Genome assembly

Lars Arvestad

in DD2399♥BB2490

***Objective*: Reconstruct a molecule from parts**

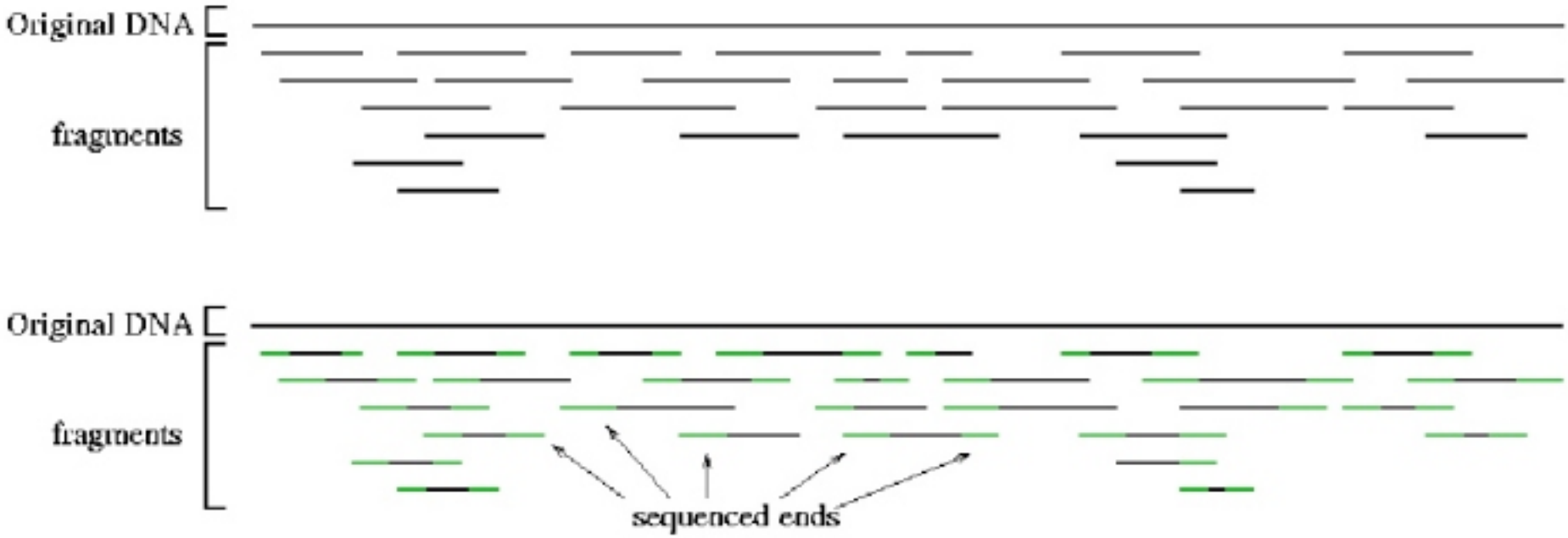
- (Gene)
- Bacterial genome
  Circular
- Eukaryotic genome
  Size?
  Haploid/diploid/polyploidy?
  Complexity?
- Genomes from a sample
  — metagenomics

Assembly applications

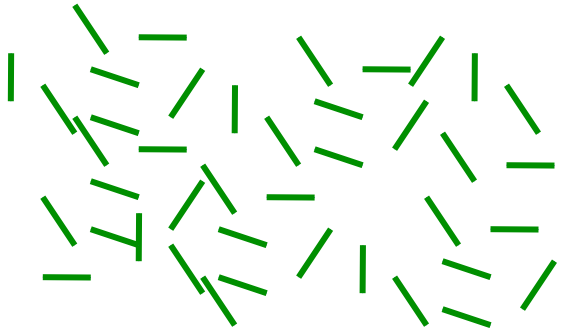
Why *you* might need to assemble

- **Get models of genomes**
  - *de novo* genome assembly
- **Fix problems** with genome models
  - When an assembly is wrong
  - When there is a region missing
- **Get models of genes and their transcripts**
  - From ”fresh” gene sequencing
  - From hits in NCBI’s Trace Archive: sequencing projects deposit early
- **Structural variant analysis**
  - Find reads from region that may differ from reference
  - Reassemble

Shotgun sequencing



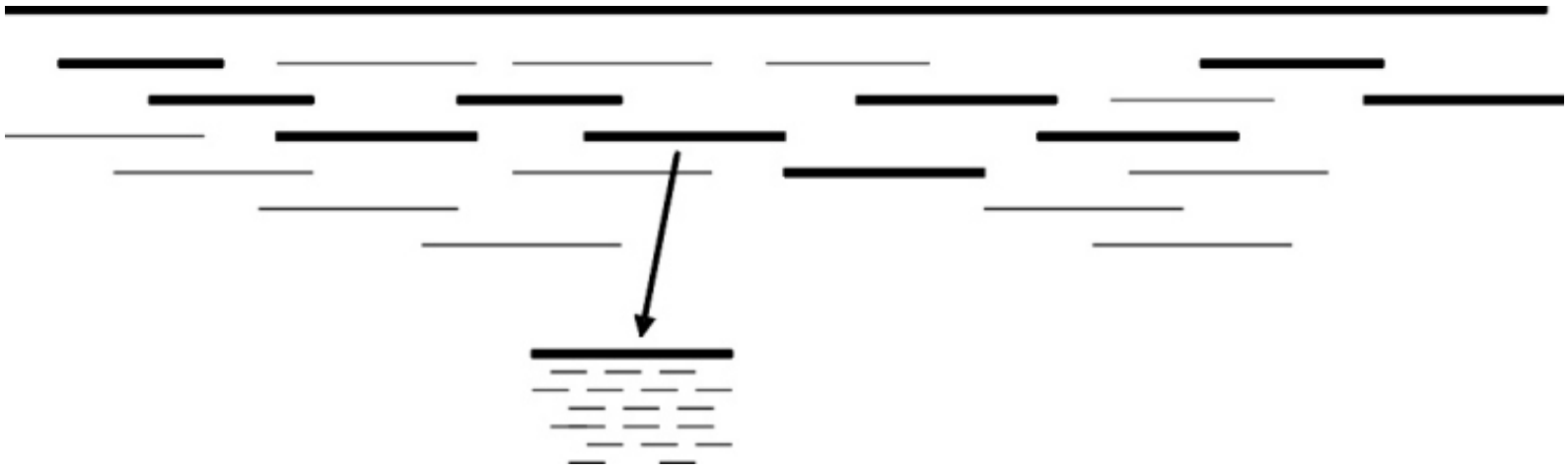
What you see:



