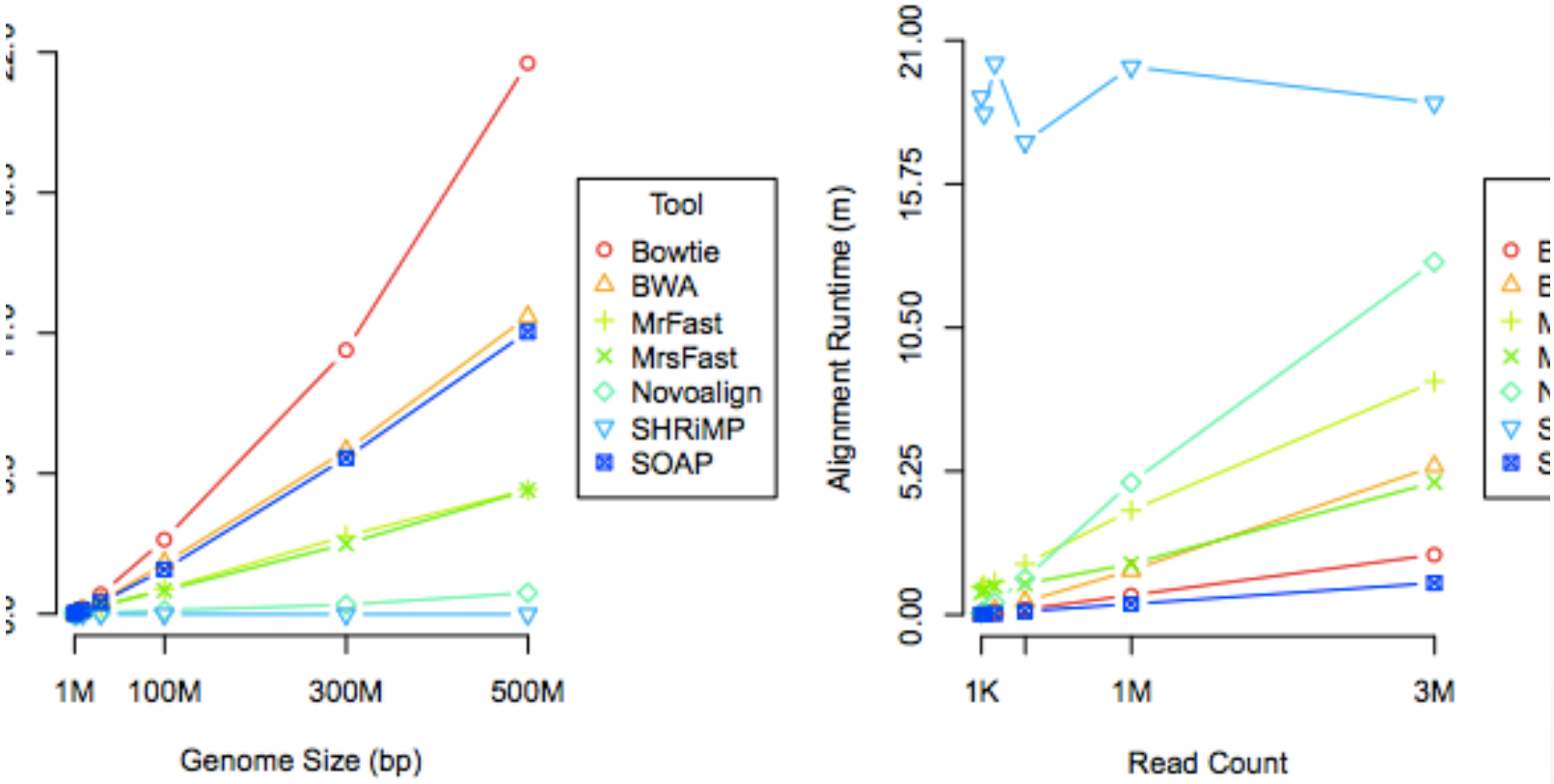
Indexing – e.g. BLAST

What the 'formatdb' command does (see Tuesday's BLAST tutorial) is *build an index* ("index") sequences by their 11-base word content – a "reverse index" of sorts.

Since this index only needs to be built once for each reference, it can be slower to build – what matters to most people is *mapping speed.*

All short-read mappers have an indexing step.

Speed of indexing & mapping.



ıeasurements: (a) shows indexing time vs. genome size, (b) shows alignment time vs. read count c

Fig 5 of Ruffalo et al. PMID 21856737, Bioinformatics 2011.

Simulations => understanding mappers

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.; http://arxiv.org/abs/1303.2411

Does choice of mapper matter?

Not in our experience.

Reference completeness/quality matters more!

nparison of Three Common Mapping Programs on the Same Chicke

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

Bowtie, BWA, and SOAP2 mapping programs on the same simulated reads fo
ts (triplicate and averaged) with decreasing completeness of the reference tran
lent results.

Pyrkosz et al., unpub.; http://arxiv.org/abs/1303.2411

Misc points

- Transcriptomes and bacterial genomes have very few repeats…
- …but for transcriptomes, you need to think about shared exons.

- For genotyping/association studies/ASE, you may not care about indels too much.

- Variant calling is less sensitive to coverage than assembly (20x vs 100x)

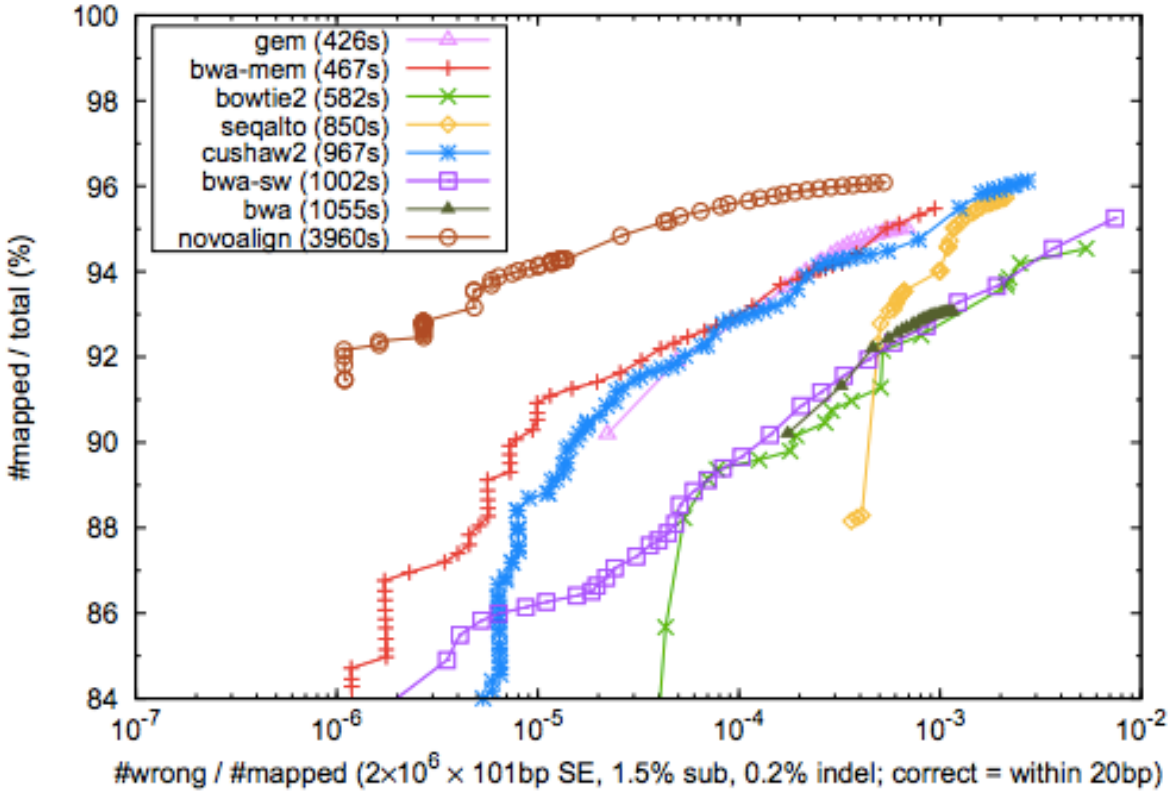
Using quality scores?

- Bowtie uses quality scores; bwa does not.

- This means that bowtie can align some things in FASTQ that cannot be aligned in FASTA.

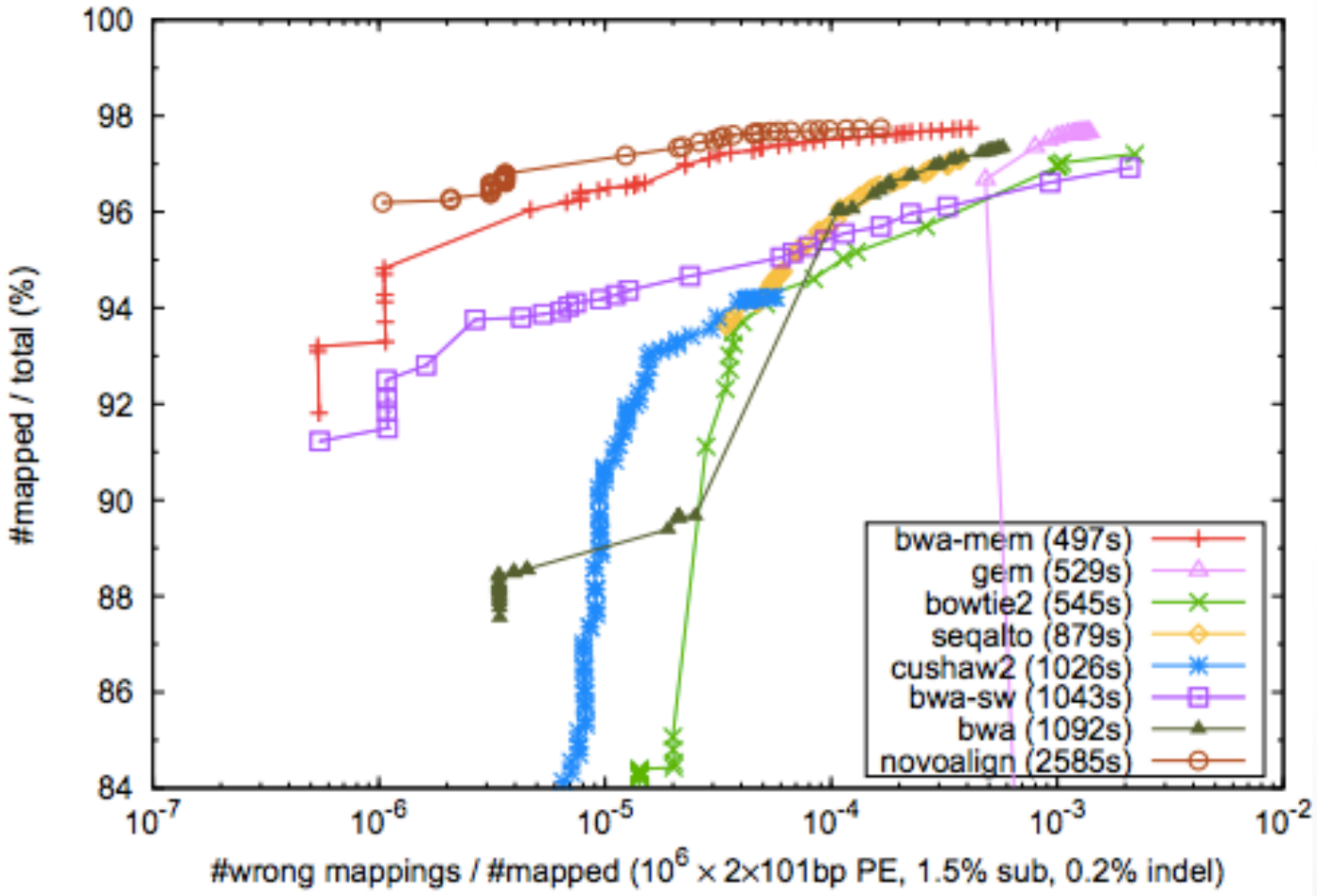
See: http://www.homolog.us/blogs/blog/2012/02/28/bowtie-alignment-with-and-without-quality-score/

