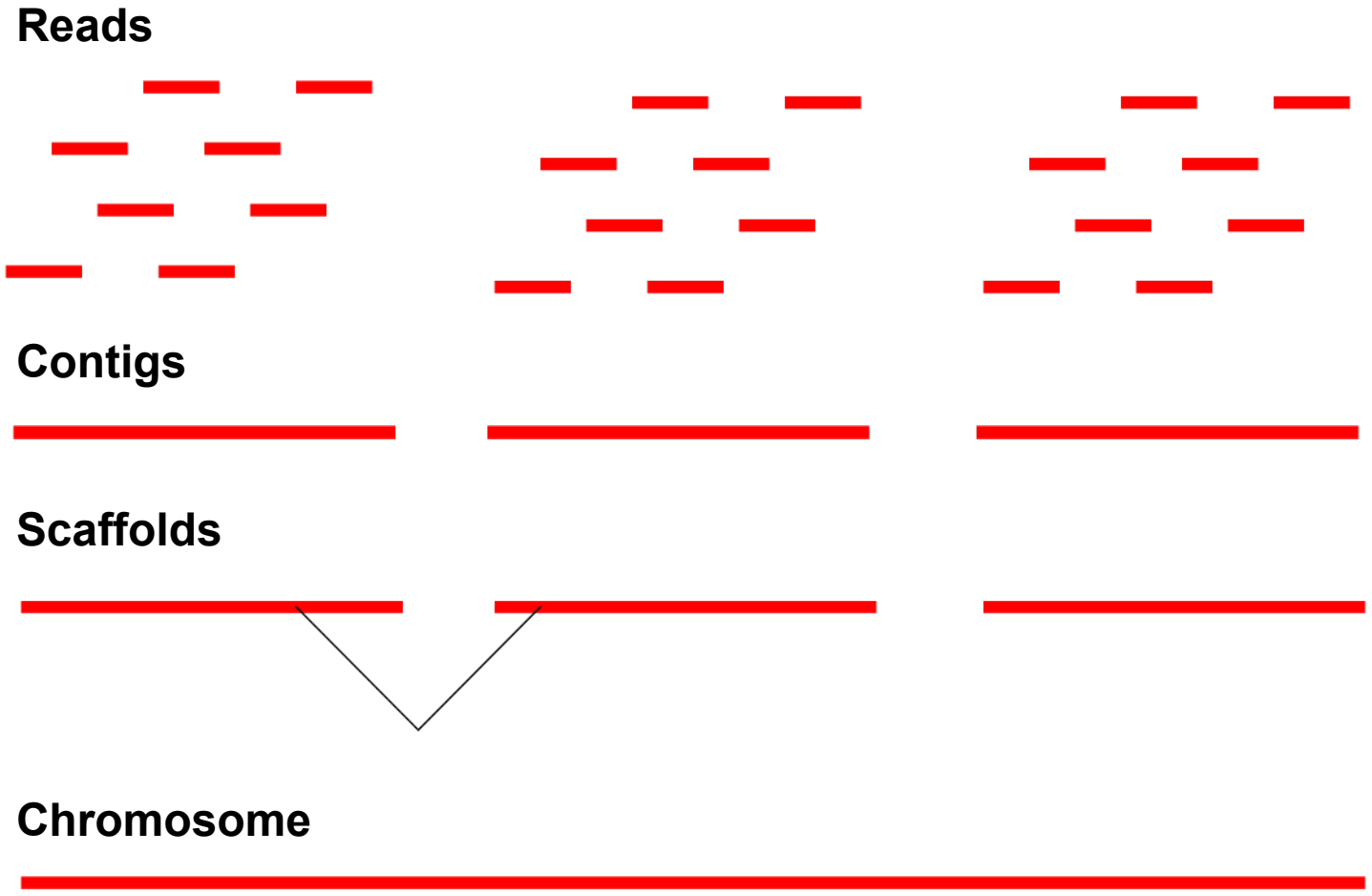
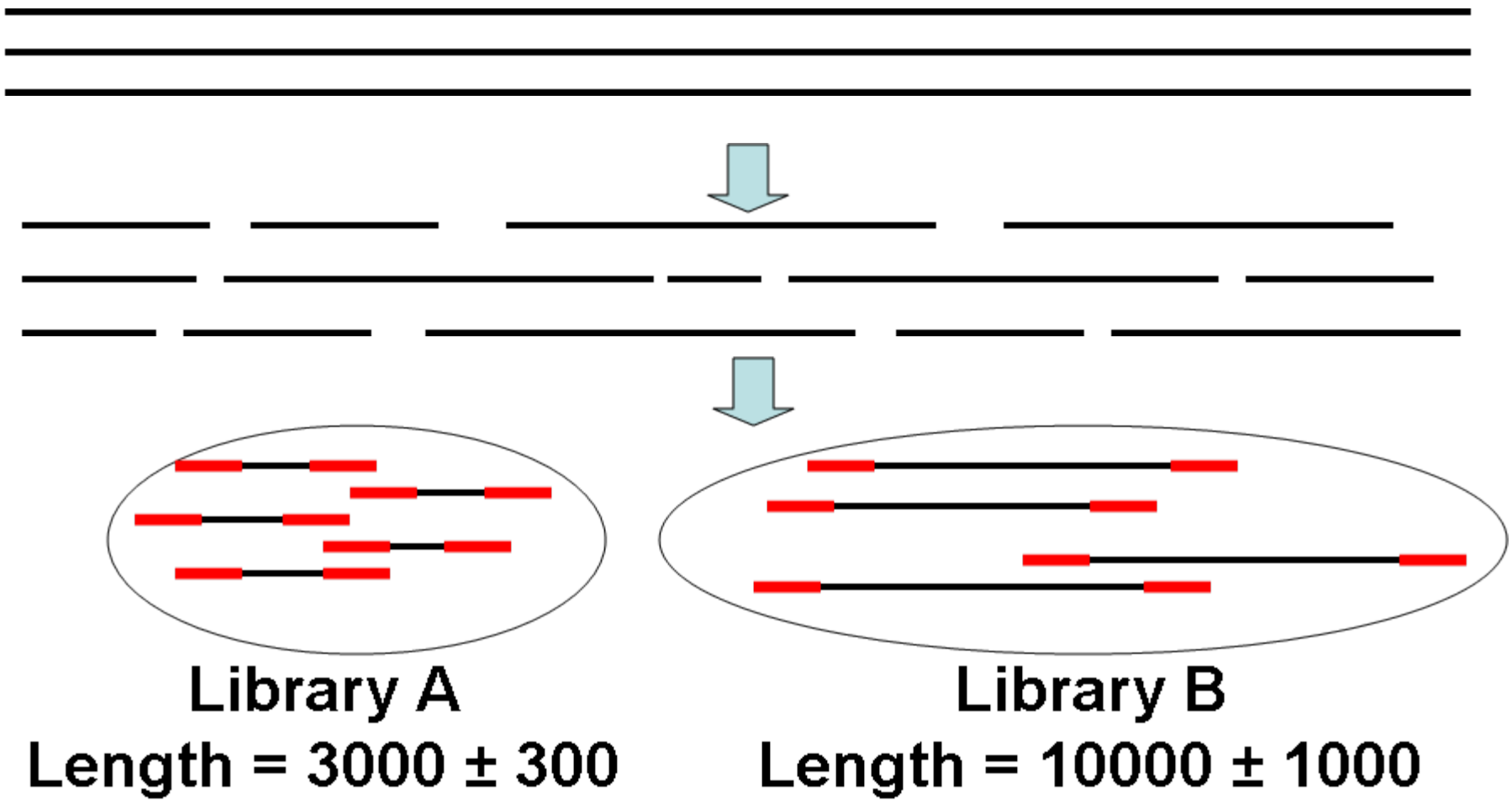
Assembly validation