BAC-to-BAC sequencing

- … *or* compartmental sequencing
- … *or* hierarchical sequencing

1. Break genome into large fragments, eg Bacterial Artificial Chromosomes (BACs)
2. Order the BACs and choose a ”tiling” of the genome. Requires a *mapping* of the genome!
3. Sequence the BACs



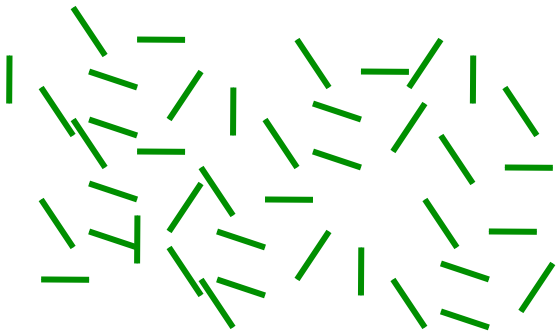
Whole-genome shotgun

- All sequencing directly on whole genomes or whole chromosomes — avoids BACs and their mapping
- One huge computational problem instead of many small BAC problems

Core problem: Assemble the shotgun pieces

In:

A set of reads of unknown orientation



Out:

Ideally: a genome model

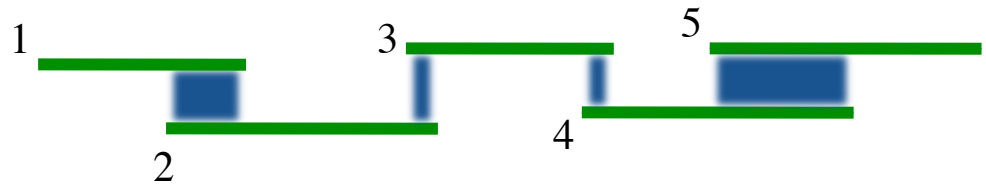
In practice:

A set of *contigs*

…and a lot of ”chaff”

Greedy assembly

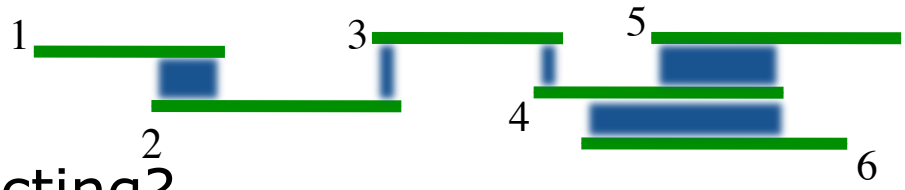
- While there are sequences with overlap:
  - Find sequences with largest overlap
  - Merge those sequences



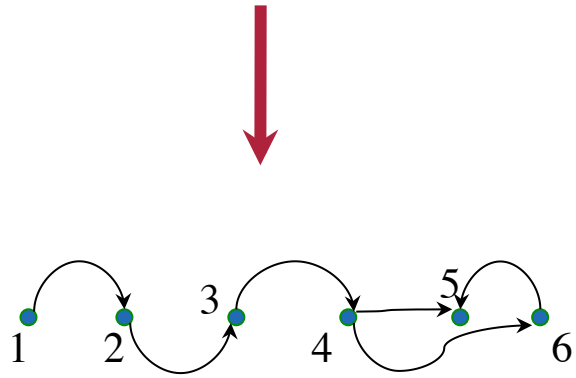
- **Advantage**:
  - Simple
- **Disadvantage**:
  - Early mistakes create bad assemblies
  - A lot of comparisons

Overlap-Layout-Consensus

- **Clean your input**
  Remove ”vector sequence”, low quality, etc



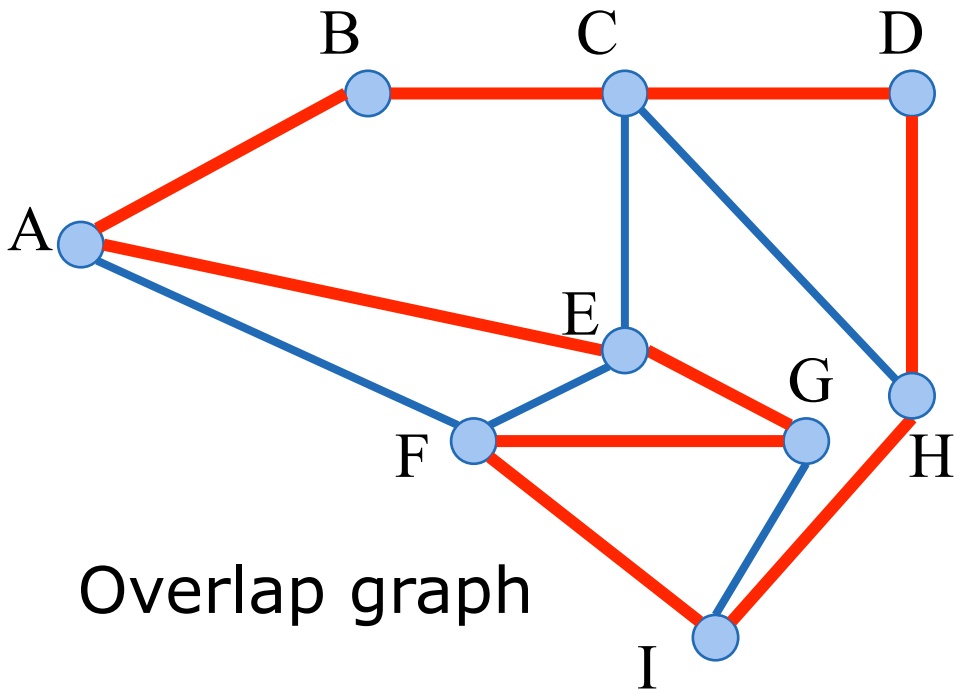
- **Overlap**: What reads are intersecting?
  - Create a node for each read
  - Create directed edge for each overlap



- **Layout**: How combine the reads?
  - Simplify graph
  - Find suitable paths in the graph
    - Hamiltonian