Comparative performance/SE



Heng Li, BWA-MEM: http://arxiv.org/pdf/1303.3997v2.pdf

Comparative performance/PE



Heng Li, BWA-MEM: http://arxiv.org/pdf/1303.3997v2.pdf

Part II:
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!

Assemble based on word overlaps:

the quick brown fox jumped

jumped over the lazy dog

the quick brown fox jumped over the lazy dog

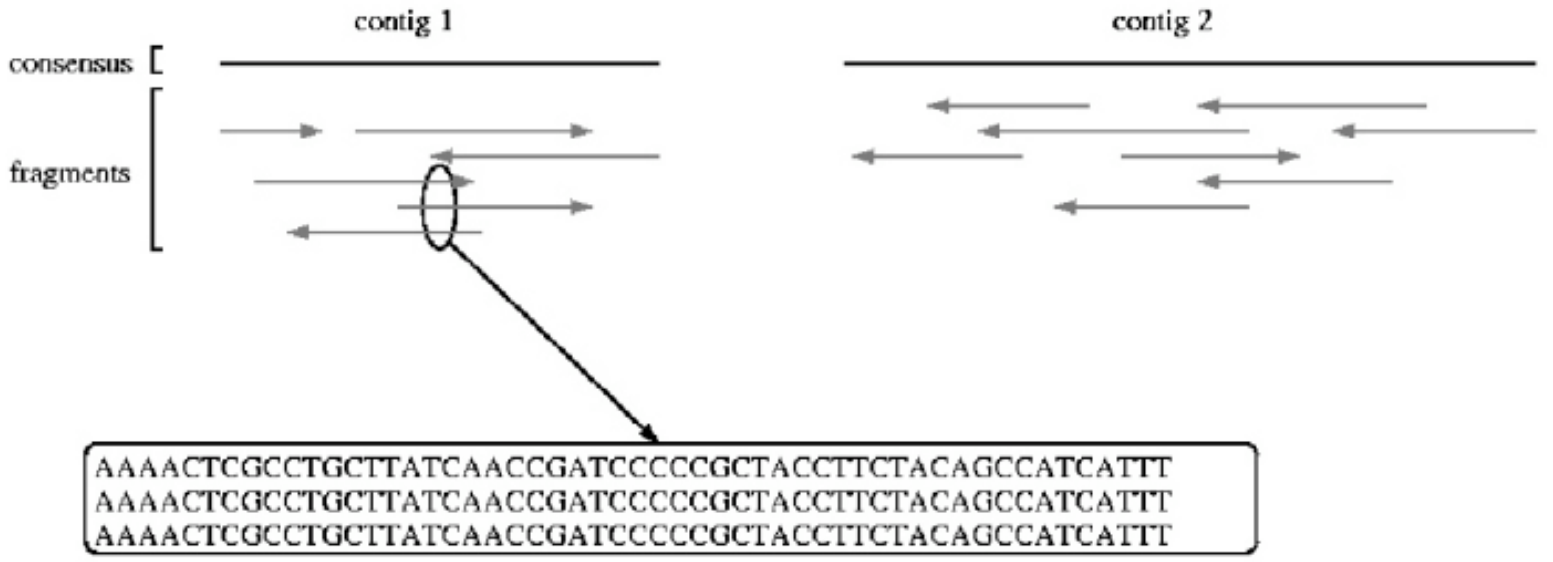
Repeats do cause problems:

my chemical romance: na na na

na na na, batman!

Shotgun sequencing & assembly

Randomly fragment & sequence from DNA;
reassemble computationally.



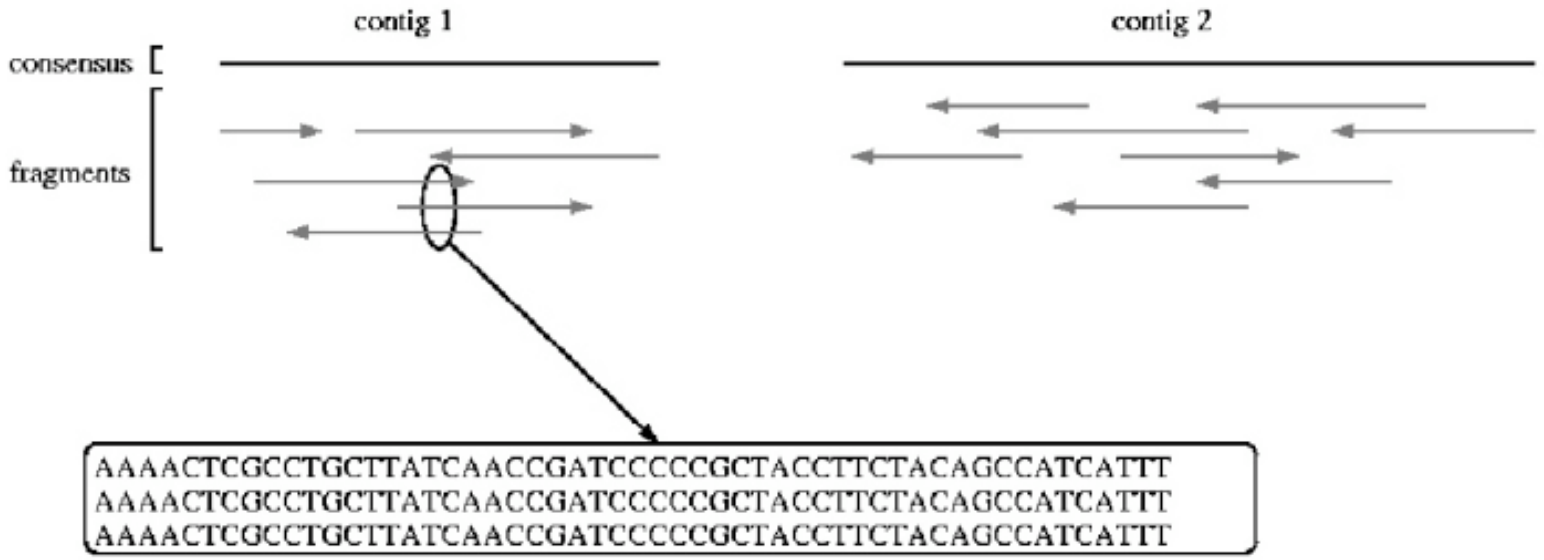
UMD assembly primer (cbcb.umd.edu)

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.



UMD assembly primer (cbcb.umd.edu)

Short read lengths are hard.

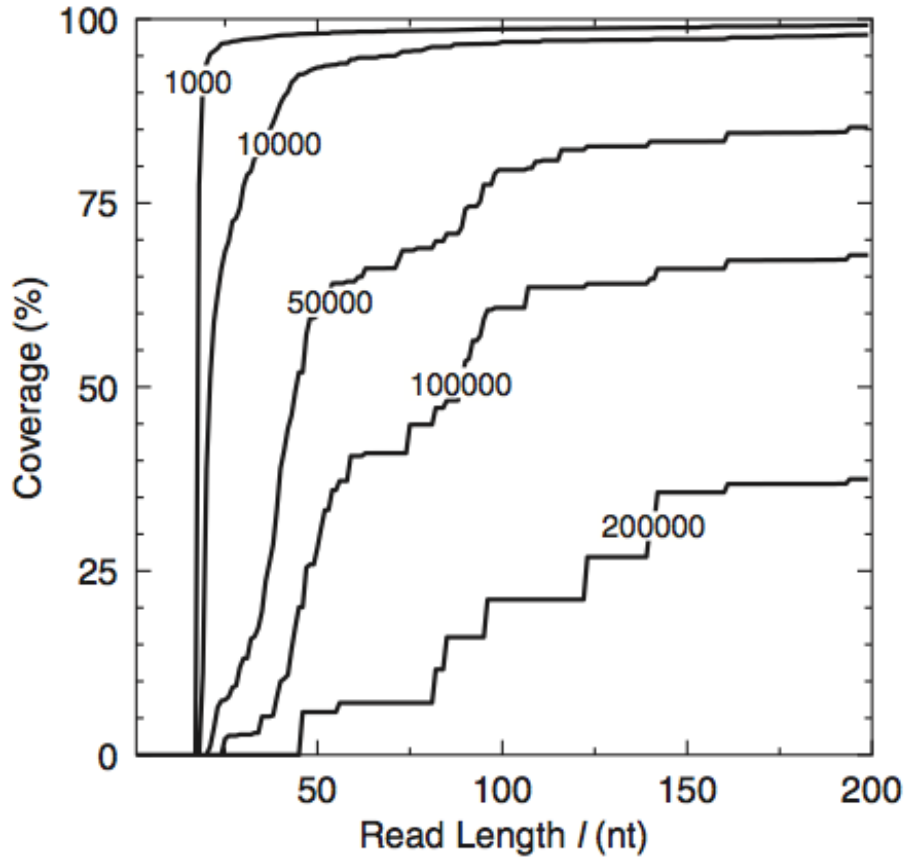


Figure 3. Percentage of the *E.coli* genome covered by contigs greater than a threshold length as a function of read length.