Review: Genome assembly

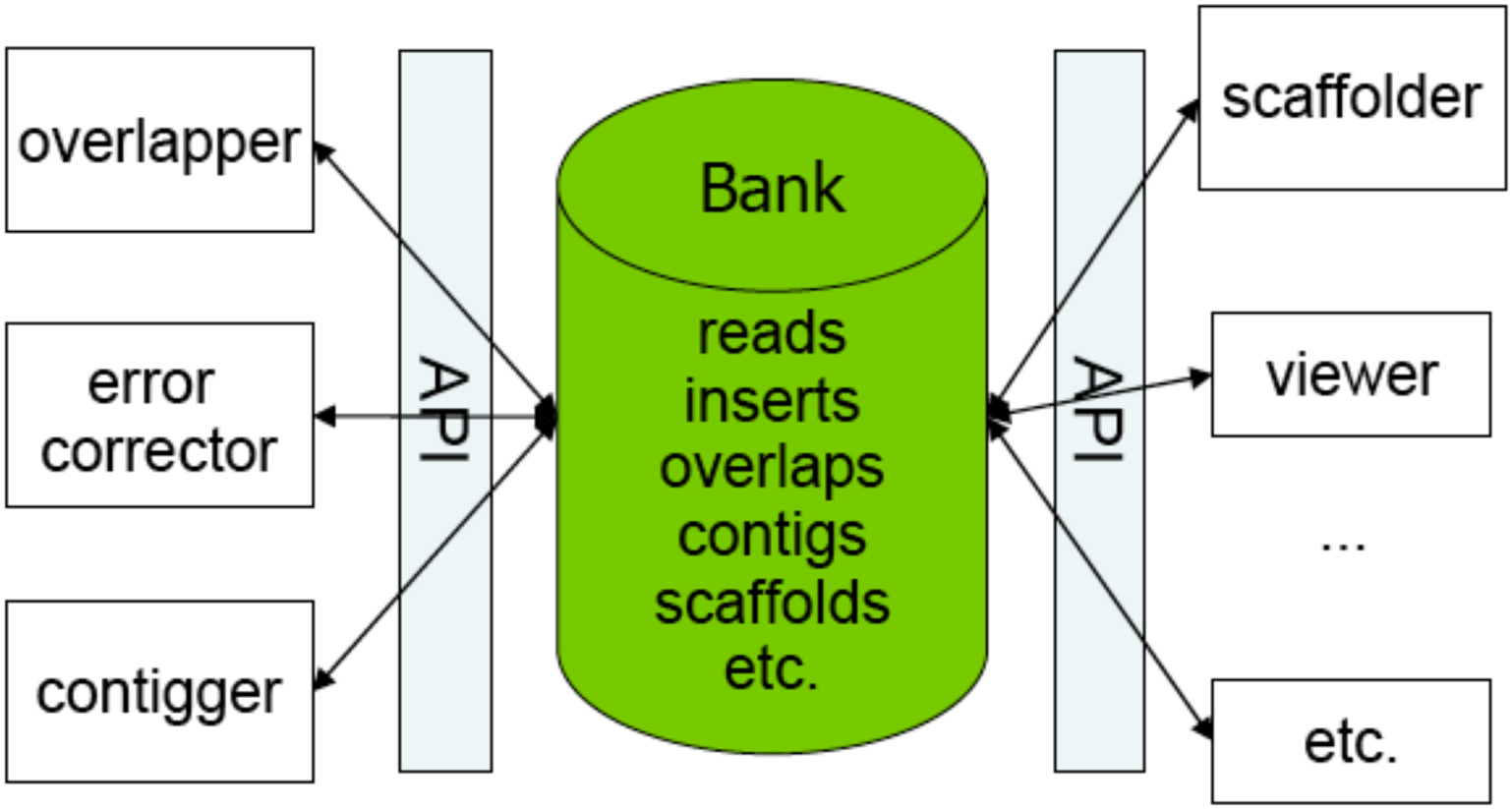


Review: Mate pair data

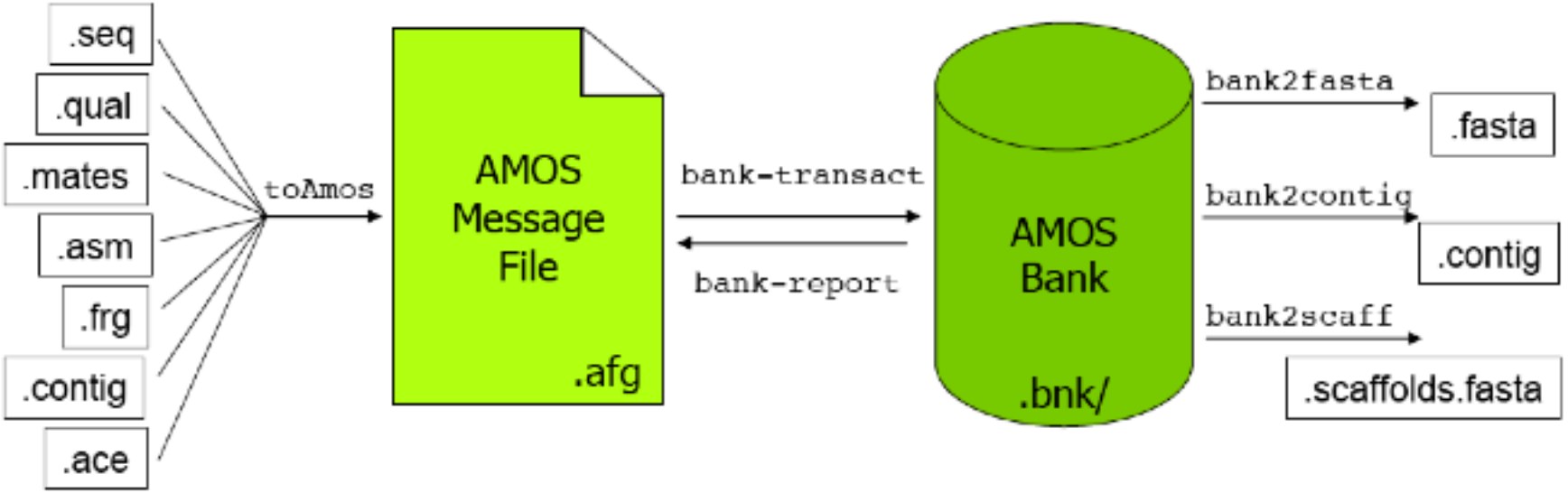


Overlap-Layout-Consensus

AMOS project: A Modular Open Source assembler



Importing data to an AMOS bank



CA Assembly w/ Surrogates to AMOS Message File (.asm, .frg)

```
$ toAmos –a prefix.asm –f prefix.frg –o prefix.afg –s
```

Is it a good assembly?

- # of contigs
- Average size of contigs
- N50
- Genome coverage
- Misassemblies?

Contig size

- Average contig length doesn't necessarily capture how big your contigs are

E.g., you have a 100kb genome.