- **Consensus:** Derive contigs from layout

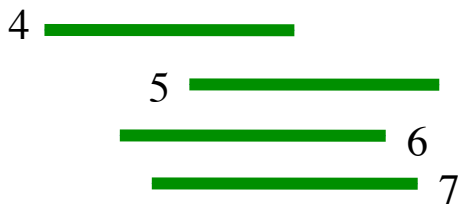
The layout stage



Overlap graph

From http://www.cbcb.umd.edu/research/assembly_primer.shtml

Consensus stage

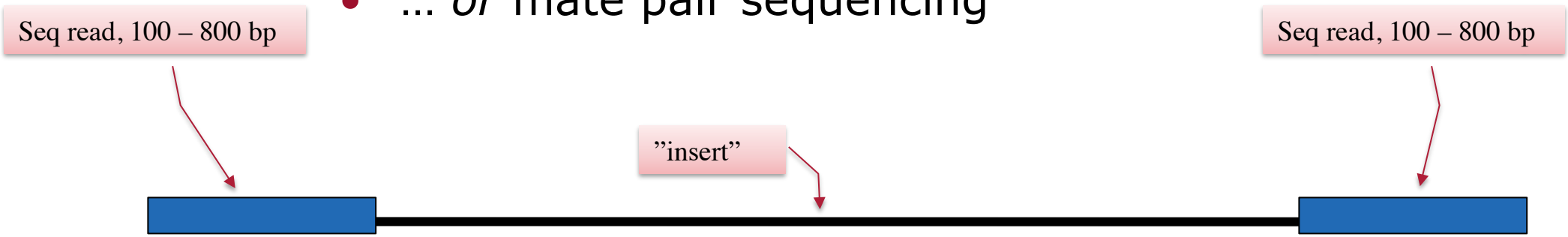



```
Seq4    TTCACACACCCTATACCAATAGTTTTCTGGCTCCTGACCATCAAACTG
Seq5                          TTTTCTGGCTCCTGACCTTCAAA-TGCCTCCATATGACTGTGCTCT
Seq6                  TACCAATAGTTTACTGGCTCCTGACCCTCAAACTGCCTCC
Seq7                      ATAGTTTTCTGGCTCCTGACCGTCAAACTGCCTCCATATGA
```

```
Cons    TTCACACACCCTATACCAATAGTTTTCTGGCTCCTGACCNTCAAACTGCCTCCATATGACTGTGCTCT
```

Paired ends sequencing

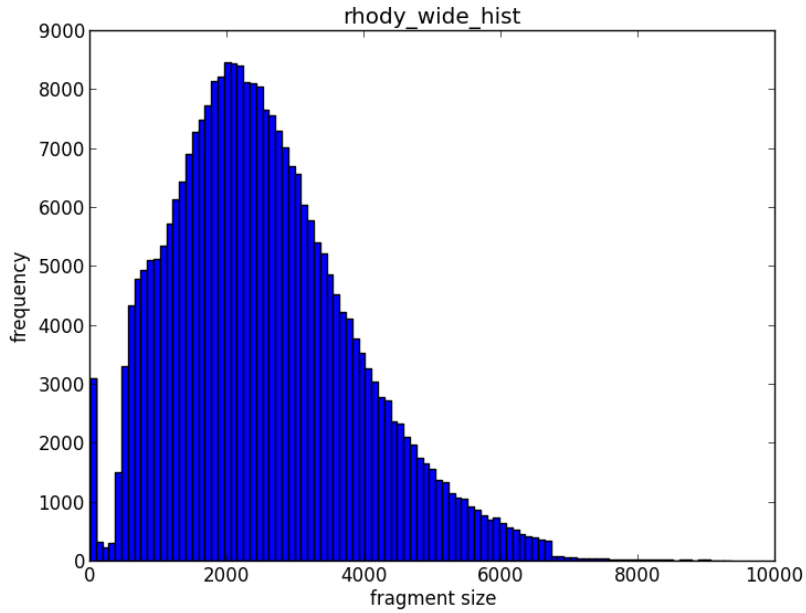
- … *or* mate pair sequencing



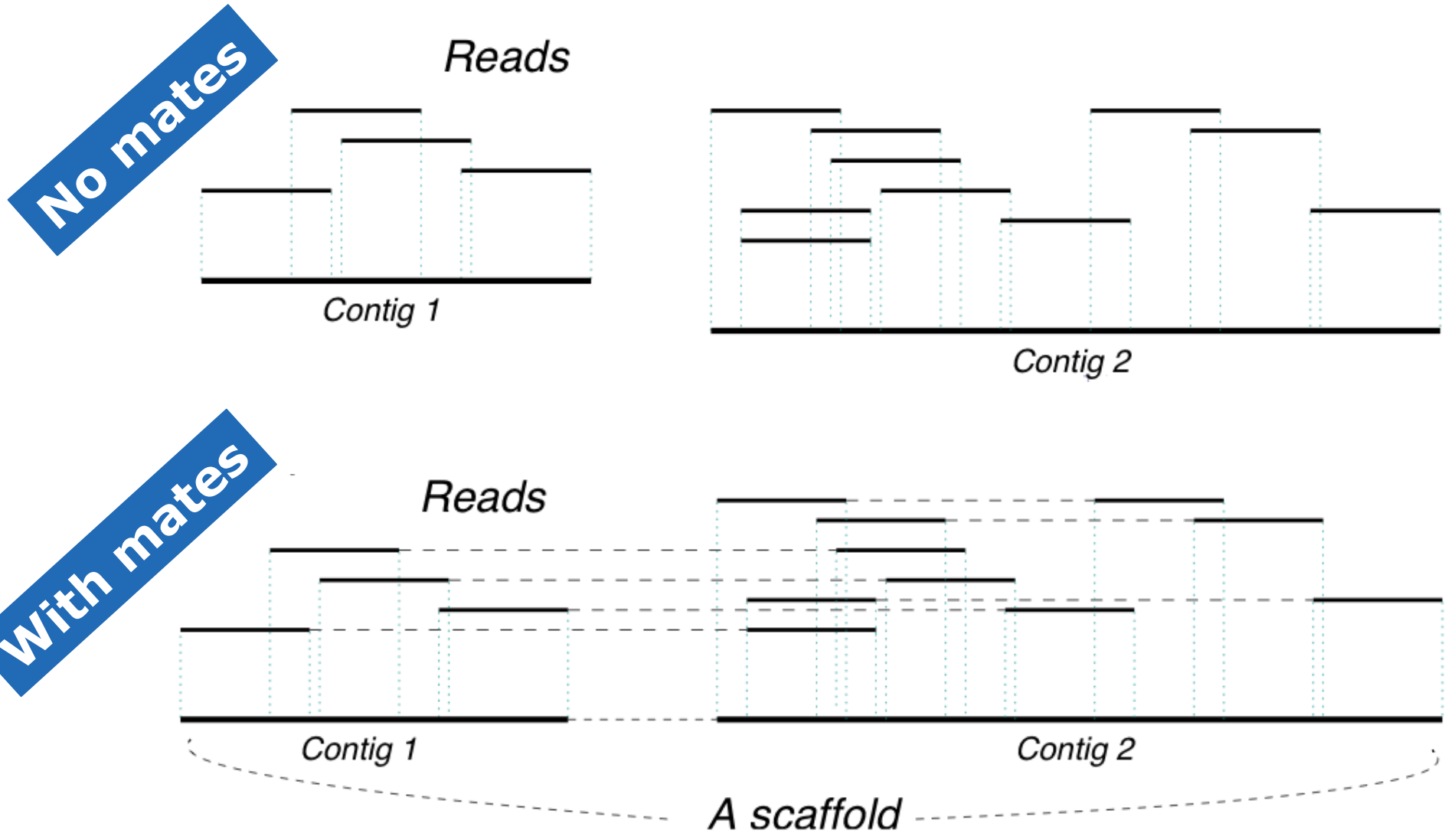
- **Advantage**: adds constraints

Insert sizes

- Paired ends: 30 – 700 bp
- Mate pairs: 2 kbp – 10 kbp



Value of read/mate pairs?