Whiteford et al., Nuc. Acid Res, 2005

Four main challenges for *de novo* sequencing.

- Repeats.
- Low coverage.
- Errors

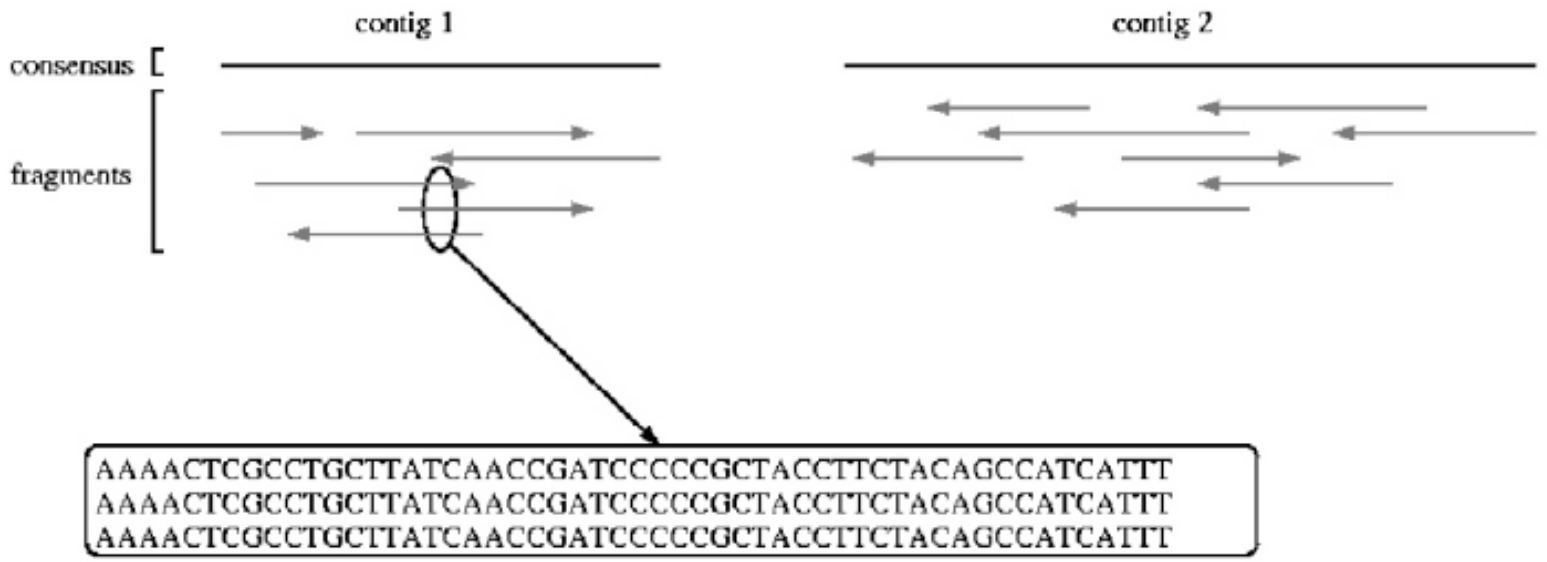
These introduce breaks in the construction of contigs.

- *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.



UMD assembly primer (cbcb.umd.edu)

Coverage

Easy calculation:

(# reads x avg read length) / genome size

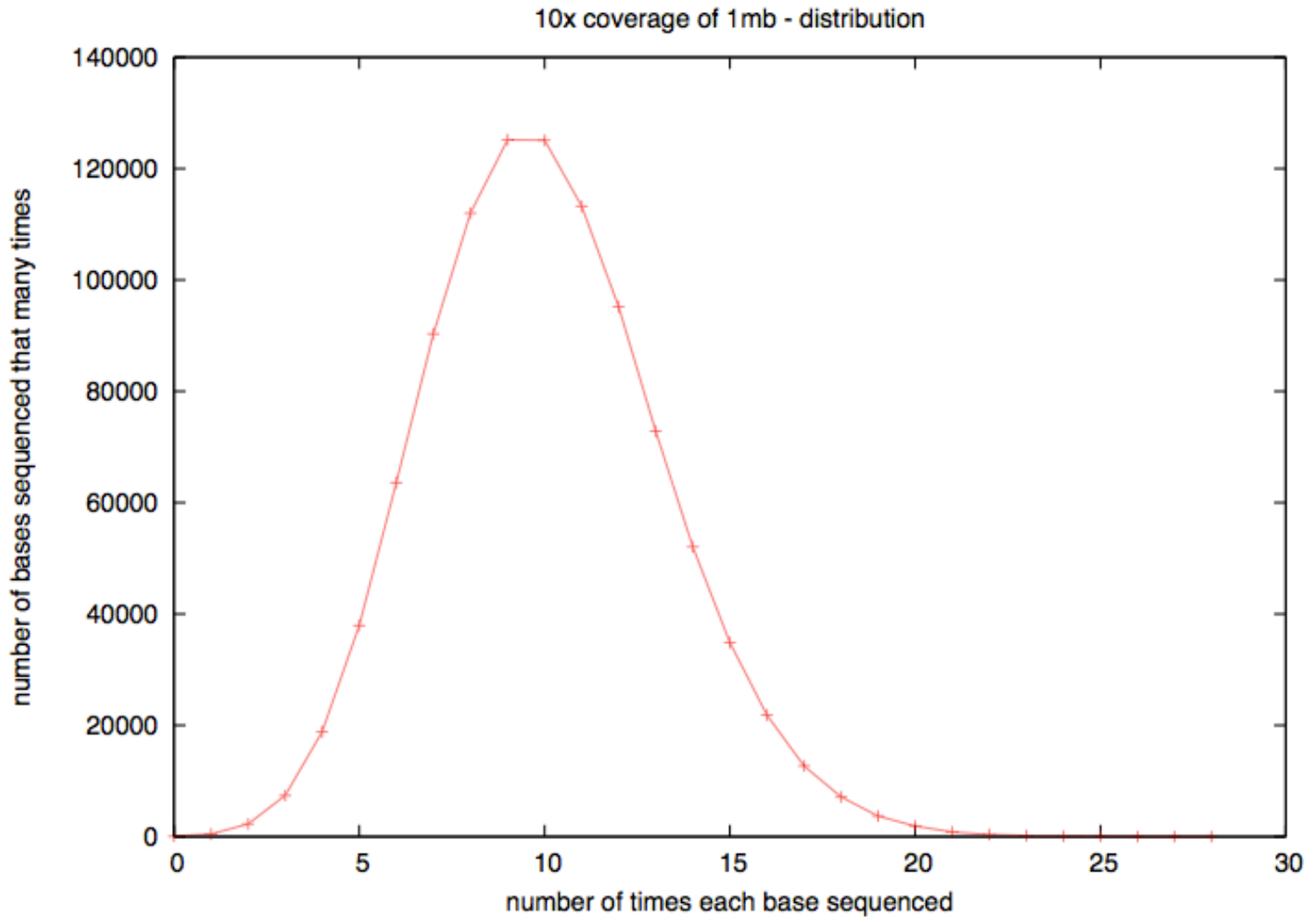
So, for haploid human genome:

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

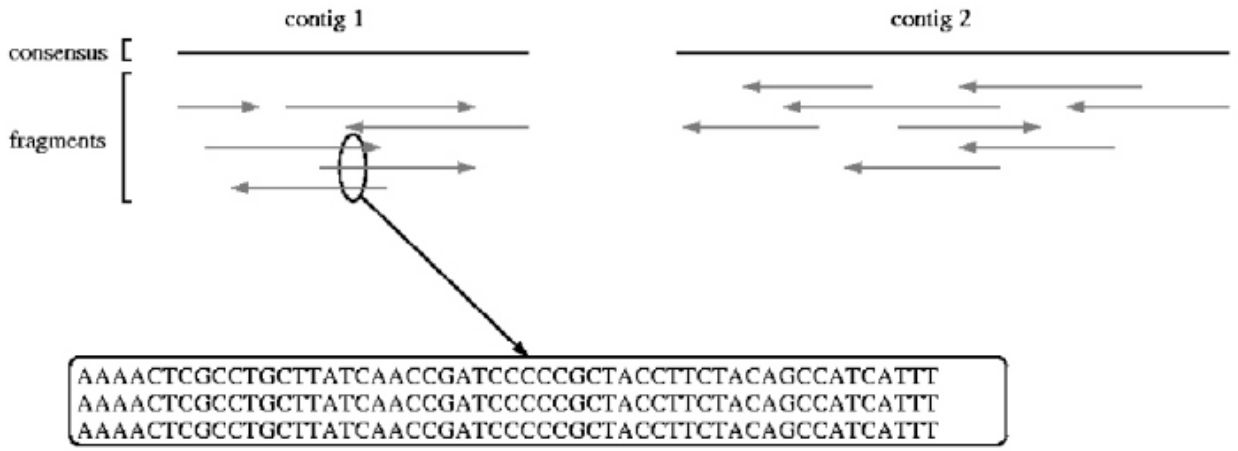
- Overlap/layout/consensus
- De Bruijn k-mer graphs