Assembly 1 contig lengths: 45 kb, 4 X 2 kb

Assembly 2 contig lengths: 5 X 20 kb

Average contig length is 20 kb in both cases.

Alternative: N50 (or N80, etc.) What is the smallest contig you need to make up 50% of the **distribution** of the contig lengths

Which is the best assembly?

Table from Pop, *IEEE Computer*

Assembler	# of contigs	Avg contig length	Total size
Phrap	56	22.4 kbp	1.26 Mbp
TIGR Assembler	76	16.8 kbp	1.28 Mbp
Celera Assembler	220	6.3 kbp	1.39 Mbp
Celera, trimmed data	101	12.5 kbp	1.26 Mbp

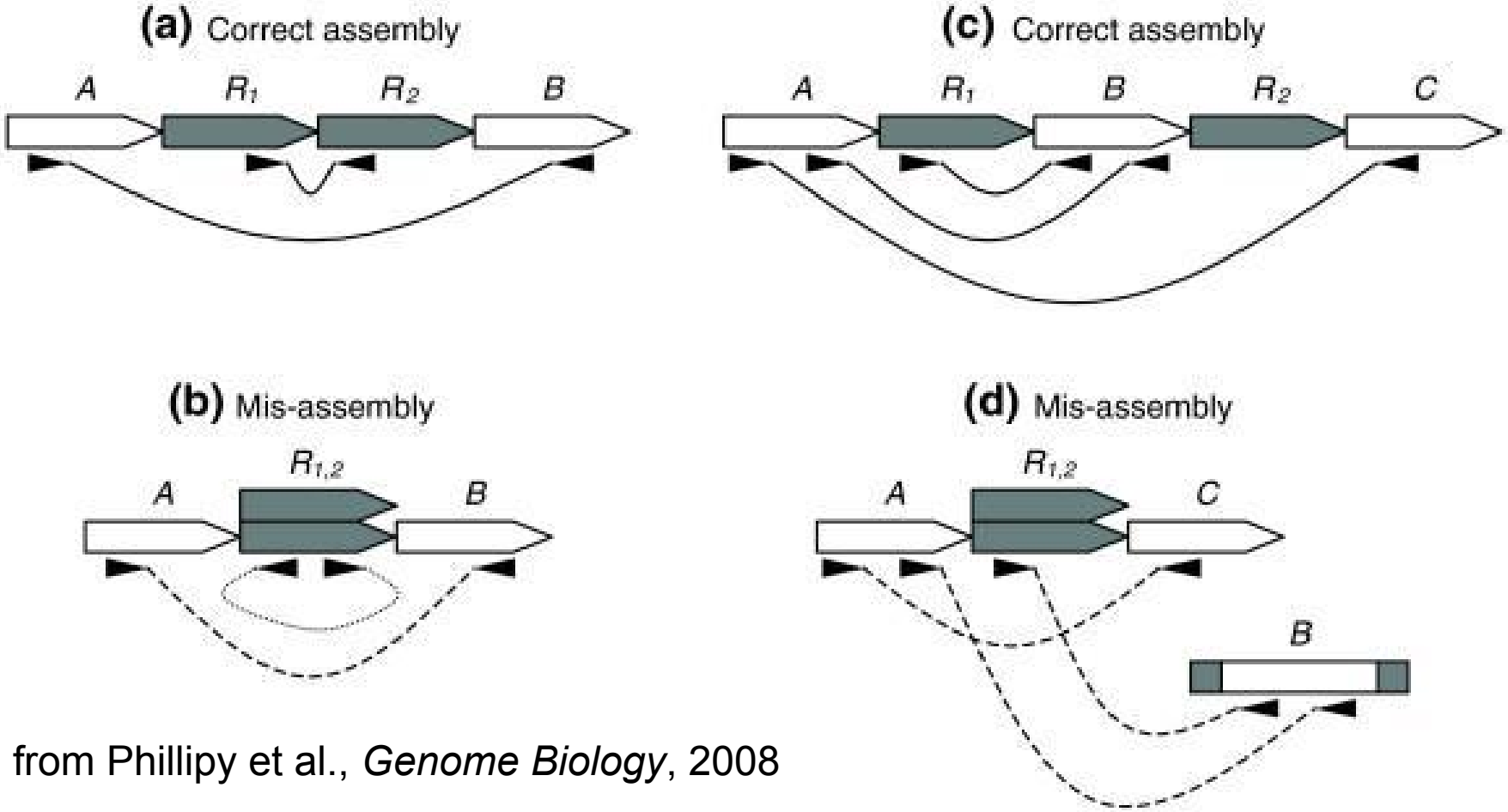
Which is the best assembly?

	# of contigs	Avg. contig length	Total size	% genome covered	# mis-assemblies
Phrap	56	22.4 kbp	1.26 Mbp	**36.0**	**14**
TIGR	76	16.8 kbp	1.28 Mbp	93.1	2
Celera	220	6.3 kbp	1.39 Mbp	99.1	1
Celera, trimmed data	101	12.5 kbp	1.26 Mbp	98.4	0

Table from Pop, *IEEE Computer*

What causes misassemblies?