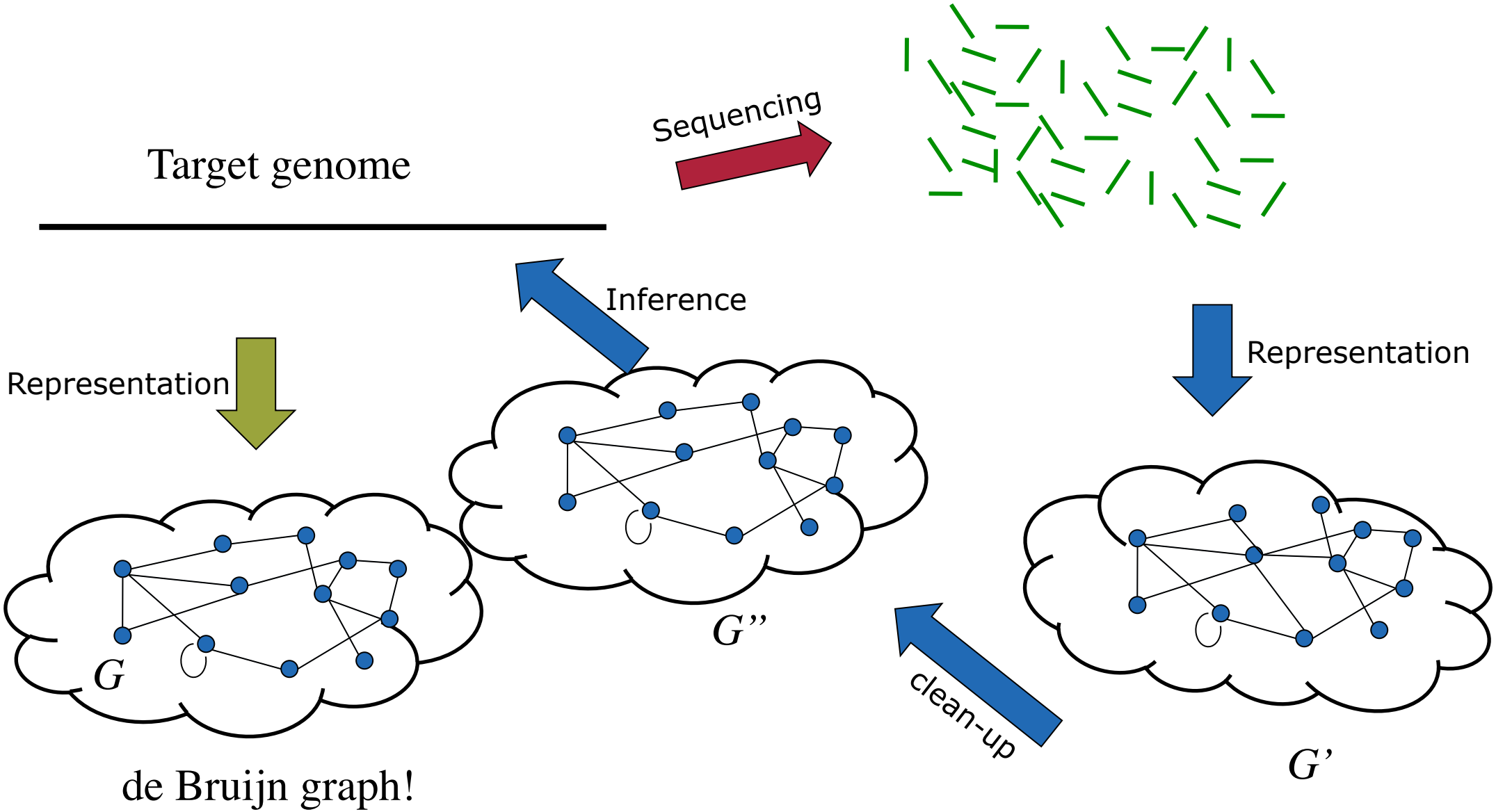


Scaffolds

- … *or* super-contigs

CCCAGTCCATTCTCCCACTGATGTCTGTAACTATATTCATCAATCTCTTT
GATTTCCAAAGCGCATACCATGGCCTCTGAATGTACTTTGCAGCTTGCCC
TTCACACACCCTATACCAATAGTTTTCTGGCTCCTGACCATCAAACTGCC
TCCATATGACTGTGCTCTTGTCTTTCCTTAGTTGCATGGGTGTCATCTTA
TGGGTCACGACCTTCTTAACTGGAACTTTCTTCTTATGGGAGCGATCCCA
TTTCTTCCAACCTCTCAAAAATTCACCCTCTTCAACAATTGACGCCTCCT
CCTTAAGATGCTCAAAATCAAAATAAAACCTAAAATCCTTCCCCTCCTGA
TCCTTCCCATCCGGATTATAATTACCTGCCAAGCATATACTCAAGTCCAT
GACAAATGTCCTGTTCAAATACATAGCCTCCCCCAACCCACACAAGAAAC
TCCACATGTAATGATTCATTCCCTTGCAATAATCCCCTCCTCTTGAATAA
TACAAGTACTTCCCTTTTCTAAAGTTCGTTTCTGATCNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNCCTCTCAAACTATCGTGAGTACGGGATTATATTTCTAAAGTAAGTAAA
TTATCCAAGATGGGCCGATCCATTACGAAAAGGAAAGATAATCCACTATT
ATTTCTTAATATAAGGGCAAATTTAAATATATAAAGATGTAATTGTTGTT
GGCGAGTGCCTCTCTTGGTTGTTGAGAGTGAAATTGACAGCAAAGTTGTA
GATTGTGACAGCCAATGTAACTTATTCACAAATTGGCCTGCCAATGGTAC
ATCATGAATCGCTATGCCACATACTTGATTATACCTTCTAAAGTACCTGT
GAATTTTATTTATTTTTTCCATTTTAAAAGTATGTTTTATTTGGAAAAAA
TATCAAATTATTATTTTACTATTATATTTTAATATATTCTTAAATAAAAA
ACACTATTAAAAAATATTATGCACCGCAATAATAAACACATATTAAACAA
ACAGATAATTTTTATATGGCATTTCACTATTGTTGTGGAATAATATCTTT

The Euler approach



Genome coverage

- … or read depth
- … or coverage depth
- … or redundancy

How many times is a position sequenced, on average?