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



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
```

K-mers – what 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

K-mers – what k to use?

Table 1B. Fraction of *K*-mers having a unique placement on the genome

K	*E. coli* (%)	*S. cerevisiae* (%)	*A. thaliana* (%)	*H. sapiens* (%)
200	98.5	95.9	97.4	97.6
160	98.3	95.6	97.1	97.2
120	98.2	95.2	96.6	96.6
80	98.0	94.7	95.4	95.2
60	97.8	94.4	94.4	93.1
50	97.7	94.2	93.4	91.2
40	97.6	93.9	92.2	88.3
30	97.4	93.5	90.4	83.4
20	97.0	92.9	86.5	71.8
10	0.0	0.0	0.0	0.0

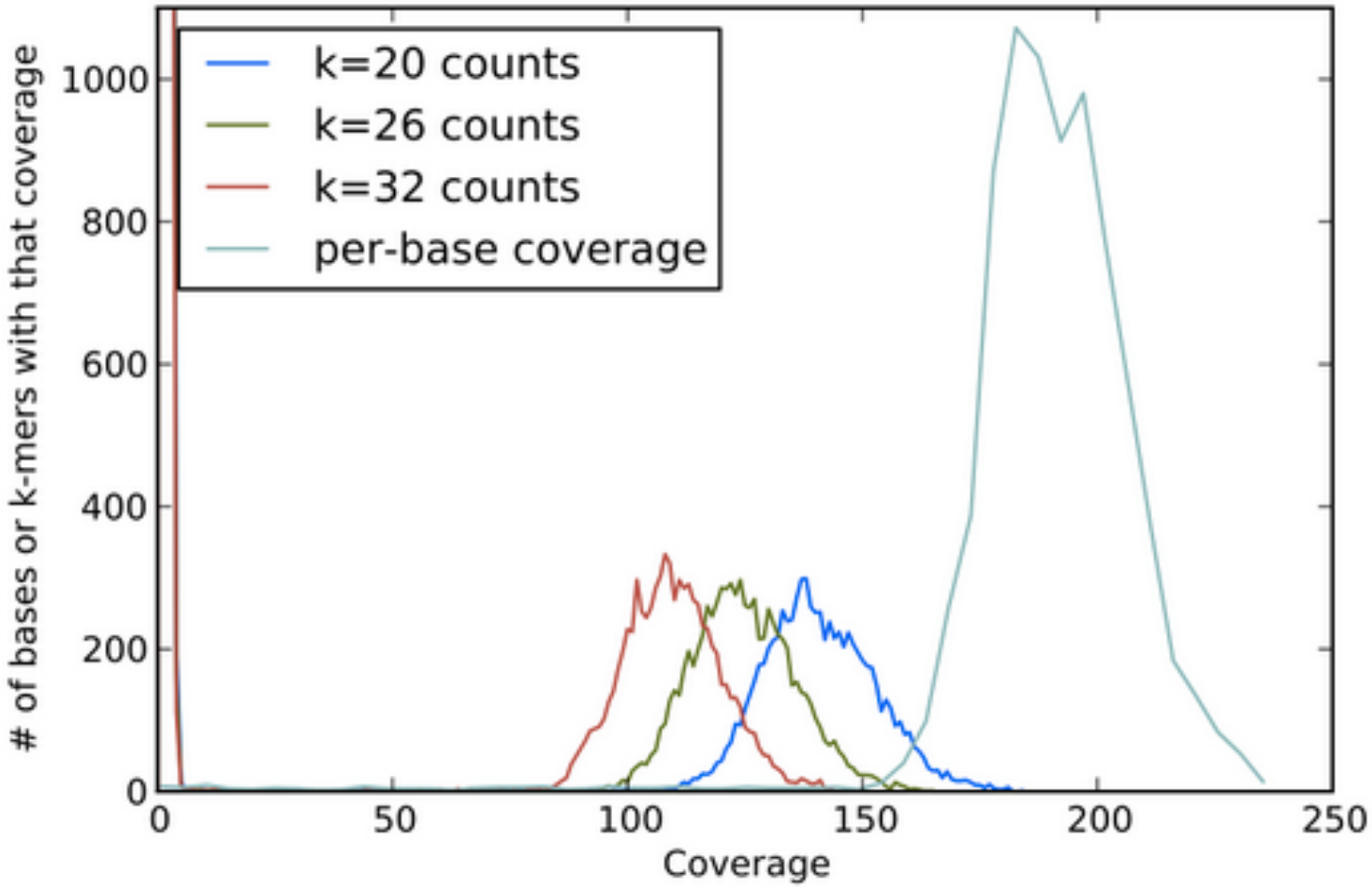
Butler et al., Genome Res, 2009

Big genomes are problematic

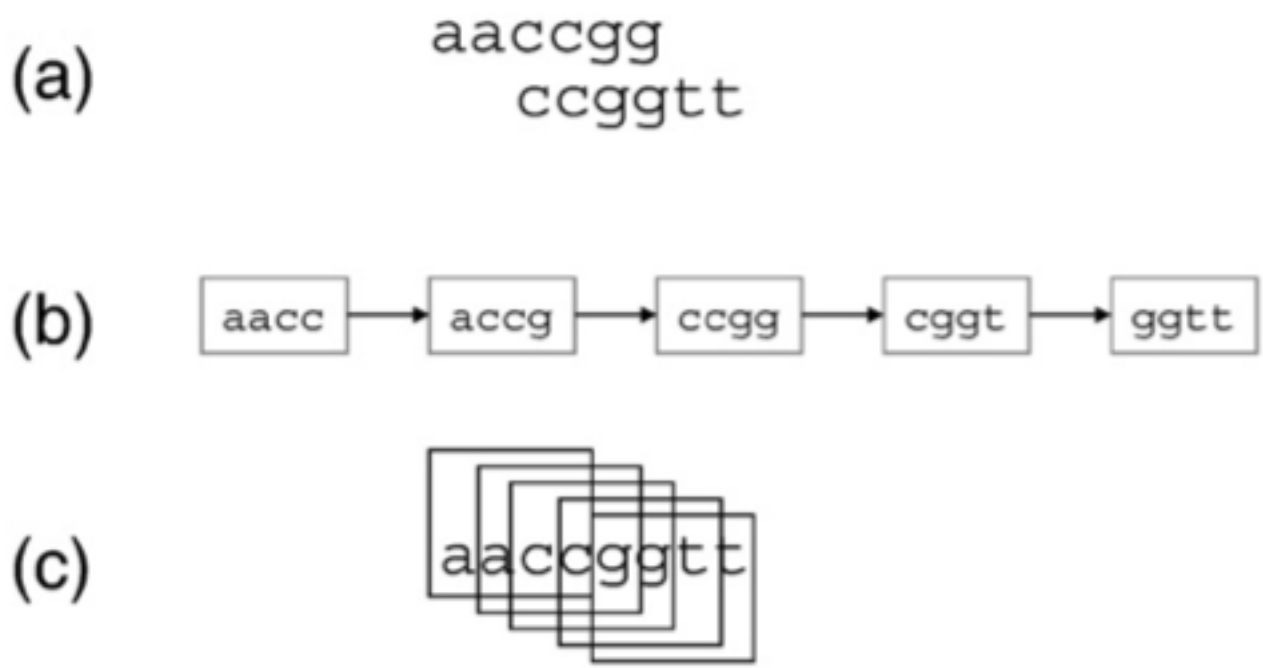