REPEATS



from Phillipy et al., *Genome Biology*, 2008

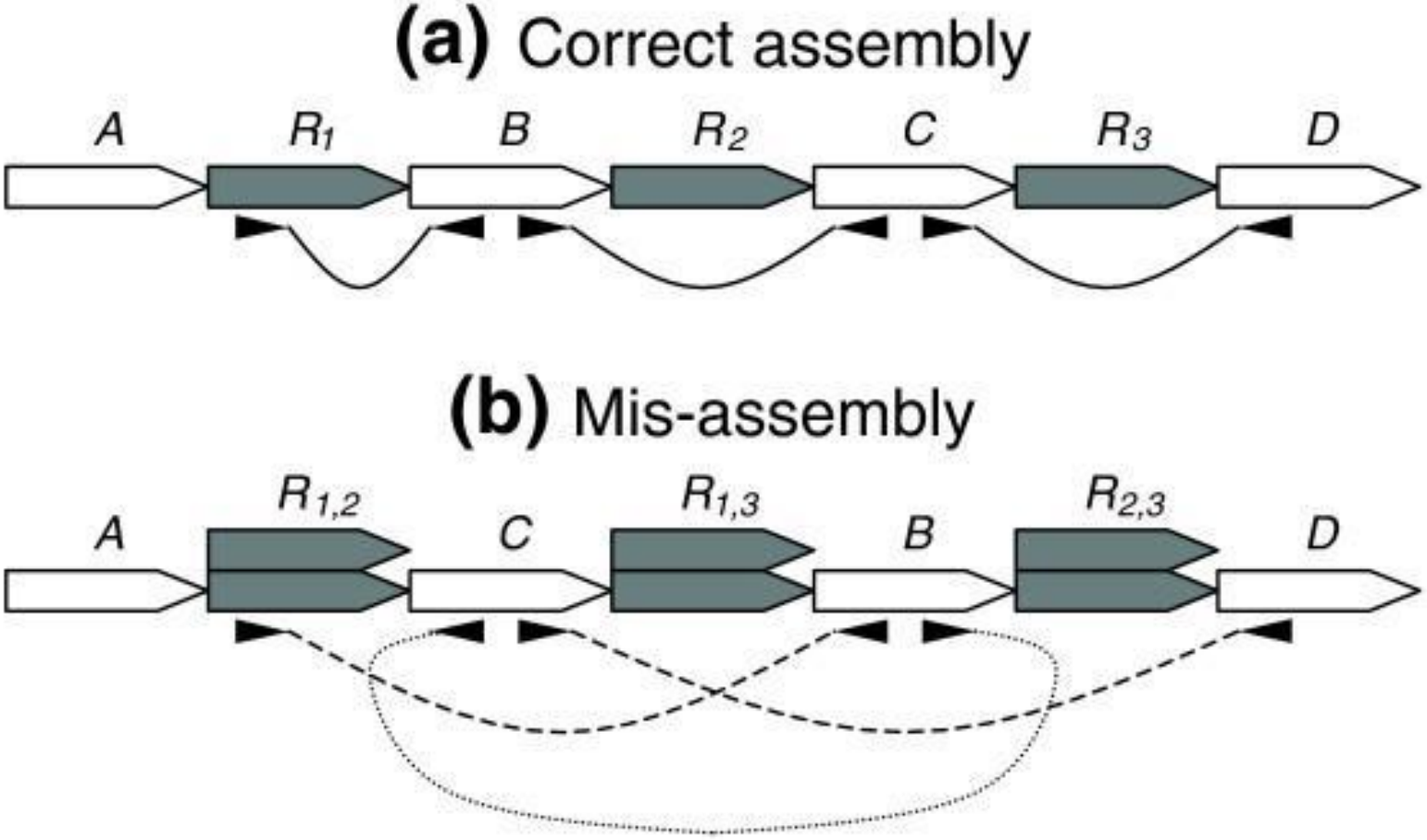
Other Factors

- Heterozygosity
- Contaminated samples
- Sequencing errors → e.g. homopolymer runs, substitutions

Types of misassembly

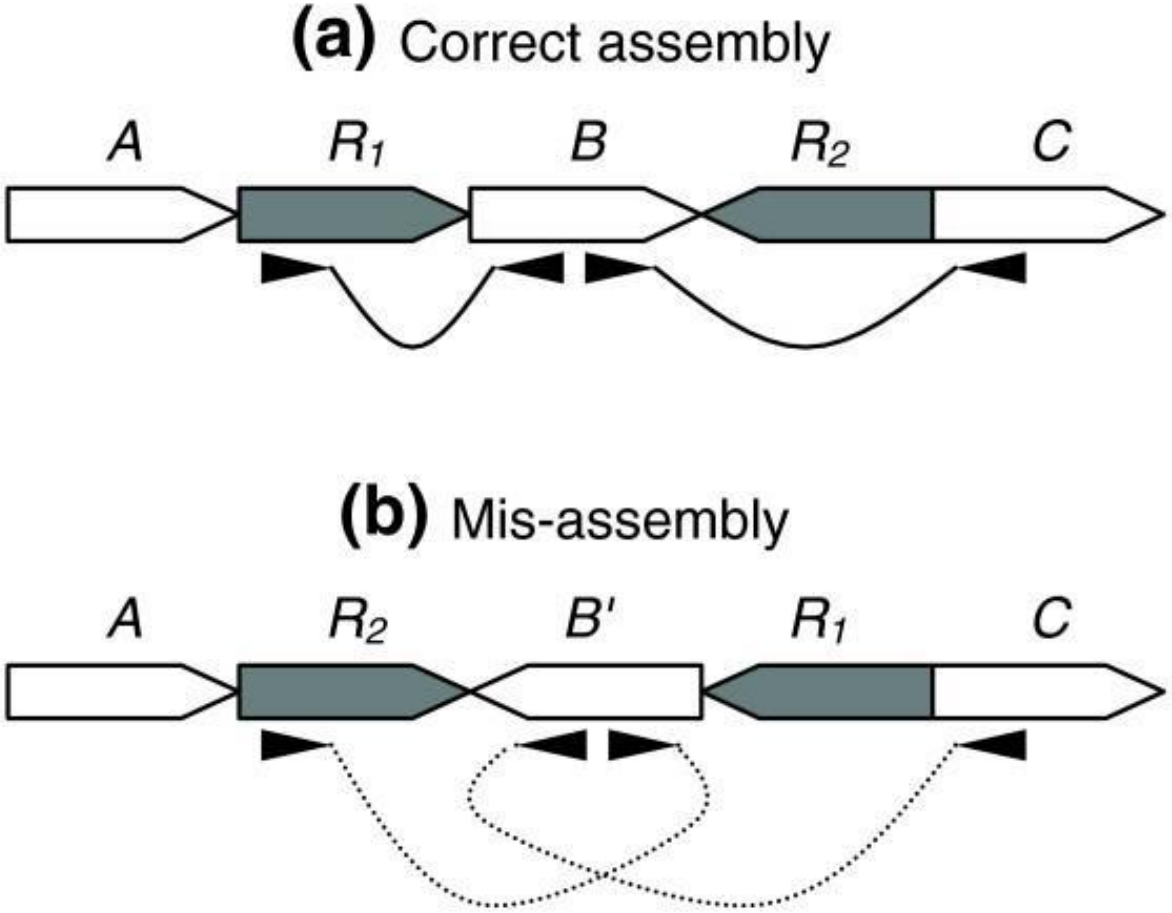
- Collapsed repeat
- Polymorphic region assembled as 2 or more copies (like the Phrap assembly in *Computer* paper)
- Rearrangement of sequence in between repeats
- Inversion misassembly

Rearrangement misassembly



from Phillipy et al., *Genome Biology*, 2008

Inversion misassembly



from Phillipy et al., *Genome Biology*, 2008

Why is it important to find misassemblies?

All downstream analysis is affected by misassembly

- Orthology/paralogy
- SNPs
- Synteny

What tools do we have to detect misassemblies?

Sequence read data: mate pair information (distance and orientation), coverage

We need to picture the **whole** assembly: consensus sequence and the multiple alignment of reads.

An AMOS bank can represent this.