- Drosophila:
- H
- D 2.59x
- Mouse lemur: 1.93x

Read-length matters!

Short-read technology:

- Panda: > 50x

How good coverage do you need?

- High coverage good, but expensive
- What if I want at least 99 % of the genome?

Lander-Waterman model

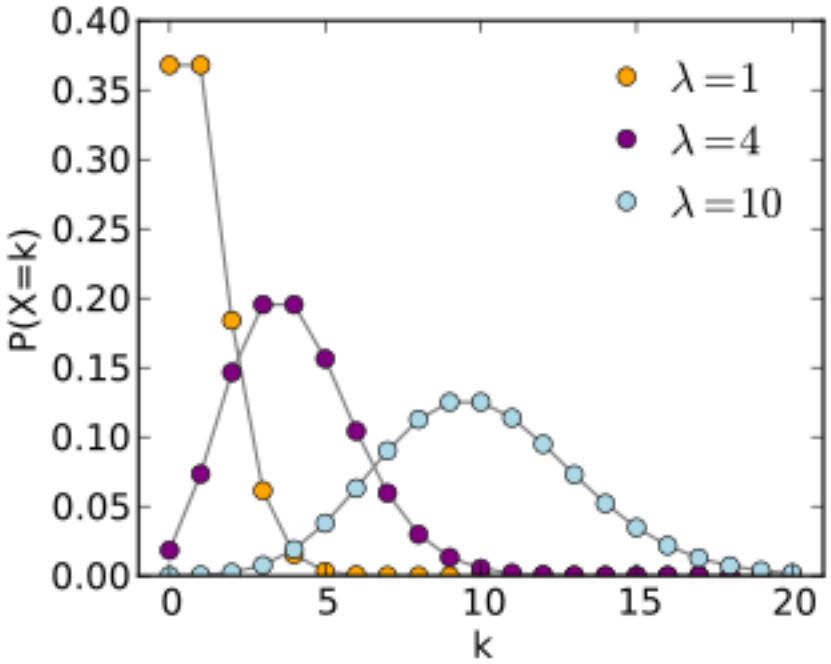
- **Assumption**: Reads are uniformly distributed
- Coverage C
- #times position i sequence: X_i
- X_i is Poission distributed

$$\Pr(X_i = k) = C^k e^{-C}/k!$$

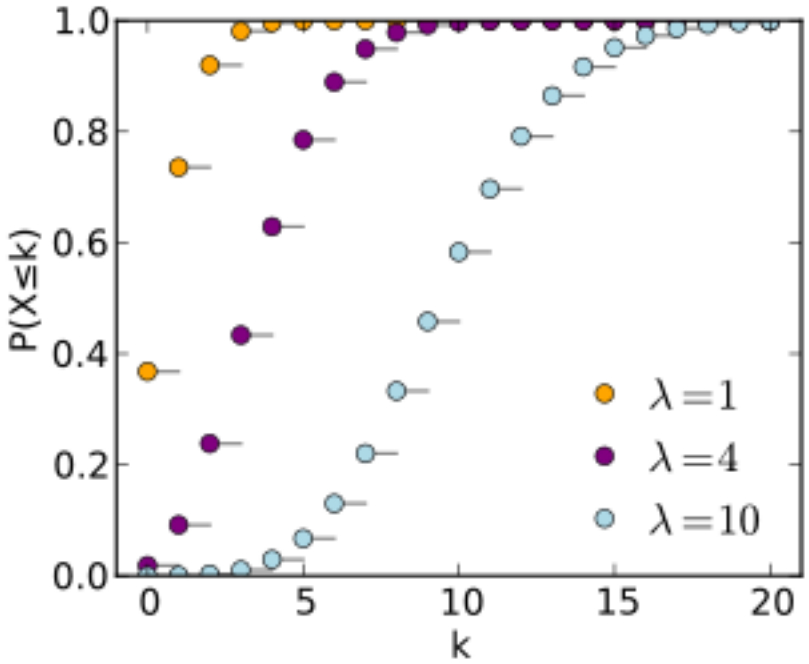
The Poisson distribution

- You won't ever get perfect coverage!

Density function



Cumulative probability



Pics from Wikipedia

More Lander-Waterman

Require $0 < \theta < 1$ overlap to join reads into a contig.

- Expected number of contigs if N reads: $Ne^{-C(1-\theta)}$

  Dog: 8x, require e.g. 10% overlap, 32×10^6 *reads:*

  24 000 contigs

- Expected contig size: $L\frac{e^{C(1-\theta)}-1}{C} + \theta$.

