Genome Assembly Tutorial

Deb Triant

25 October 2017

Programming for Biology

Cold Spring Harbor, NY

Genome assembly with PacBio data using Canu

Why are we using this program?

- *Relatively* user-friendly and easy to install
- Not computationally intensive with small data sets

Genome assemblers

- Other useful assemblers:

  Illumina data:

  w2rap-contigger

  https://github.com/bioinfologics/w2rap-contigger

  - can take lots of computer power to run
  - works with single paired-end library

  soapdenovo2

  https://sourceforge.net/projects/soapdenovo2

  - relatively easy to install and run
  - works with large genomes

Genome assemblers

Illumina data:

DISCOVAR denovo

https://software.broadinstitute.org/software/discovar/blog/?page_id=98

large genome assembler

ALLPATHS-lg

http://software.broadinstitute.org/allpaths-lg/blog/?page_id=12

- large genome assembler
- needs at least one mate-pair library and requires specific insert size for paired-end library

Genome assemblers

PacBio data:

falcon & falcon-unzip

https://github.com/PacificBiosciences/FALCON

http://profs.scienze.univr.it/delledonne/Papers/2016%20Chin%20NMethods.pdf

Very powerful assemblers but not easy to install or use.

Quiver/Arrow/pbalign - genome alignment and polishing. Part of PacBio Genomic Consensus package.

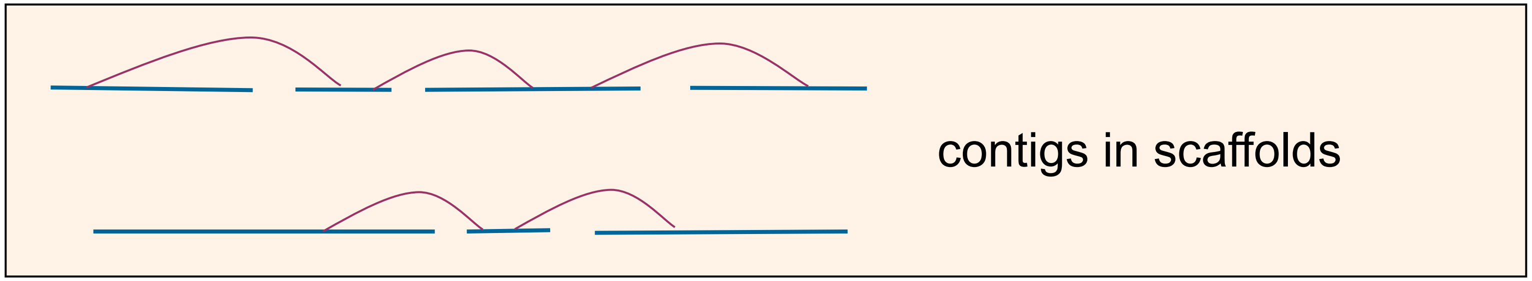
https://github.com/PacificBiosciences/GenomicConsensus

Again, powerful but not the easiest to use.

Canu - evening workshop

http://genome.cshlp.org/content/27/5/722

Linear assemblies



<u>contig</u>: a contiguous sequence of bases….

```
CTGCCCCCTGTGCCAATGGGTTTGAGGCTCTTCCCACTTCCTTTTCTATTAGATTCAATGTATCTGGTTTTATGTTGAGG
TCCTAGATCCACTTGGACTTGAGCTTTGTACAAGATGACATATATAGGTCTGTTTTTATTCTTCTACATACAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
```

<u>scaffold</u>: a sequence of contigs, separated by gaps….

```
TCCTAGATCCACTTGGACTTGAGCTTTGTACAAGATGACATATATAGGTCTGTTTTTATTCTTCTACATACAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
NNNNNNNNAGTTTTTACCACAATTGCTCTATAGTAAAGCTTGAGGTCAGGGTTGGTGATCCCTCCAGCCATTCTTTCATT
ATTAAGAATTGTTTTCCCTAGTCTGGGTTTTTTGCTTTTCCAGGCGAATTTGAGAATTGCTCTTTCCATGTCTTTGAAGA
ATTGTGTTNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNGGATTTTGATGGGGTTTGCATTGAATCTGTAGATT
GTCTTTGGTAAGATGGTTAGTTTTACTATGTTAATTCTGCCAATCCACAAGCATGGGAGCGCTCTCCATTTTCTGAGATC
TTCTTCAATTTCTTTCTTGAGAAACTTGAAGTTATTGTCATACA
```