Brainstorm

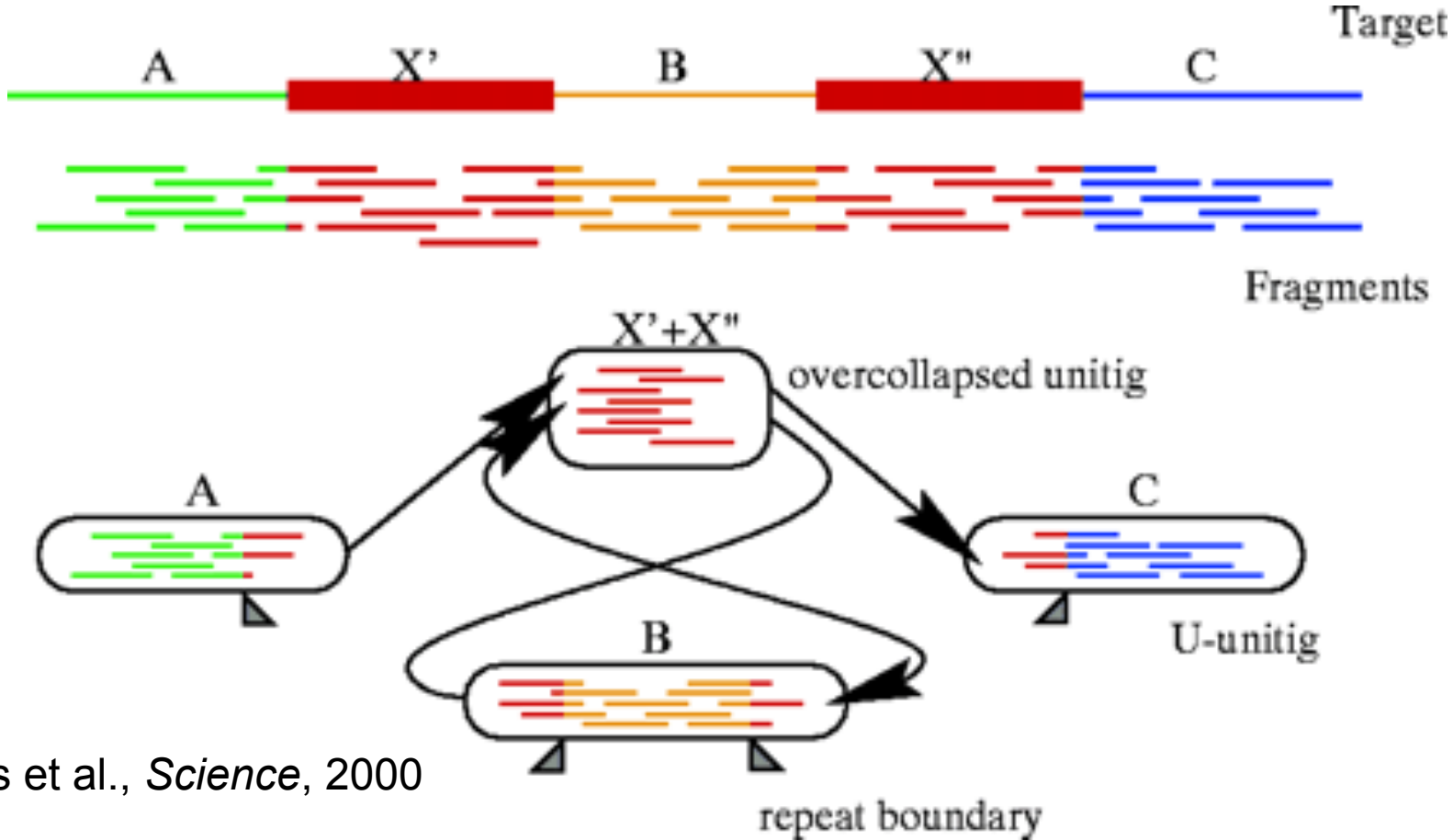
- Suppose you have the following information available to you:
  - How many reads go to a specific region of a contig
  - The known sizes of paired end reads
  - Alignment data between two reads using semiglobal alignment
  - Read orientation
- What are some metrics you can use to find misassemblies?

Statistical methods

- A-statistic (TIGR assembler)
- C/E statistic (Yorke et al. at UMD)
- Good-minus-bad (Sun et al. at IU Bloomington)

A-statistic

- We expect reads to be sequenced randomly from the genome.



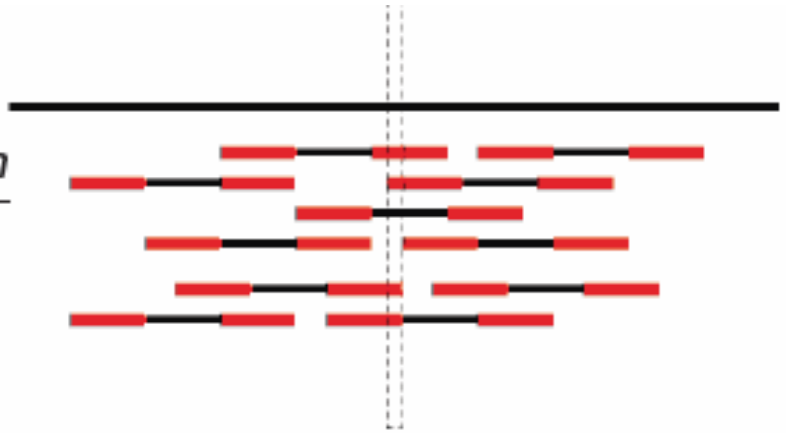
Myers et al., *Science*, 2000

C/E statistic

Compression/Expansion

Zimin et al., *Bioinformatics* 2008

$$CE = \frac{samplemean - populationmean}{\frac{populationstddev}{\sqrt{samplesize}}}$$

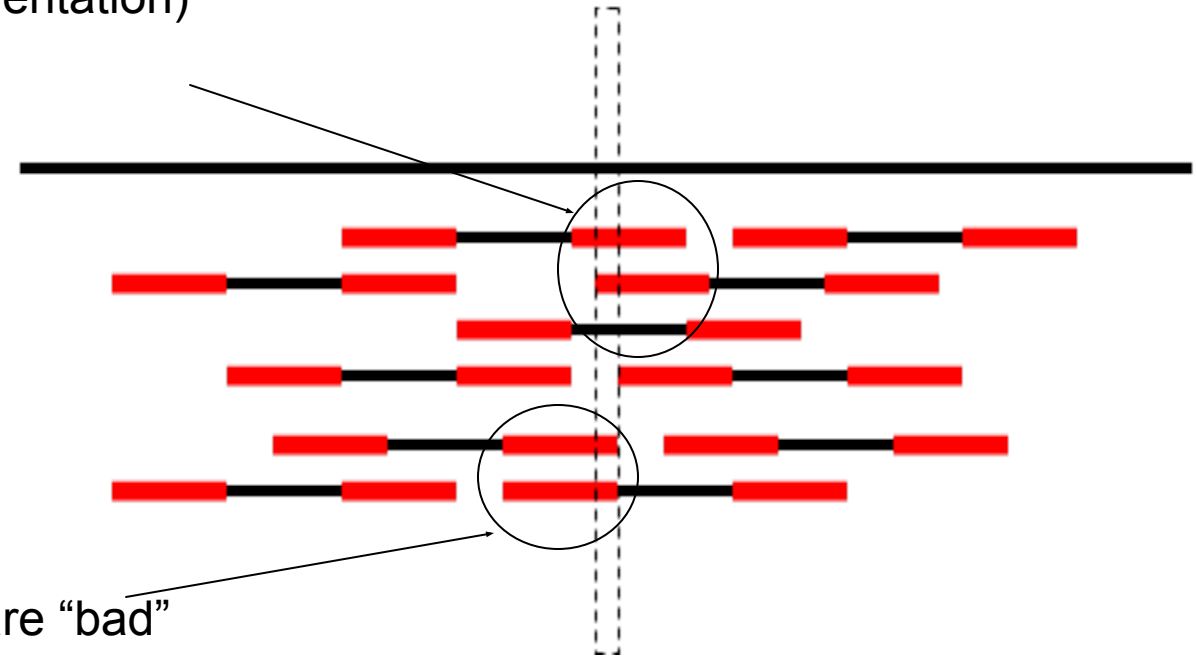


- 3 indicates expansion,
- < -3 indicates compression

Good-minus-bad

These three mate pairs are “good”

(correct distance and orientation)



These two mate pairs are “bad”