  Dog, assume $L = 500$*: contigs are* $\sim$ *83 700 bp*

Lander-Waterman and reality

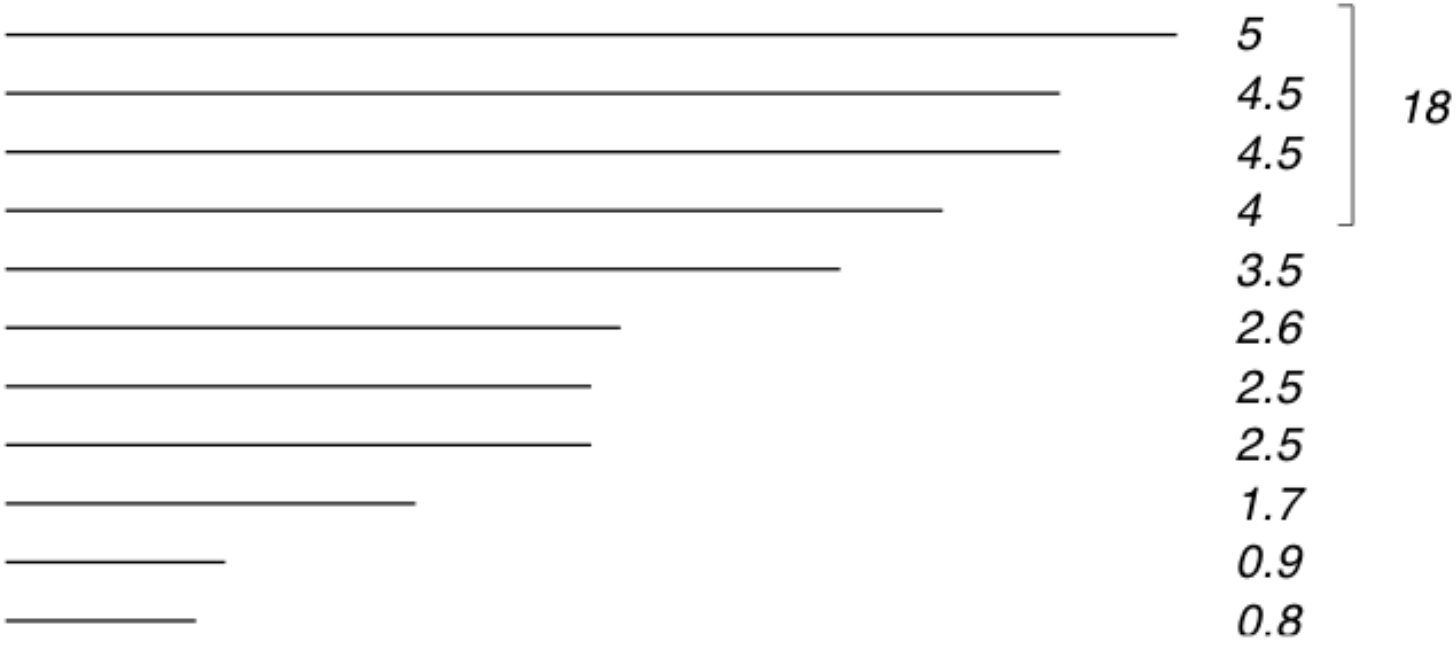
> ”For both a simulated unassisted 2x mouse genome assembly (Margulies et al. 2005) and the assisted 1.9x cat genome assembly of Pontius et al. (2007) euchromatic genome coverage by assembled contigs was only 65%, significantly less than the theoretical Poisson expectation (Lander and Waterman 1988) of 85%.”
>
> *Green, 2007*

- Why this discrepancy?

Quality: N50

N50: "covering half the assembly"
NG50: "covering half the actual genome"
Scaffold N50: Looking scaffolds, not contigs

- Operational definition:
  1. Sort all contigs by size
  2. Add contig sizes, one by one, towards the smallest
  3. Stop when you have contigs covering half the genome
  4. The length of the last contig is the N50
- Given contigs from a 30 Mbp genome:



N50 is 4 Mbp, because 5+4.5+4.5+ 4 > 30

N50 characteristics

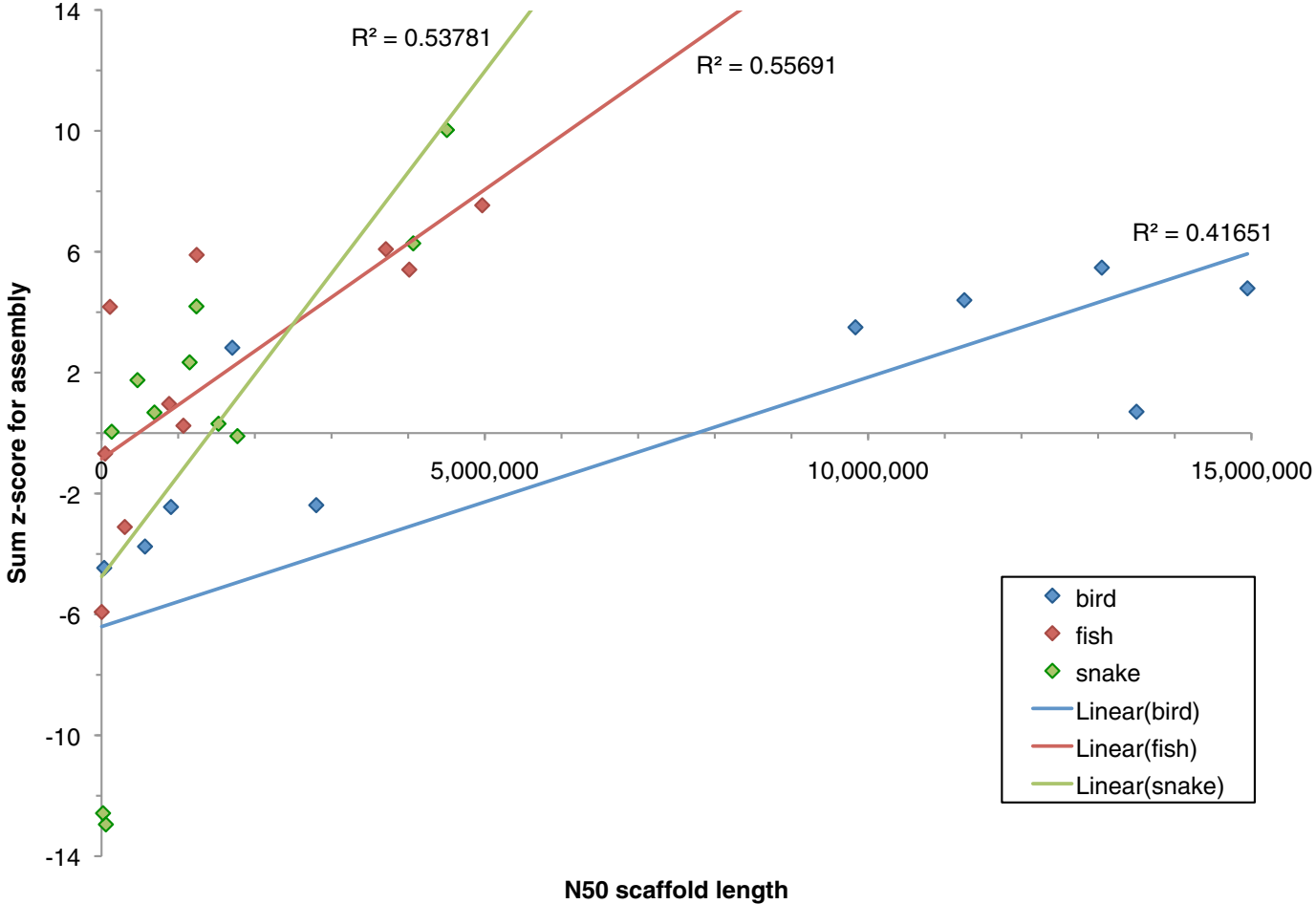
- High N50 ⇒ Good contigs ⇒ Good assembly
- Low N50 ⇒ Many small contigs ⇒ Genome badly sequenced ⇒ Bad assembly

- Bad assembly could have a high N50:

> ”The standard of judging assembly quality by size of contigs is questionable. Large contigs may simply reflect overly aggressive joining of contigs, thereby creating larger contigs with mis-assemblies. As a consequence, genome scientists who are not experts at assembly can be completely misled by statistics about contig sizes, and as a result might prefer the ’larger’ but incorrect assembly when given a choice.”
>
> Salzberg & Yorke, 2005

Utility of N50?