Species	Ploidy	Genome size (kb)	Reference N50 (kb)	Component N50 (kb)	Edge N50 (kb)	Ambiguities per megabase	Coverage (%)	Coverage by perfect edges ≥10 kb (%)
C. jejuni	1	1800	1800	1800	1800	0.0	100.0	100.0
E. coli	1	4600	4600	4600	4600	0.0	100.0	100.0
B. thailandensis	1	6700	3800	1800	890	2.7	99.8	99.5
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

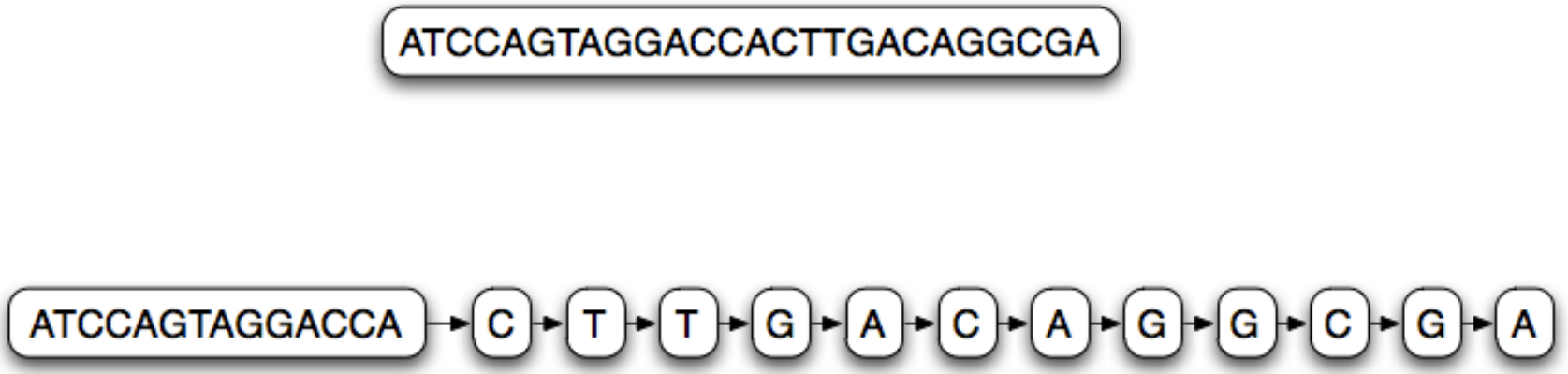


K-mer graphs - overlaps



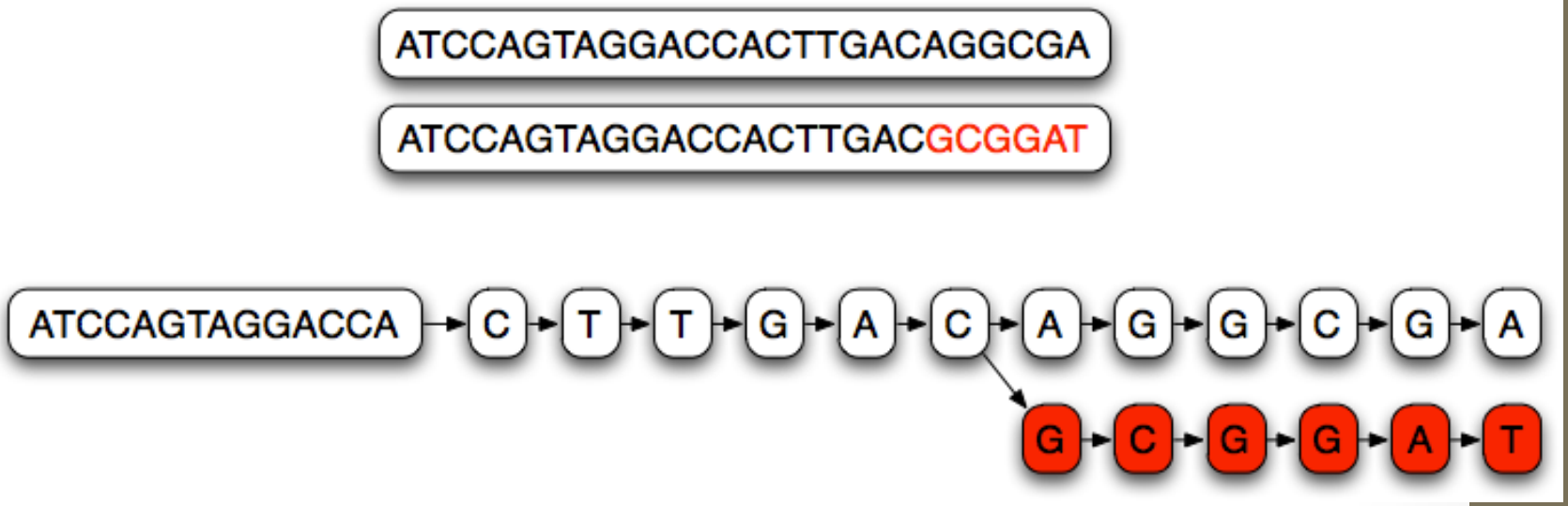
J.R. Miller et al. / Genomics (2010)

K-mer graph (k=14)



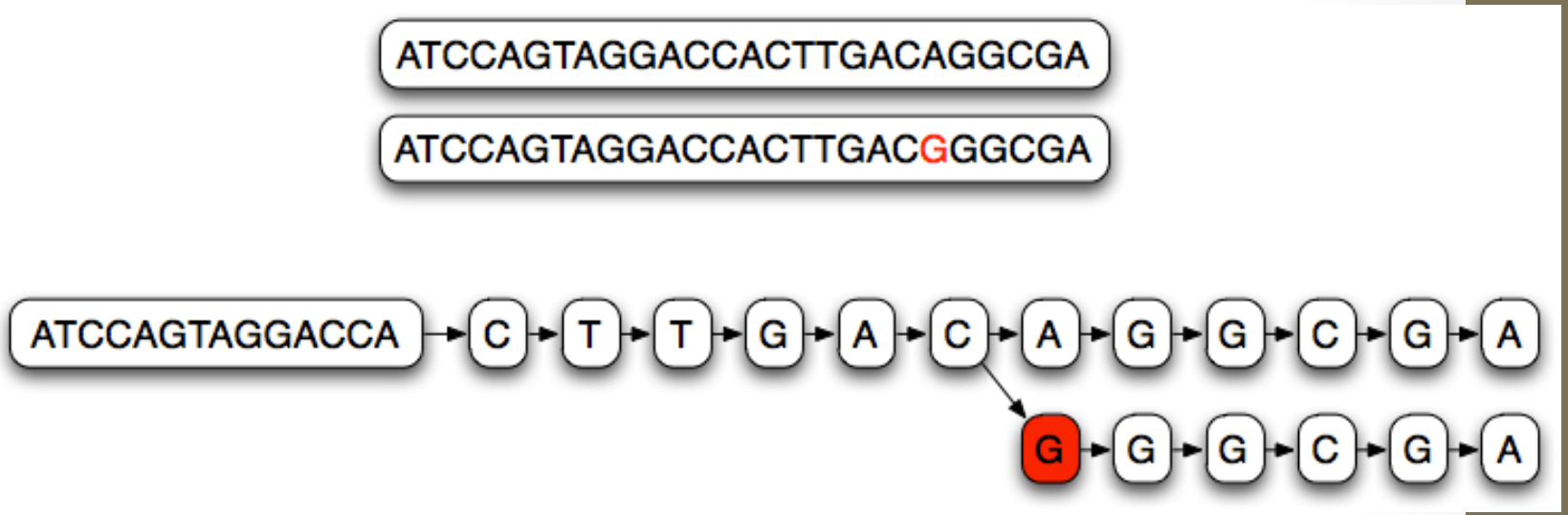
Each node represents a 14-mer;
Links between each node are 13-mer overlaps

K-mer graph (k=14)



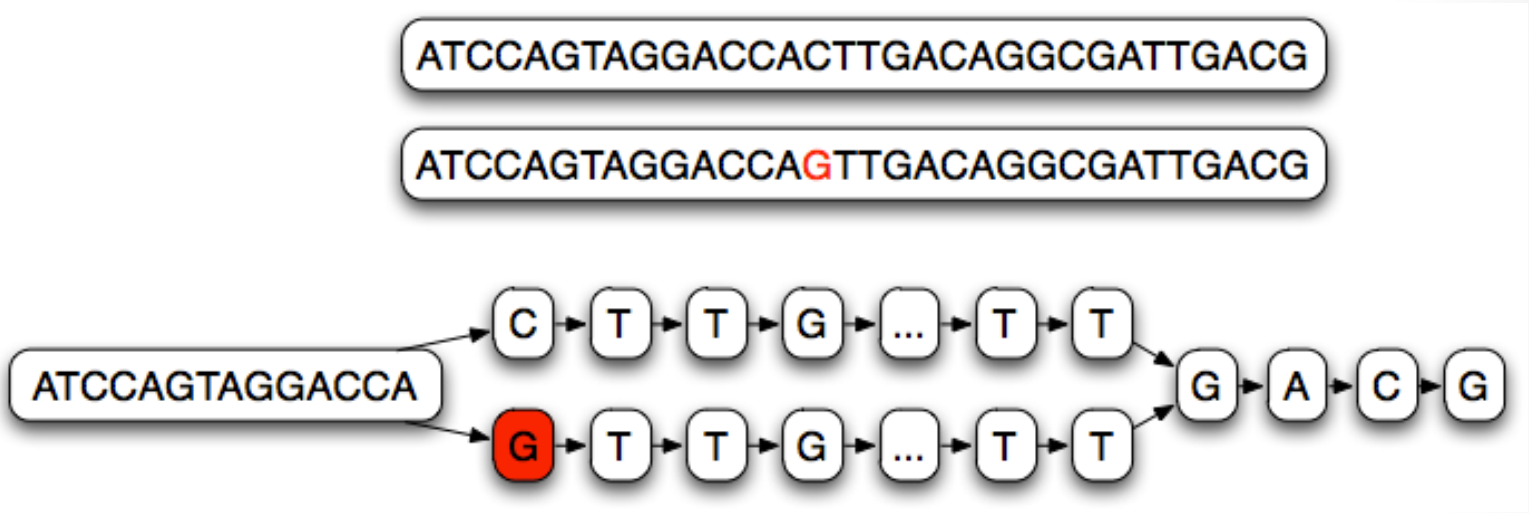
Branches in the graph represent partially overlapping sequences.

K-mer graph (k=14)