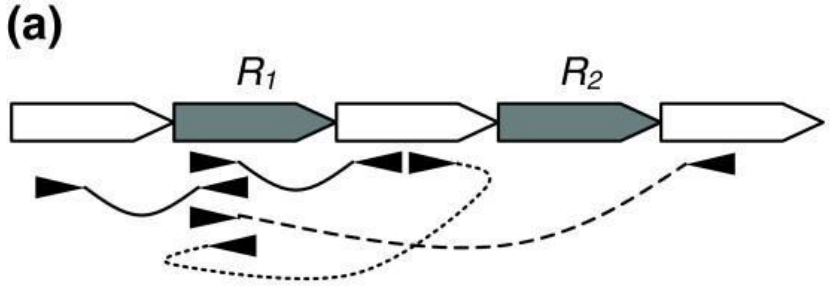
(incorrect distance or orientation)

Good-minus-bad = 3-2 = 1

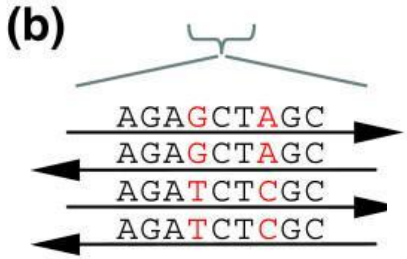
Combination methods

- The problem with statistical methods is that they generate many false positives.
- Machine-learning approach (Jeong-Hyeon Choi)
  - Calculate several statistics across the genome
  - Create a classifier using a labeled training set that combines the different statistics to more confidently predict mis-assemblies
- amosvalidate (Adam Phillipy)
  - Detect “features” using several approaches
  - Combine nearby features into “suspicious” regions
  - Written in the AMOS framework – takes an AMOS bank as input

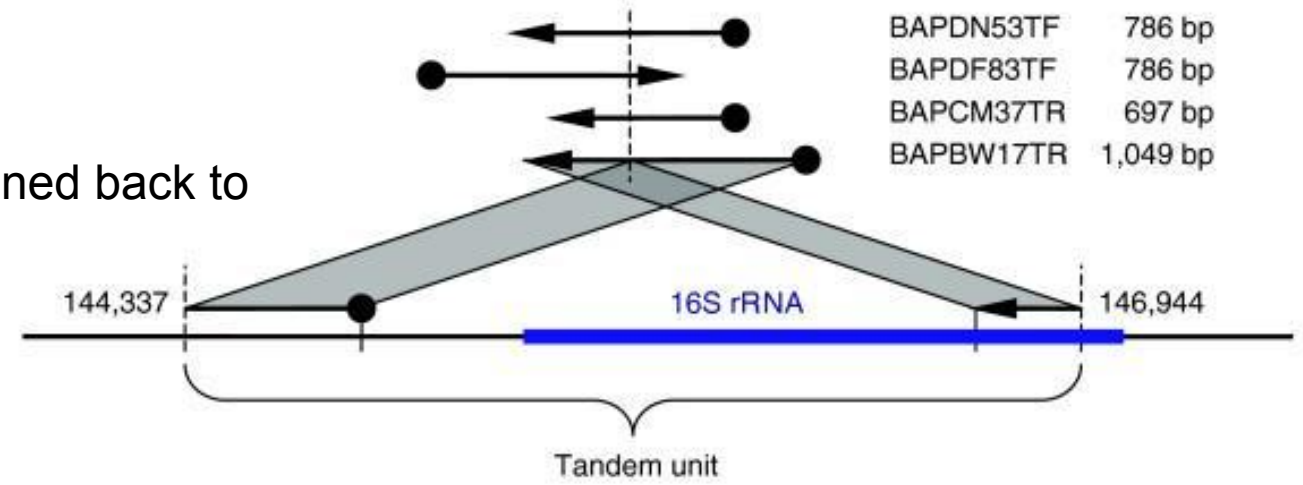
Amosvalidate: combine many types of evidence