From the ”Assemblathon 2” genome assembler assessment (Bradnam *et al.*, Gigascience, 2013):



”[W]e find that N50 remains highly correlated with our overall rankings […]. However, it may be misleading to rely solely on this metric when assessing an assembly’s quality.”

Quality: E-size

- Definition: the size of a randomly chosen contig
- First appeared 2012
- *My opinion*: reflects fragmentation better

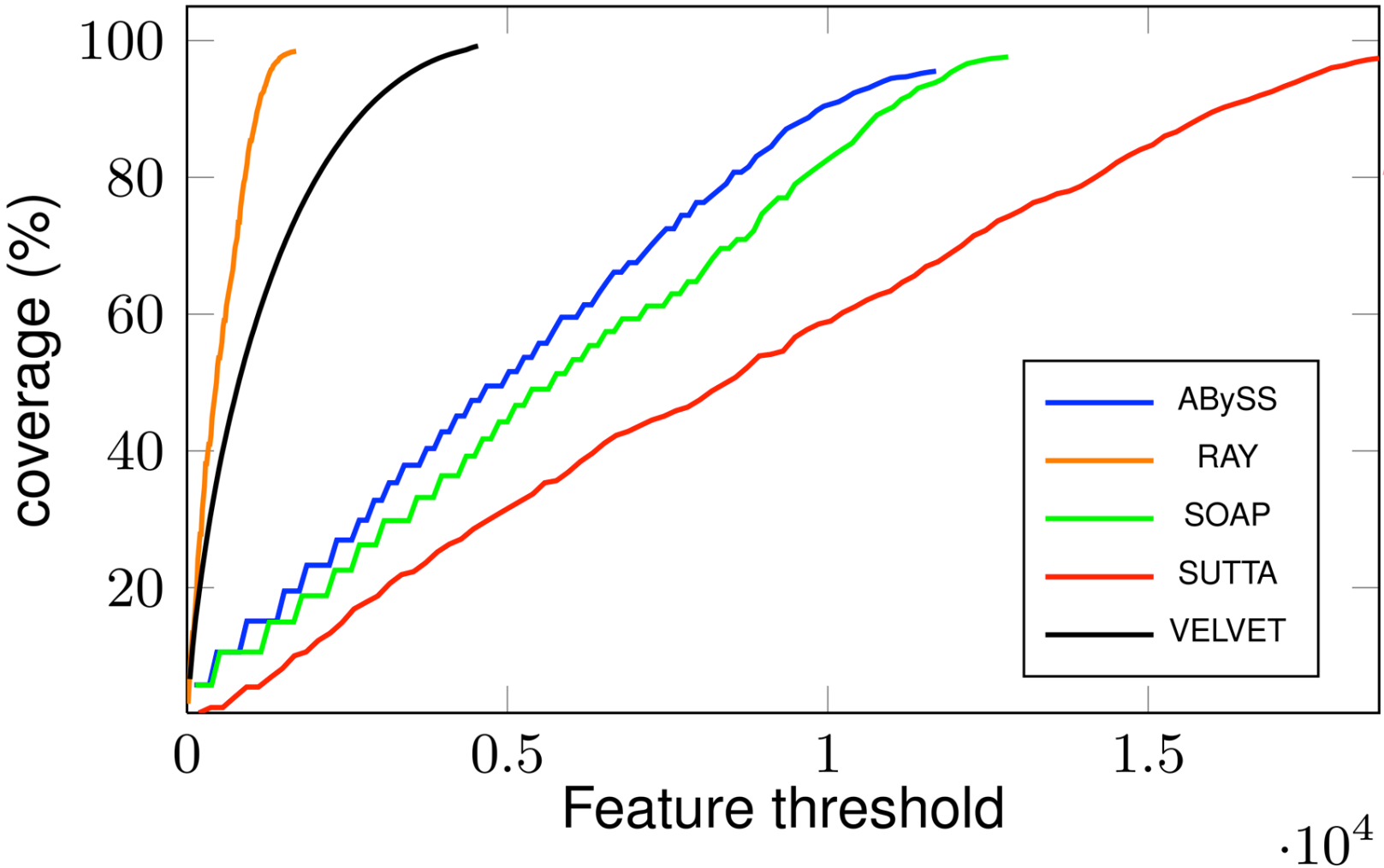
Example

- Assembly *A*:
  Contigs 10 * 100 bp
- Assembly *B*:
  Contigs 499 + 5 * 100 bp

	N50	E-size
A	100	100
B	100	166

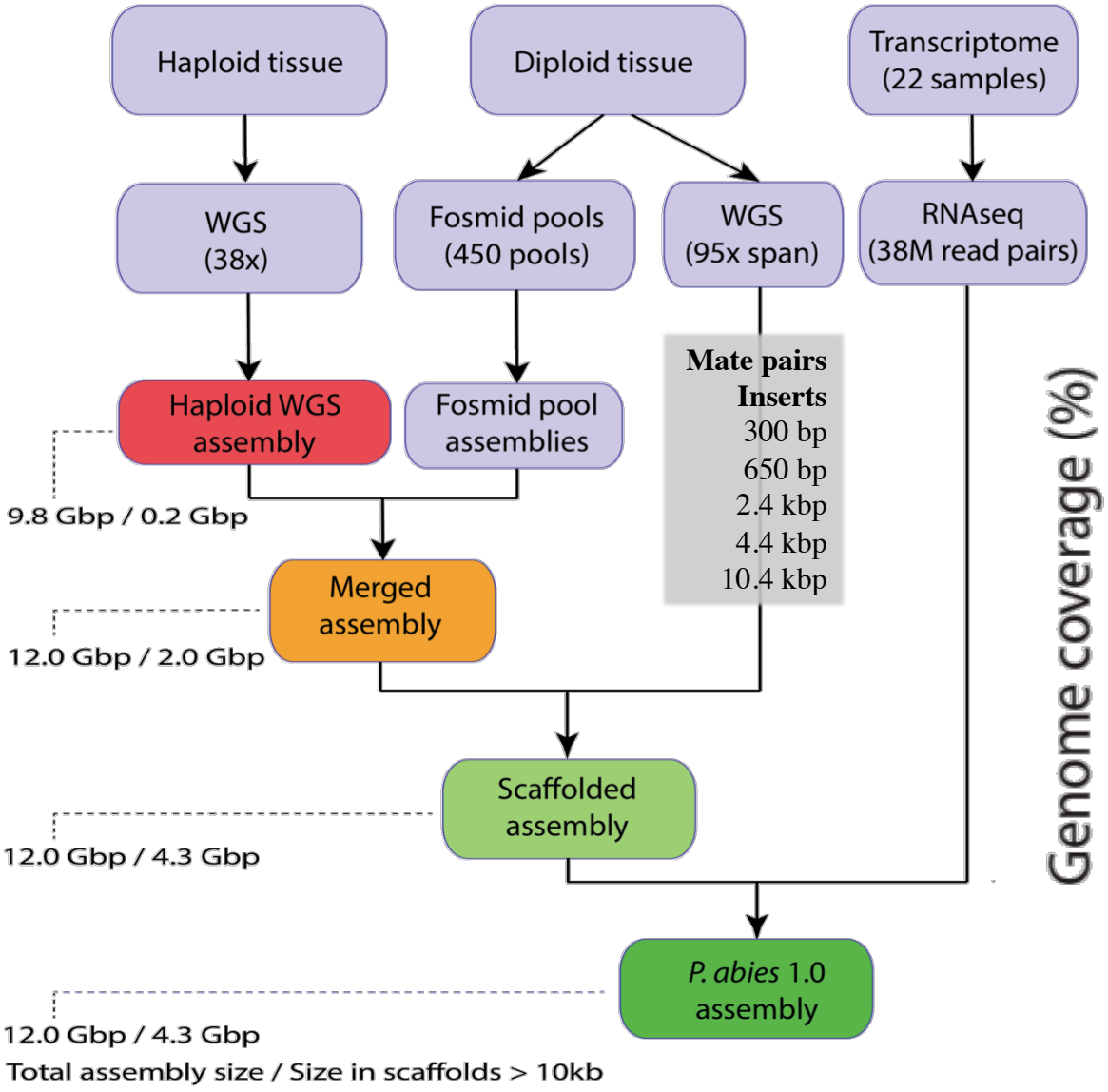
Feature Response Curve

by Vezzi, Narzisi, and Mishra, PLoS One, 2012

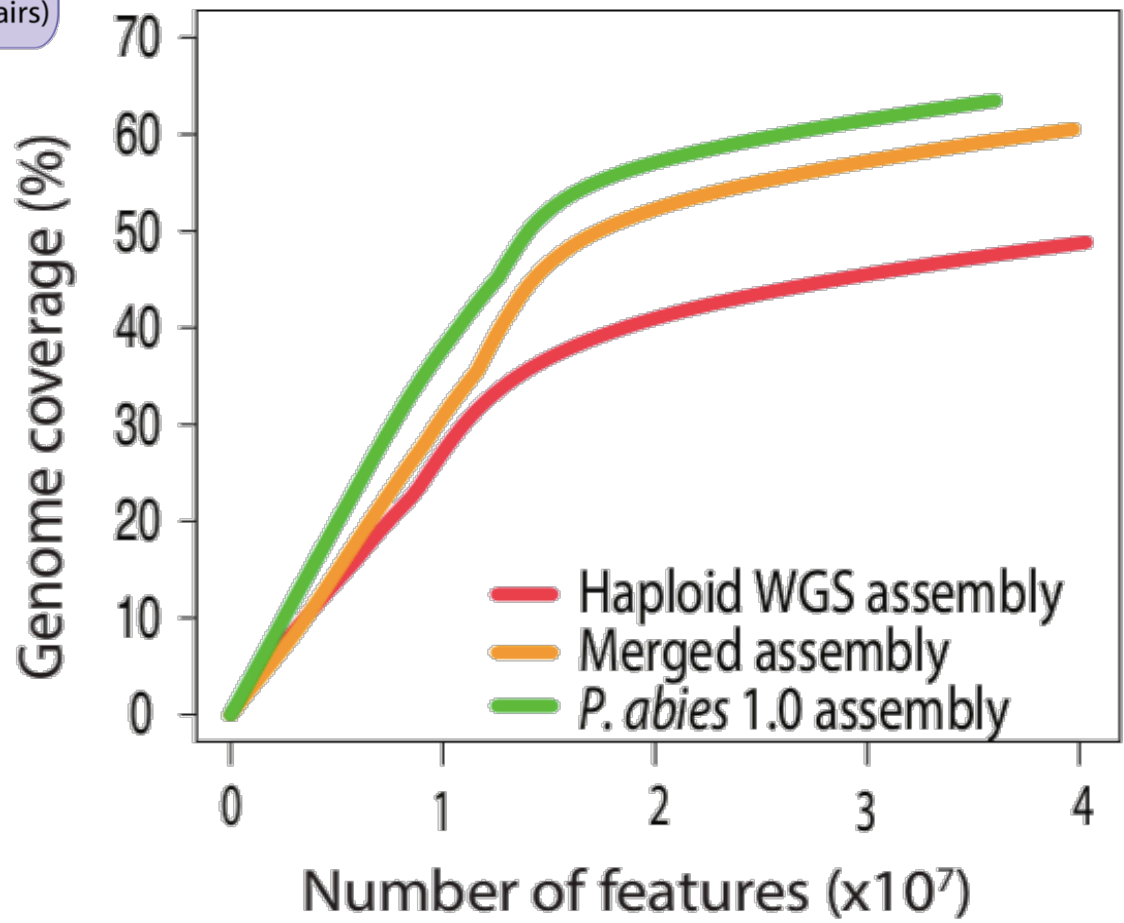


For comparing assemblies on the same data

Norway spruce: assembly process



Feature Response Curve



Assembly resources

- NCBI's Trace Archive
- Lots of assemblers
    - Cap3
    - Phrap
    - Minimus
    - Velvet
    - AllPaths
    - ABySS
    - CABOG
    - MaSuRCA
    - … and many more

Student presentations

Half of next lecture!

(Aim for 10 min presentations).

1. de Bruijn graph based assembly
   - Based on Pop's review paper (and references therein if needed!)
   - What is the basic graph construction?
   - How do you find an assembly in a de Bruijn graph?
   - What are the major problems with this approach?
   - How do small read errors affect assembly?

2. Assembly comparison/evaluation
   - Based on Vezzi *et al* " Feature-by-Feature – Evaluating *De Novo* Sequence Assembly"
   - What "features" are they using?
   - How do they compute the graphs?
   - Any limitations?