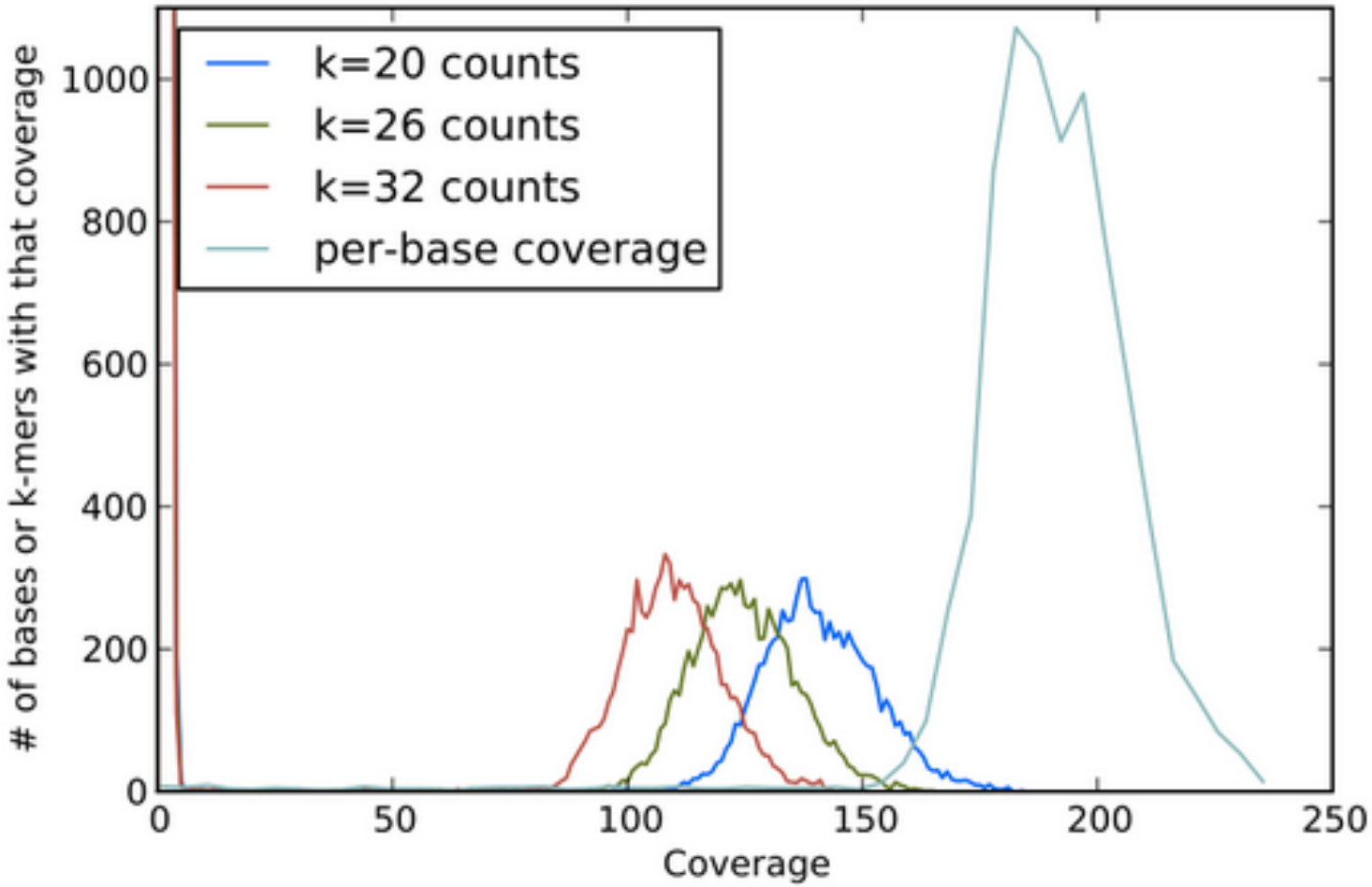
Single nucleotide variations cause long branches

K-mer graph (k=14)

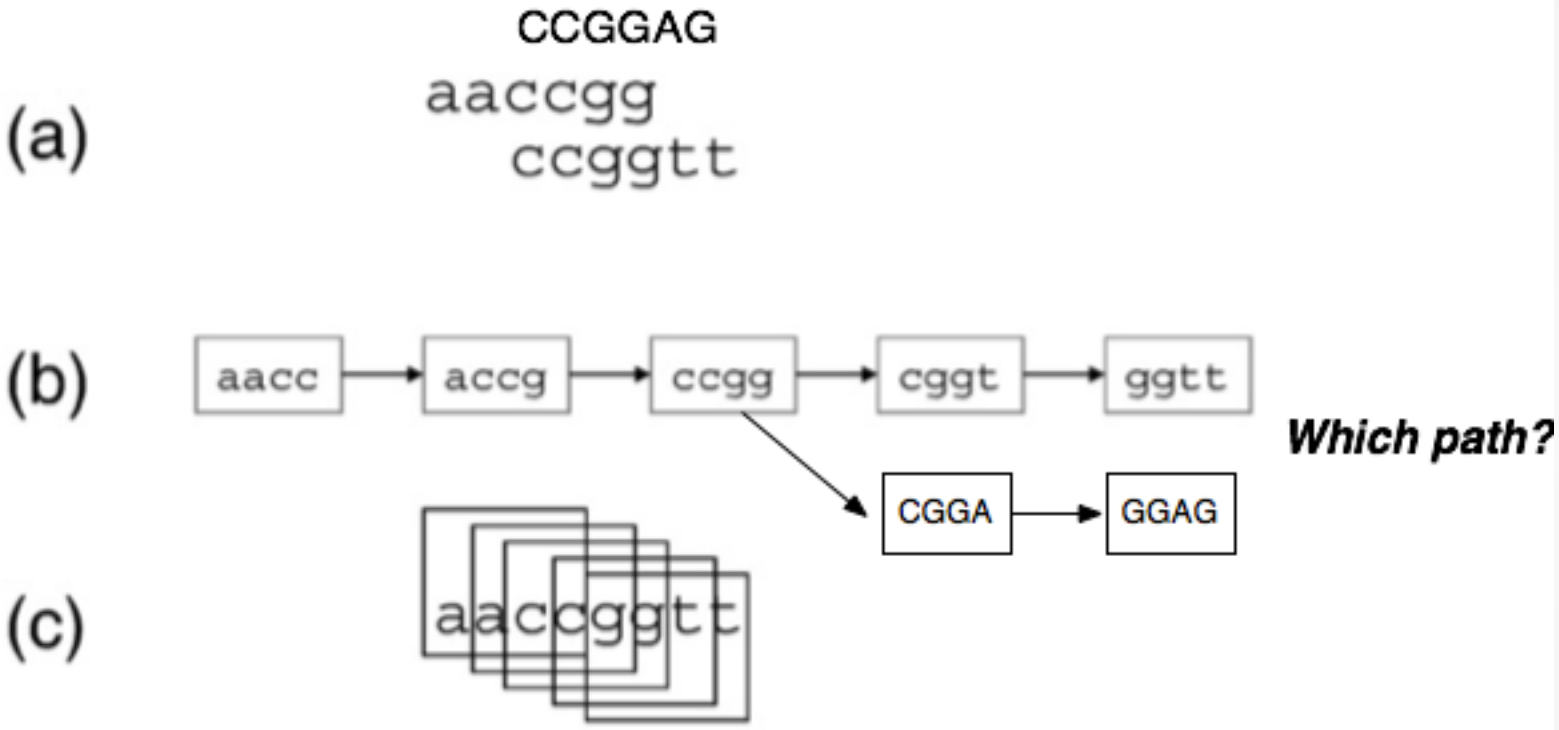


Single nucleotide variations cause long branches;
They don't rejoin quickly.

Choice of k affects apparent coverage

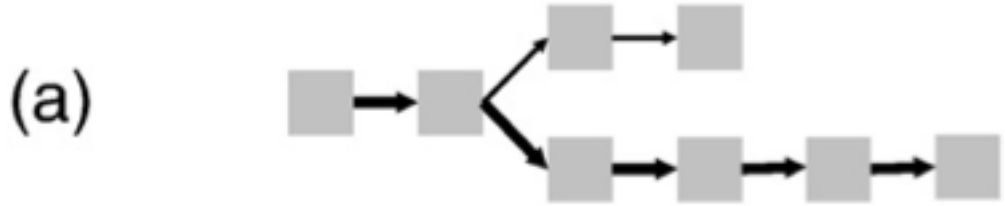


K-mer graphs - branching



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

K-mer graph complexity - spur

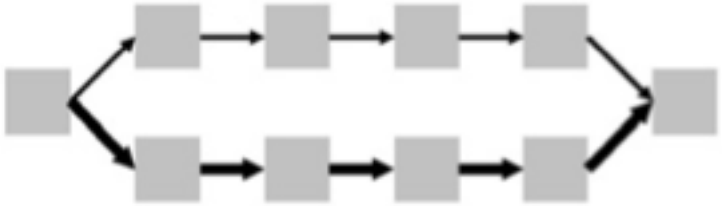


Can be caused by error at the end of some overlapping reads, or low coverage

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

K-mer graph complexity - bubble

(b)

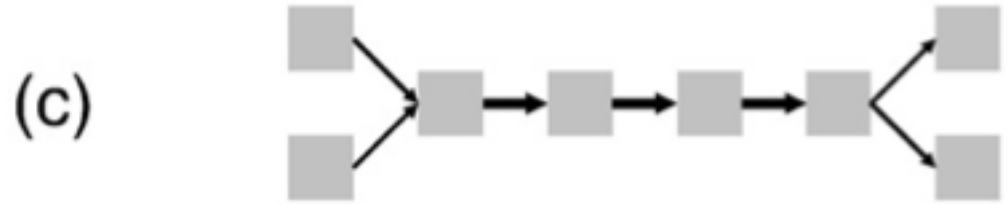


Multiple par

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