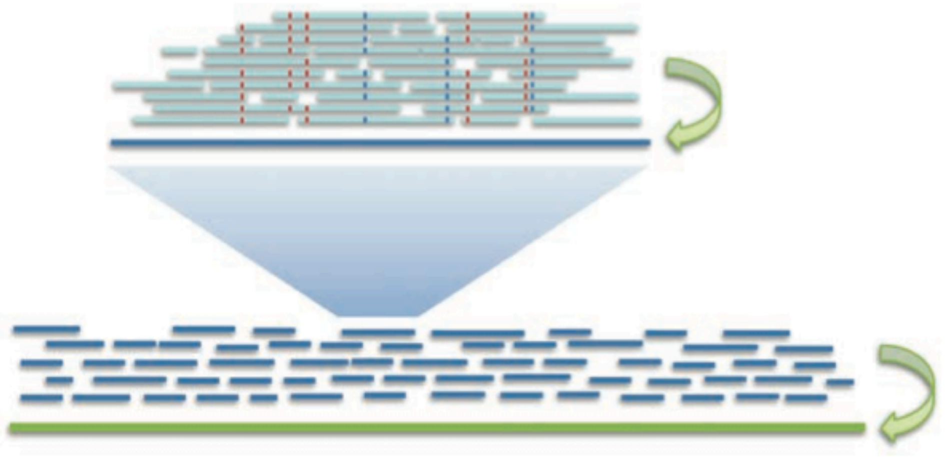
Number of Ns = predicted gap size, with error bars (can’t be displayed in fasta format)

Long read sequencing

Longest sequencing reads as seed data set

Use seed dataset to map shorter reads

Perform preassembly: construct preassembled reads from seed reads through consensus procedure



Preassembled reads used for genome assembly

Error correction: mapping high-quality short PacBio reads to long reads

PB-only Correction & Polishing
Chin *et al* (2013) *Nature Methods.* 10:563–569

Canu: “unitigs”