Mate pair distance, orientation, and correlated SNPs



Unassembled reads, aligned back to the assembly



from Phillipy et al., *Genome Biology*, 2008

Assembly Visualization

- EagleView
- Consed
- Hawkeye

Hawkeye: Scaffold View

Coverage

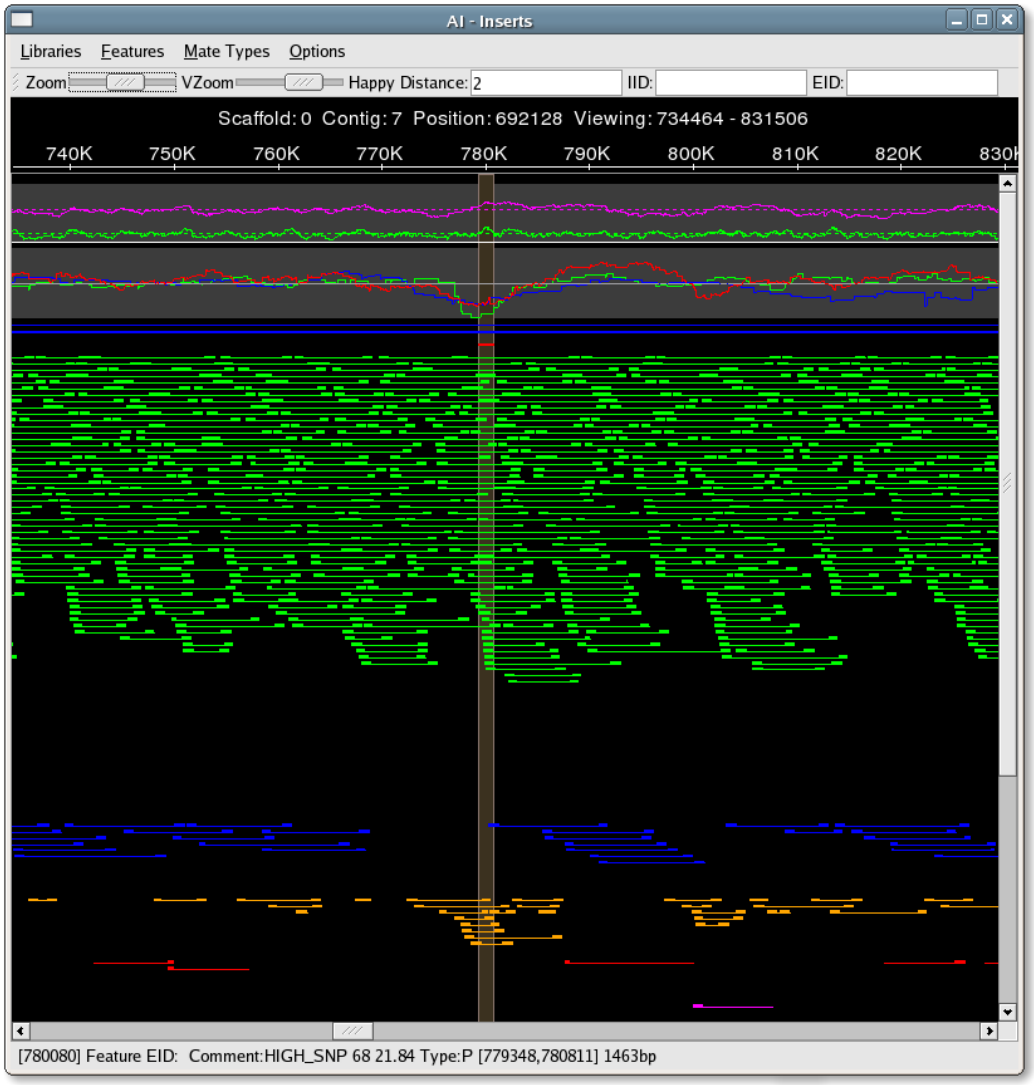
CE Statistic

Happy

Stretched

Compressed

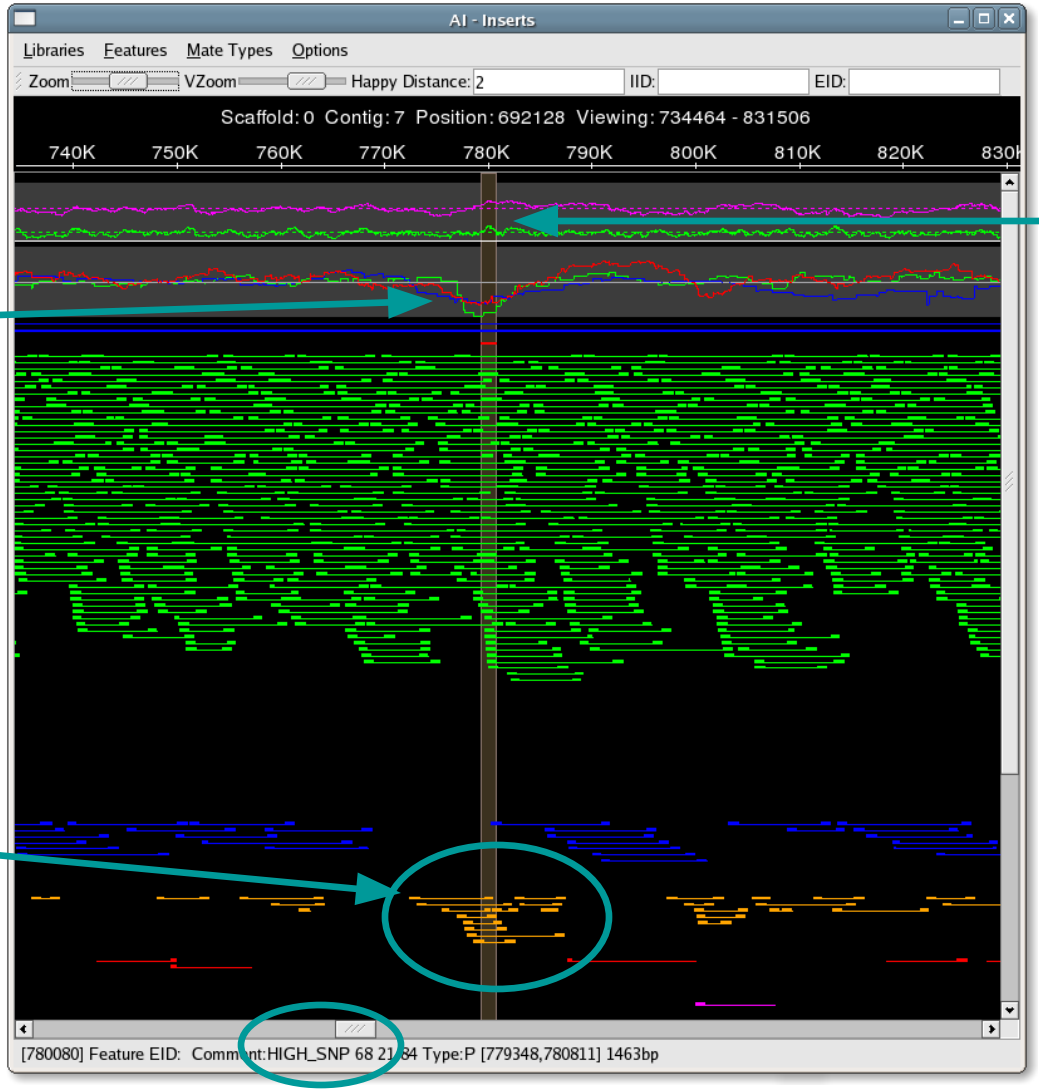
Misoriented



SNP Feature

Linking

A Collapsed Repeat



Read Coverage Spike

-5.5 CE Dip

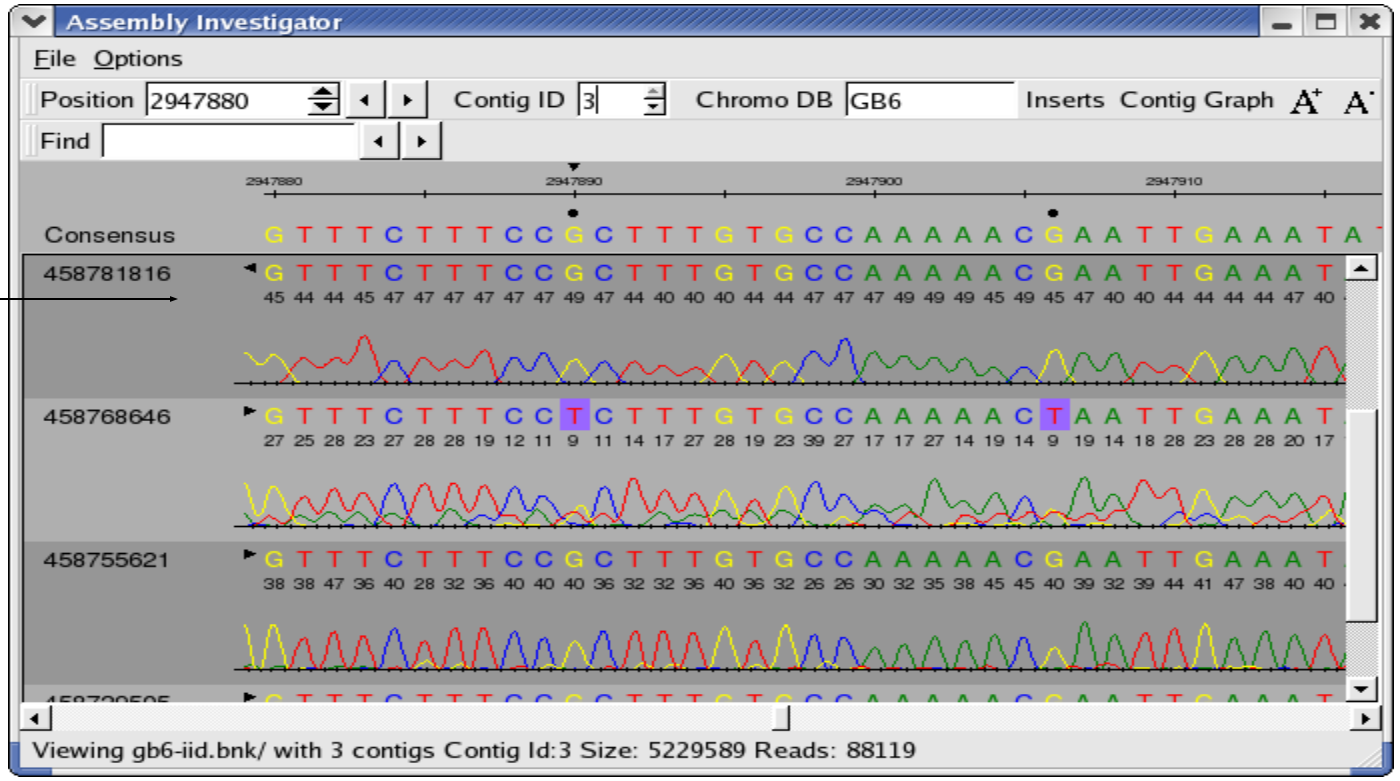
Compressed Mates Cluster

68 Correlated SNPs

Hawkeye Contig View

Quality Values

Normalized Chromatogram

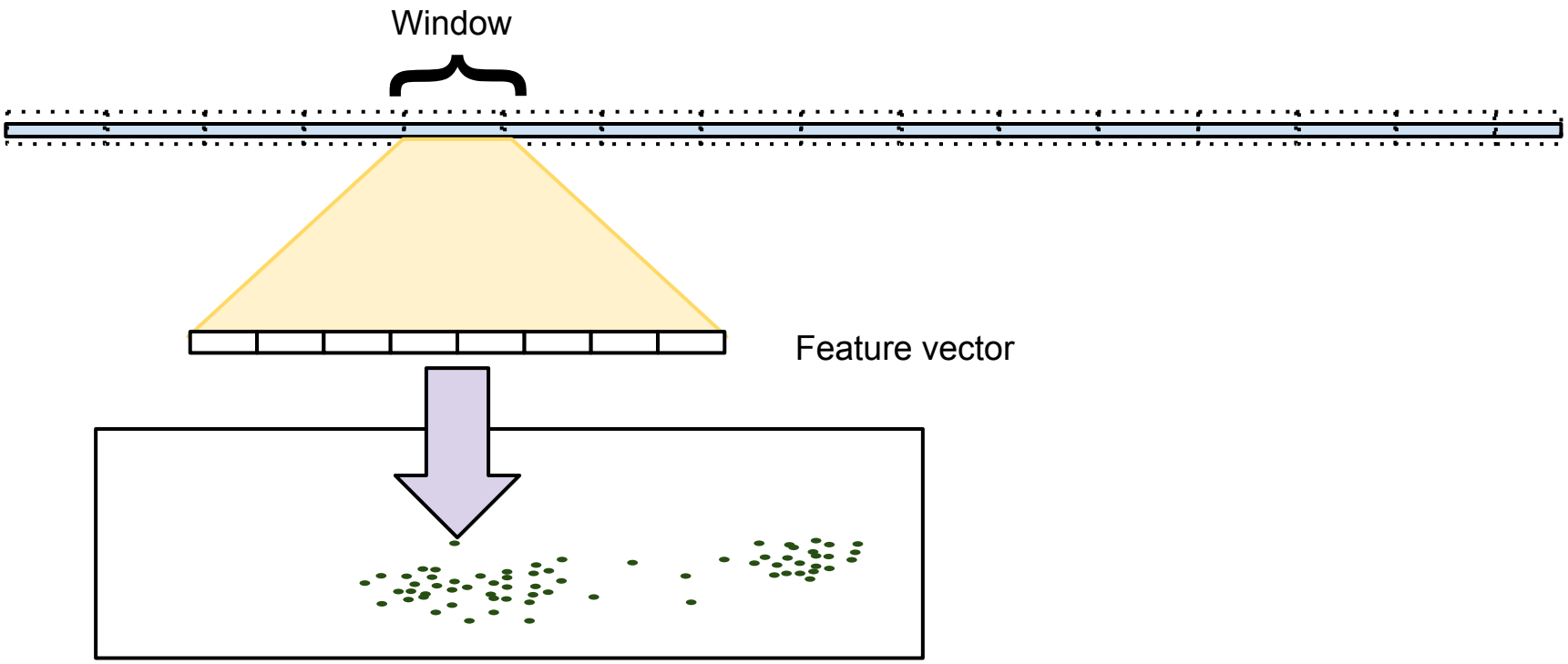


Takeaway

- Verifying assembly “correctness” is a real challenge and an open problem.
- Assembly errors do occur and need to be considered in downstream analysis (gene finding, genome comparisons, etc.)
- How to best do this is still an open problem.

Finding errors with unsupervised learning

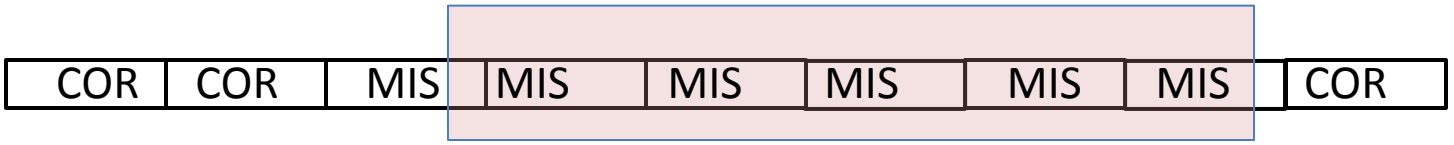