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

K-mer graph complexity – "frayed rope"



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.

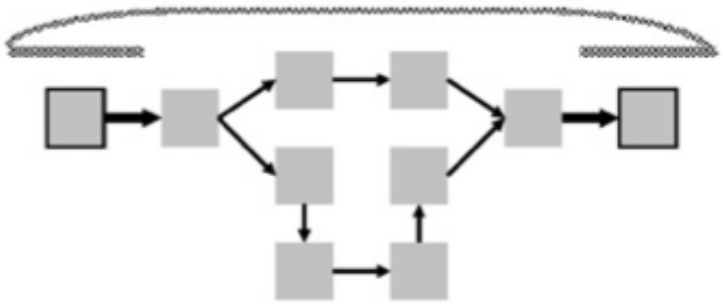
J.R. Miller et al. / Genomics (2010)

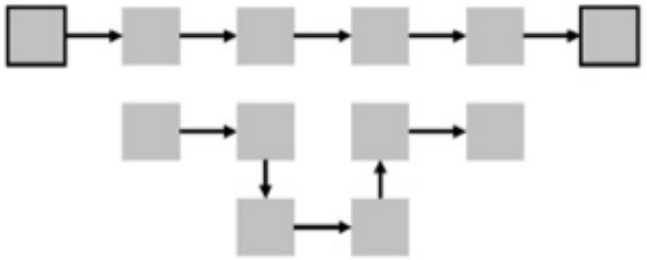
Mate threading

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

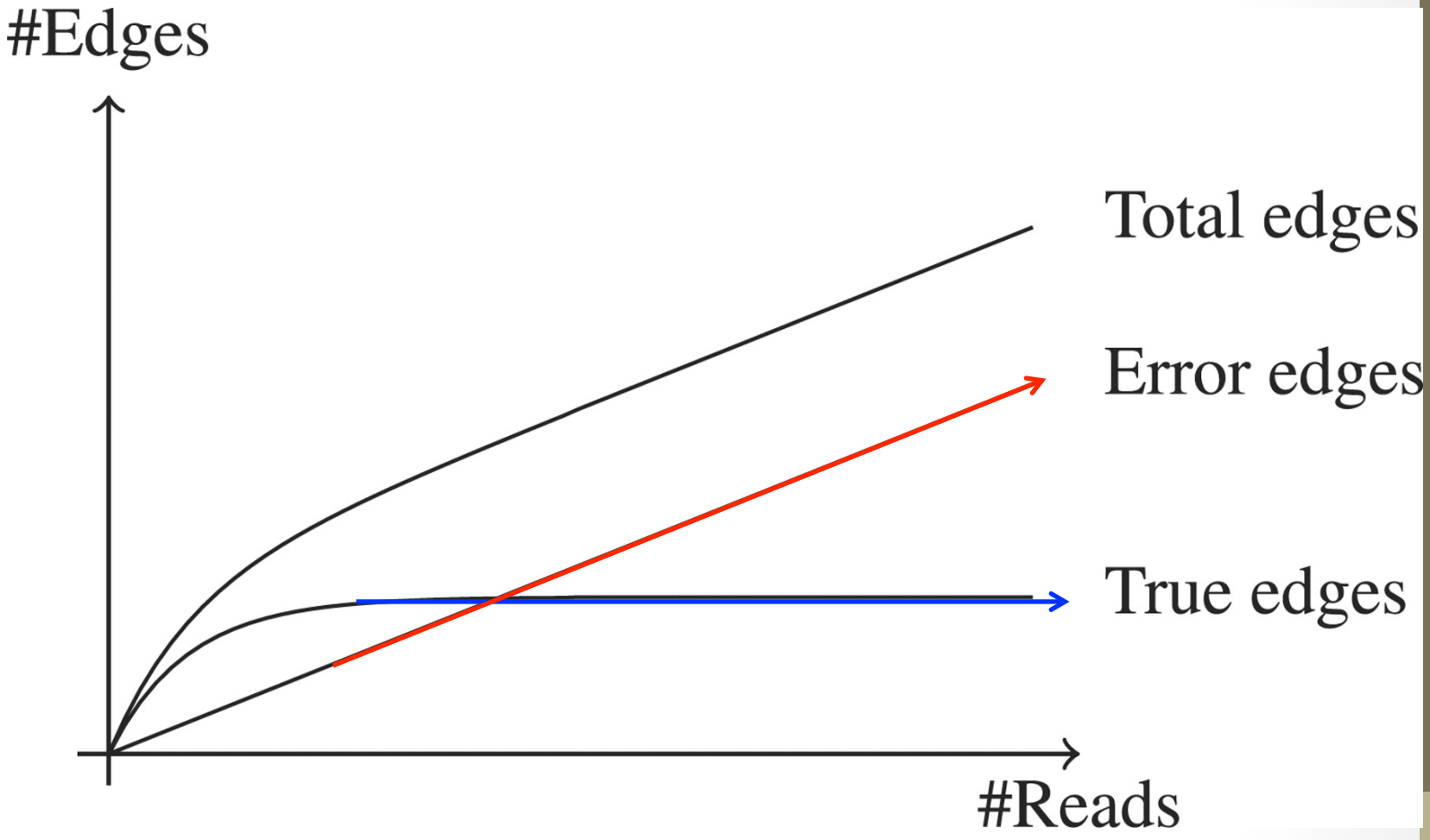
GCGTCAGGTAG**C**AGACCACCGCCATGGCGACGATG

GCGTCAGGTAGGAGACCACCG**T**CATGGCGACGATG

GCGT**T**AGGTAGGAGACCACCGCCATGGCGACGATG

GCGTCAGGTAGGAGACC**G**CCGCCATGGCGACGATG

De Bruijn graphs scale poorly with erroneous data

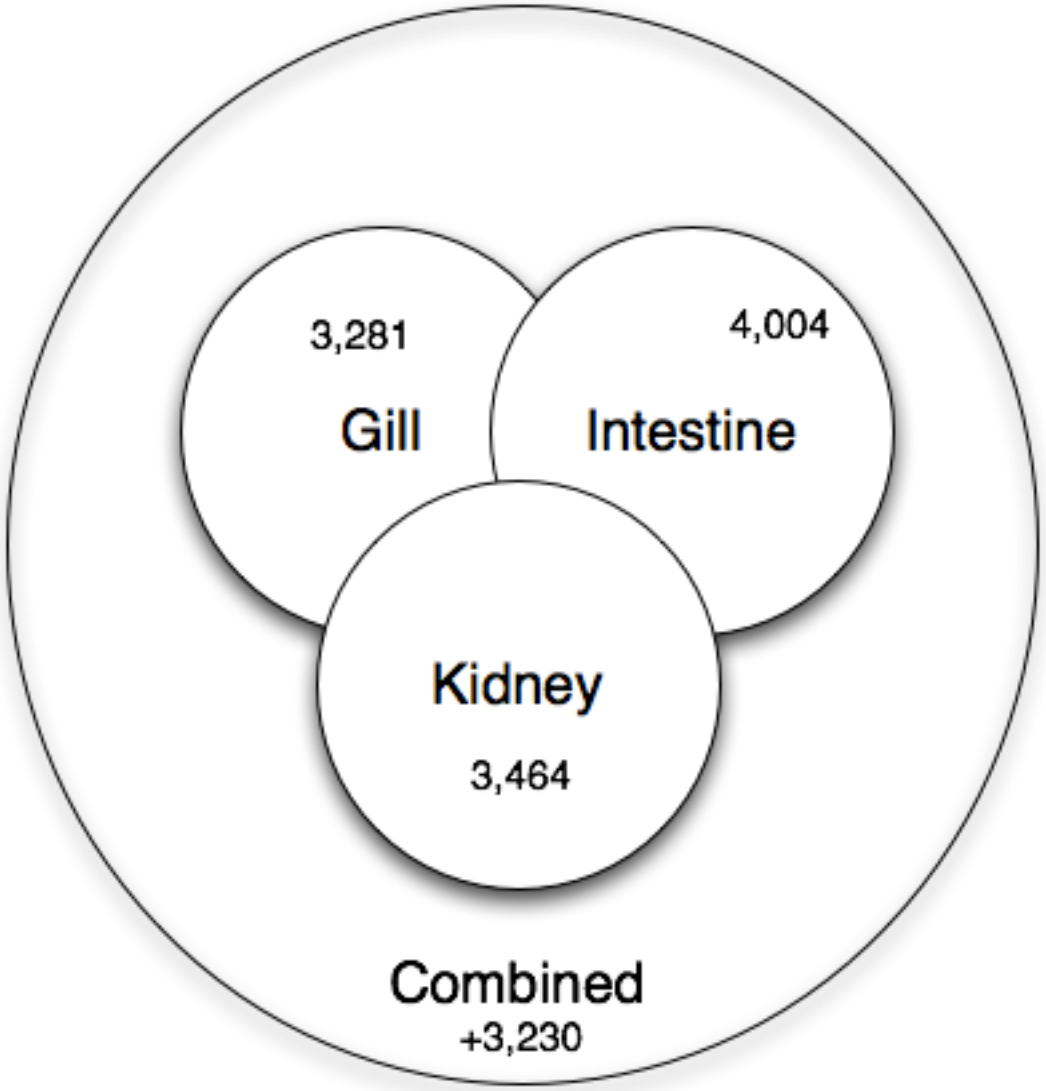


Conway T C , Bromage A J Bioinformatics 2011;27:479-486

Bioinformatics

Co-assembly is important for sensitivity