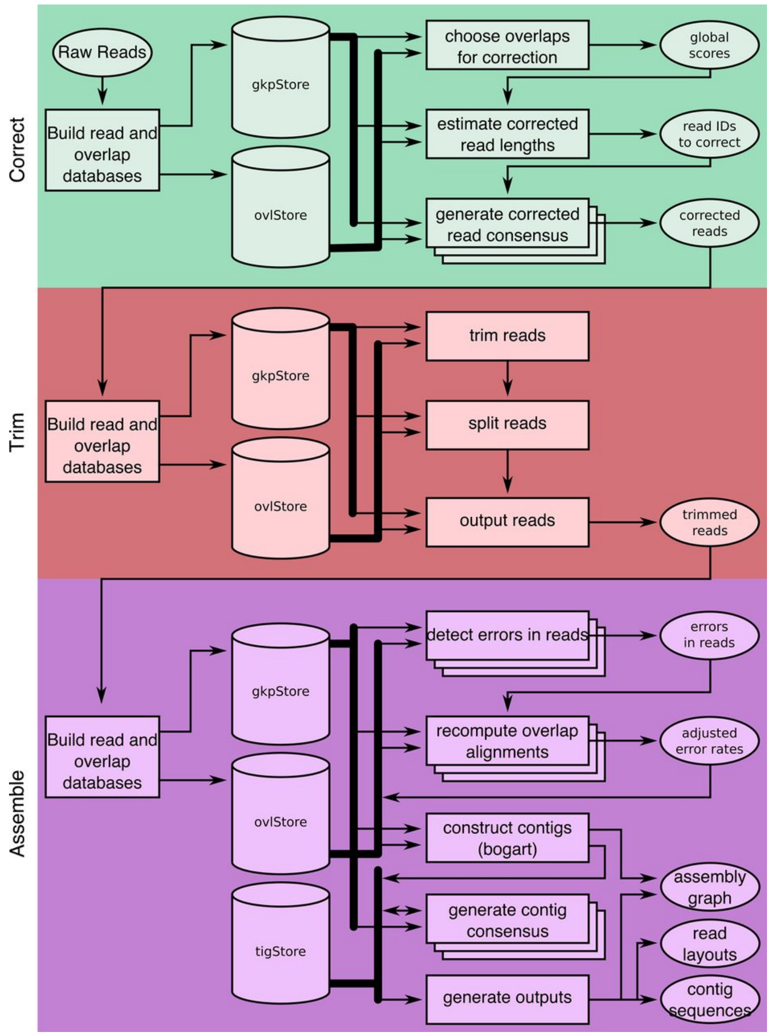
Using Canu

- Canu is an assembler that specializes in PacBio or Oxford Nanopore long-read data.
- Derived from Celera assembler.
- Corrects reads, trims suspect regions (e.g., adaptors) then assembles the corrected reads.
- Can be run with one command to do all steps or each step can be run separately (correcting, trimming and assembling).
- Koren S, Walenz BP, Berlin K, Miller JR, Phillippy AM. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Research*

  http://canu.readthedocs.io/en/latest/index.html

A full Canu run includes three stages: correction (green), trimming (red), and assembly (purple).

Generates k-mer histogram and conducts all-vs-all overlaps



Correct:
Select best overlaps to generate corrected reads. Uses longest 40X reads for correction

Trim:
Identifies unsupported regions and trims reads to longest supported range

Assemble:
Identify sequencing errors, constructs best overlap graph and outputs contigs.

Sergey Koren et al. Genome Res. 2017;27:722-736

Using Canu

- Outputs contigs and unitigs.

- The unitig construction task finds sets of overlaps that are consistent, and uses those to place reads into a multialignment layout. The layout is then used to generate a consensus sequence for the unitig or a “high-confidence contig”.

Installing Canu

- We will use the sample Pac Bio *E. coli* data found on the canu homepage: http://canu.readthedocs.io

  25x *E.coli* fastq files with quality scores.

- To install Canu:
- Make a directory for installation

```
$ git clone https://github.com/marbl/canu.git
$ cd canu/src
$ make -j 4
```

Installing Canu

- Executables found in canu/Darwin-amd64/bin
  - We need to move canu to /usr/local/bin

```
$ which canu
$ cd /usr/local/bin
```

  - Create a symbolic link to executables

```
$ ln -s ~/Deb/canu/Darwin-amd64/bin/canu
```

  - Don't forget sudo!

```
$ sudo ln -s ~/Deb/canu/Darwin-amd64/bin/canu
$ which canu
```

  - Need sudo to remove canu

Installing Canu

- Download canu test E.coli dataset

http://canu.readthedocs.io/en/latest/quick-start.html?highlight=25x

- 25X subset (223Mb)

```
$ curl -L -o pacbio.fastq
http://gembox.cbcb.umd.edu/mhap/raw/ecoli_p6_25x.filtered.fastq
```

-o naming downloaded file pacbio.fastq

How many reads within the file?

Running Canu

- First we will run default with correcting, trimming and assembling all in one command:

```
$ canu  -p ecoli -d ecoli-full  genomeSize=4.8m  -pacbio-raw \
your_path/canu/pacbio.fastq saveReads=true \
 gnuplotImageFormat=svg  **to save output 2>saveReads.log
```

```
-p assembly-prefix
-d output directory
input types:
 -pacbio-raw -pacbio-corrected -nanopore-raw -nanopore-corrected fastq
or fasta format

*gnuplot option specific to our version
```

****USE THE HELP DOCS!!!!**

Running Canu - Manual assembly

Correct, Trim and Assemble Manually

- You can also do the three top-level tasks by hand. This would allow trying multiple construction parameters on the same set of corrected and trimmed reads.

Previous command to correct, trim & assemble:

-
  ```
  canu  -p ecoli -d ecoli-separate  genomeSize=4.8m  -pacbio-raw \
  your_path/canu/pacbio.fastq saveReads=true \
  gnuplotImageFormat=svg  2>full_out.log **to save output
  ```

Let's split it up to do each step separately!