Each type of error can be quantified, and together they can generate vector that captures read behaviour for a given section of the assembly.



Data sets and evaluation

UNIVERSITY OF NOTRE DAME

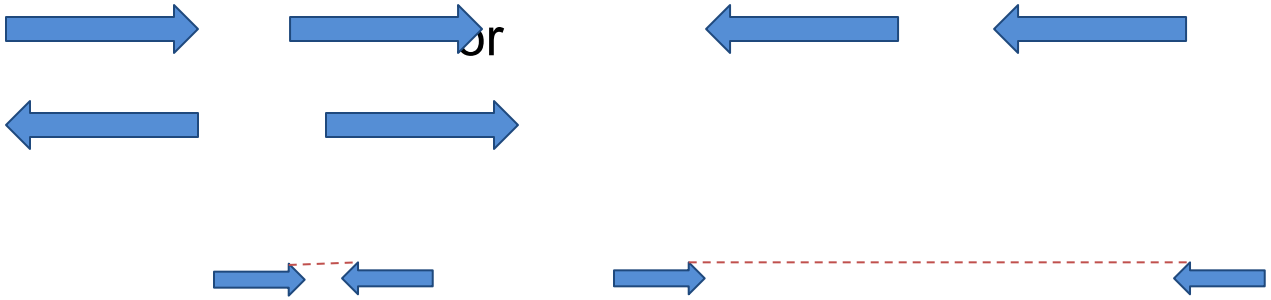
- Windows were classified as COR/MIS based on whether they intersect with *amosvalidate's* "suspicious" regions



Our Approach

These modifications yield eleven window-based features:

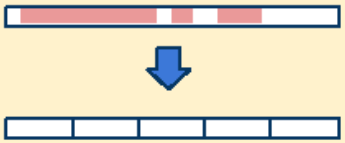
1. Read Coverage
2. Good
3. Normal
4. Outie
5. Singleton
6. Long/Short
7. Compression/expansion
8. Number of SNPs
9. Sum of SNPs
10. Linking
11. Spanning



Our Approach

UNIVERSITY OF NOTRE DAME

Refactor *amosvalidate* utilities to gather feature data on a window-by window basis

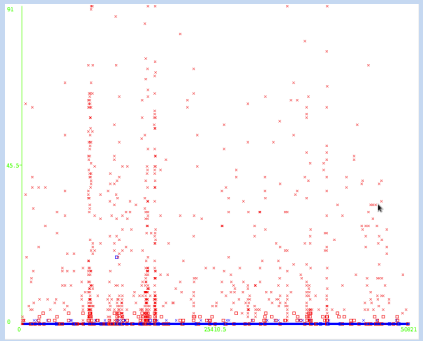


Run assembly

Produce ace, afg,asm file

Run refactored code on raw assembly file to generate input file (arff) for WEKA

Run Weka clustering



View clustering results

Visualization example - read coverage

UNIVERSITY OF NOTRE DAME