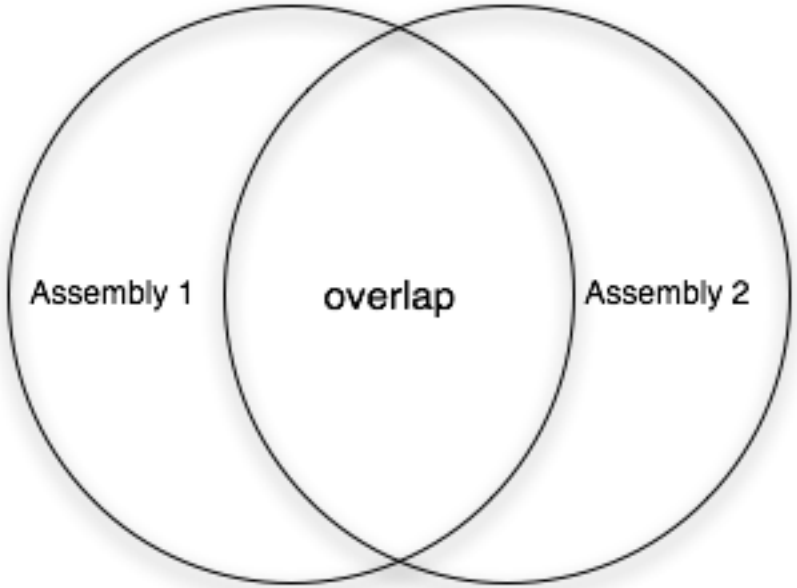
Shared low-level transcripts may not reach the threshold for assembly.



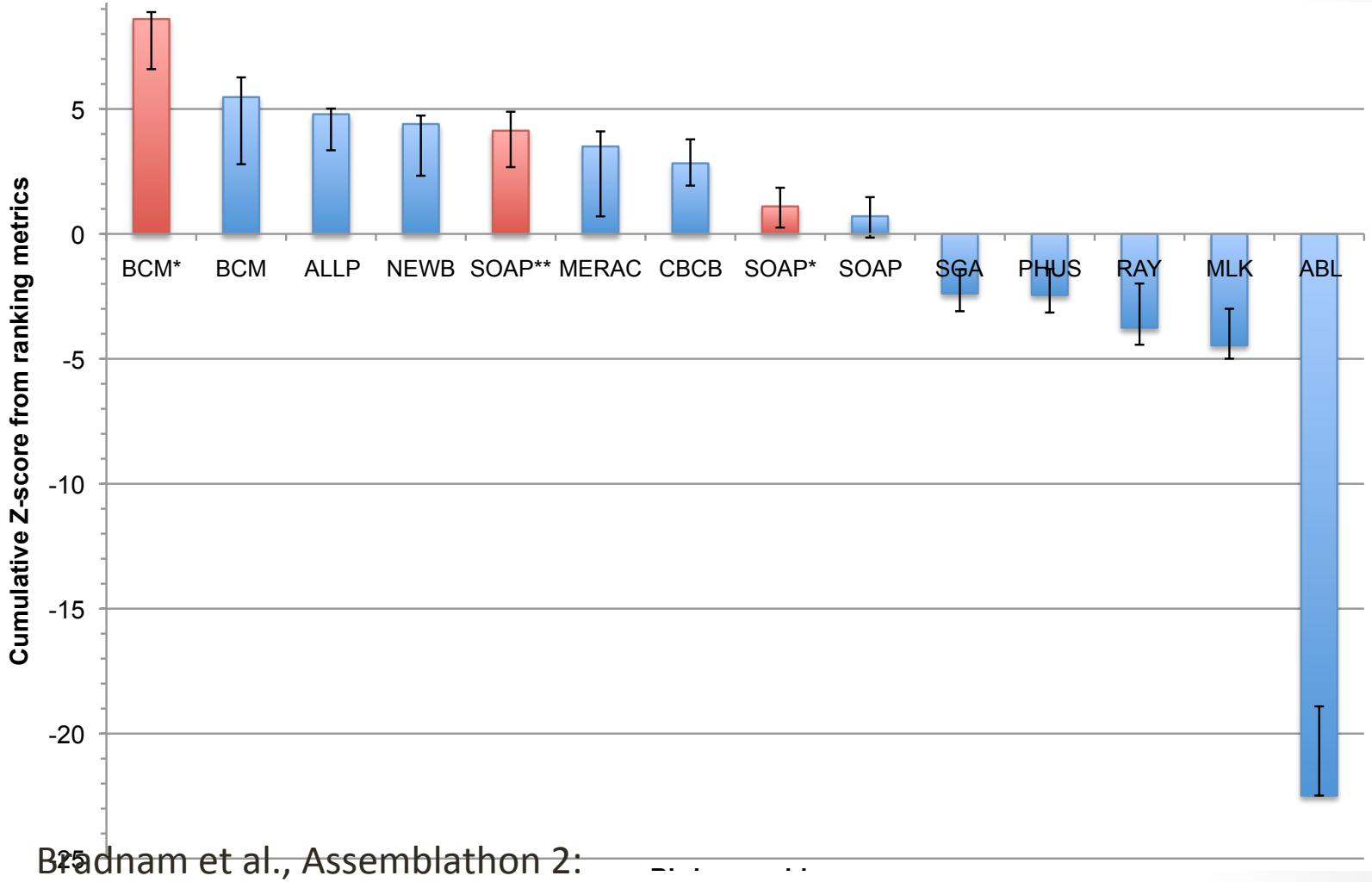
Is your assembly good?

- 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?

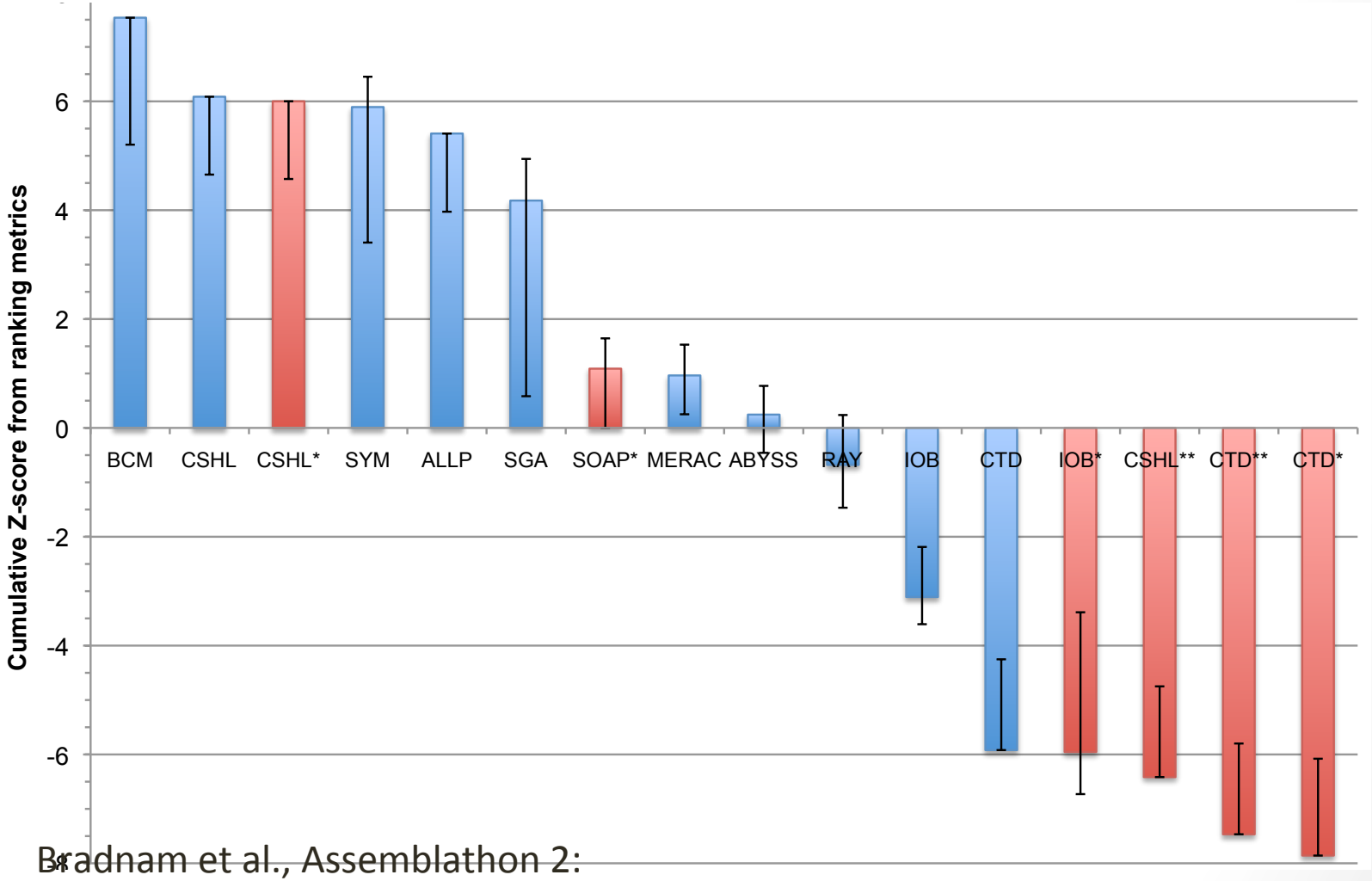


What's the best assembler?



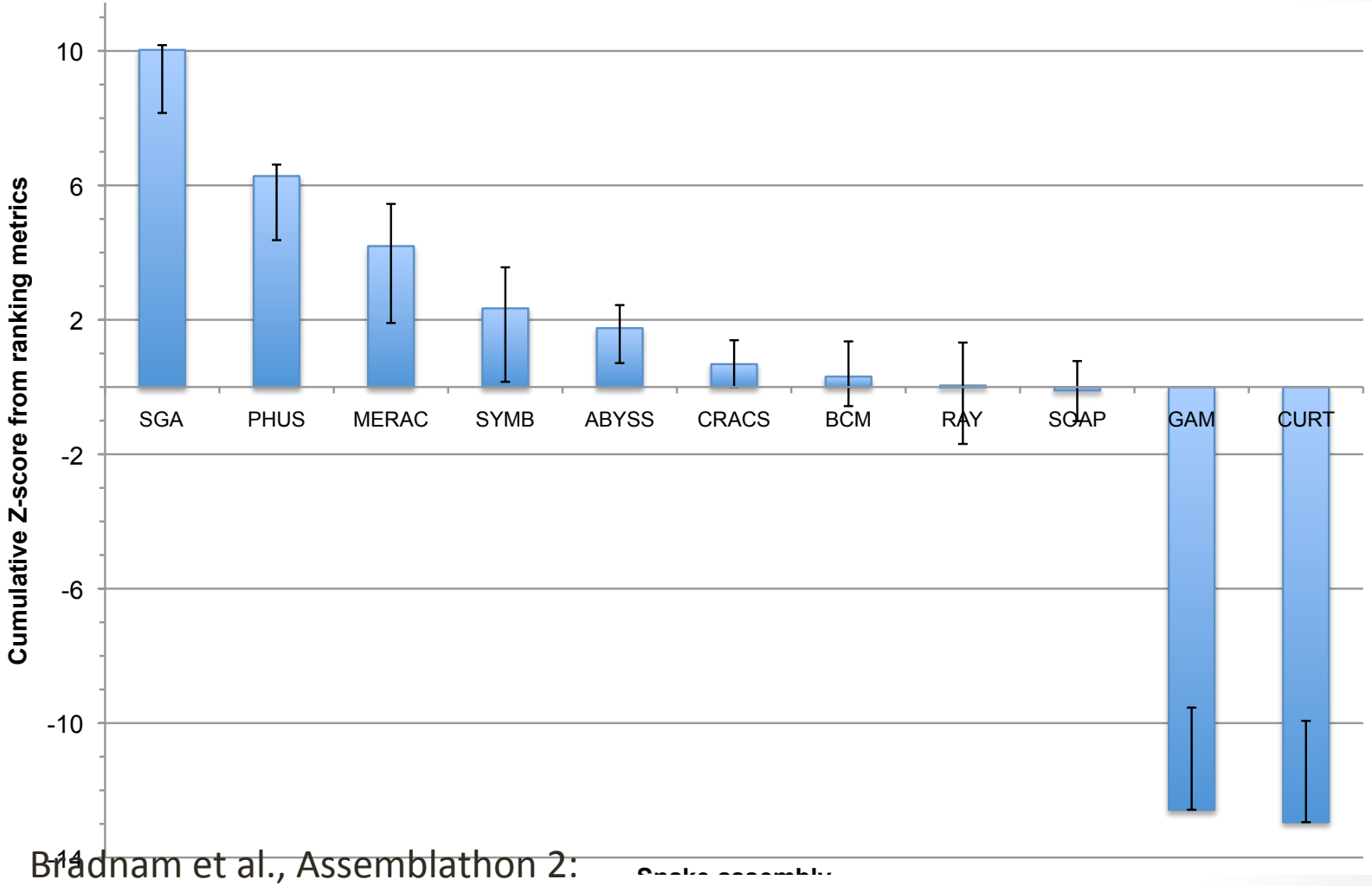
Bradnam et al., Assemblathon 2:
http://arxiv.org/pdf/1301.5406v1.pdf

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

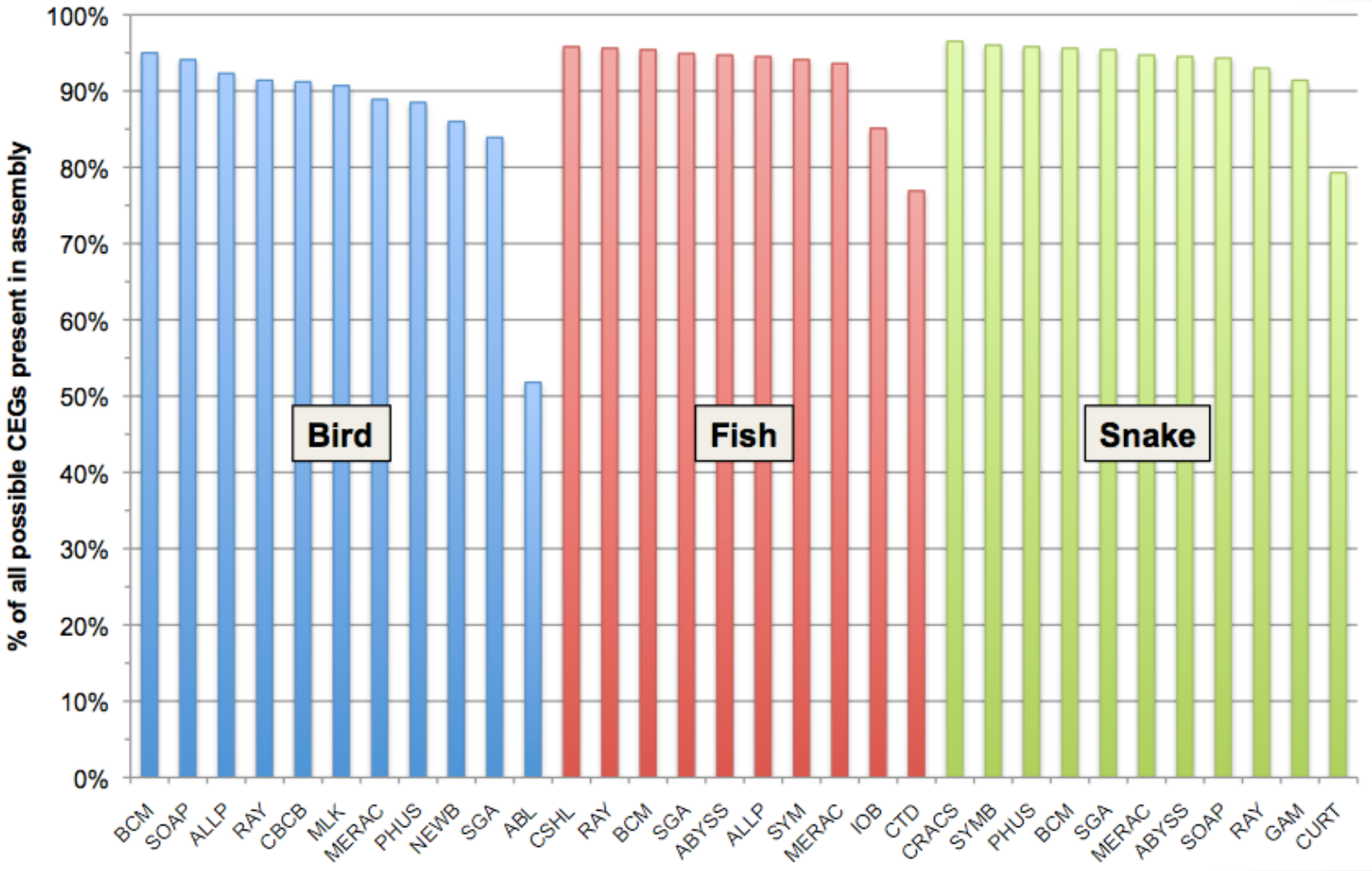
Note: the teams mostly used *multiple* software packages

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

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.

- 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.

- Go forth! Map! Assemble!