Using Canu – Manual assembly

First correct the raw reads:

-
  ```
  canu  -correct -p ecoli -d ecoli-manual  genomeSize=4.8m  \
  -pacbio-raw your_path/canu/pacbio.fastq \
  saveReads=true gnuplotImageformat=svg  2>correct_out.log
  ```

Then trim the output of the correction:

-
  ```
  canu -trim -p ecoli -d ecoli-manual  genomeSize=4.8m \
  -pacbio-corrected ecoli-full/ecoli.correctedReads.fasta.gz \
  saveReads=true gnuplotImageformat=svg  2>trim_out.log
  ```

*corrected reads created during first step

Using Canu - manual assembly

Finally, assemble the output of the trimming twice using two error rates (can use as many as you like):

-
  ```
  canu -assemble -p ecoli -d ecoli-erate-0.013 genomeSize=4.8m \
  correctedErrorRate=0.013 -pacbio-corrected ecoli-   \
  manual/ecoli.trimmedReads.fasta.gz saveReads=true   \
  gnuplotImageformat=svg  2>assemble-0.013_out.log
  ```

-
  ```
  canu -assemble -p ecoli -d ecoli-erate-0.025 genomeSize=4.8m \
  correctedErrorRate=0.025 -pacbio-corrected ecoli-   \
  manual/ecoli.trimmedReads.fasta.gz saveReads=true   \
  gnuplotImageformat=svg  2>assemble-0.025_out.log
  ```

*trimmed reads created during second step

*The error rate specifies the difference in overlap between two corrected reads which is typically <1% (canu default value 0.045) for PacBio data and <2% (canu default 0.144) for Nanopore data (<1% on newest chemistries). Higher rate, more sensitive.

*Notice in the output there are separate directories for each error rate you specify.

Canu output files - all found in helpdocs

- **ecoli*.report** - assembly analysis log

READS

- **ecoli*.correctedReads.fasta.gz** - The sequences after correction, trimmed and split based on consensus evidence. Typically >99% for PacBio and >98% for Nanopore but it can vary based on your input sequencing quality
- **ecoli*.trimmedReads.fasta.gz** - The sequences after correction and final trimming. The corrected sequences above are overlapped again to identify any missed hairpin adapters or bad sequence that could not be detected in the raw sequences.

SEQUENCE

- **ecoli*.contigs.fasta** - Everything which could be assembled and is part of the primary assembly, including both unique and repetitive elements.
- **ecoli*.unitigs.fasta** - Contigs, split at alternate paths in the graph
- **ecoli*.unassembled.fasta** - Reads and low-coverage contigs which could not be incorporated into the primary assembly.

How many contigs to we have? How many unitigs?

Canu output files - all found in helpdocs

SEQUENCE

- **ecoli*.contigs.fasta** - Everything which could be assembled and is part of the primary assembly, including both unique and repetitive elements.
- **ecoli*.unitigs.fasta** - Contigs, split at alternate paths in the graph
- **ecoli*.unassembled.fasta** - Reads and low-coverage contigs which could not be incorporated into the primary assembly.

The header line for each sequence provides some metadata on the sequence:

tig # len=<integer> reads=<integer> covStat=<float> gappedBases=<yes|no> class=<contig|bubble|unassm> suggestRepeat=<yes|no> suggestCircular=<yes|no>

Canu output files - all found in helpdocs

GRAPHS

- **ecoli*.**gkpStore **-** *svg

LAYOUT

The layout provides information on where each read ended up in the final assembly, including contig and positions. It also includes the consensus sequence for each contig.

- **ecoli*.unitigs.layout.readToTig -** The position of each read in a contig
- **ecoli*.contigs.layout.tigInfo**, **ecoli*.unitigs.layout.tigInfo -** A list of the contigs (unitigs), lengths, coverage, number of reads and other metadata.

Genome coverage

- How much coverage is enough?

```
$ ls -lht *.fasta
```

4.5 Mb - 223Mb fastq reads (25X), 13,124 reads

Filter reads further:

0.75 - 9,843 reads, 168 Mb, ~18X

0.50 - 6,562 reads, 111Mb, ~12.5X

0.25 - 3,281 reads, 56Mb, ~6.3X

Genome coverage

- How much coverage is enough?

  Full data set:

  4.5 Mb - 223Mb fastq reads (25X), 13,124 reads

Percent Reads	Coverage	Contigs	Unitigs	Unassembled Reads	Genome Size(Mb)
1.0	25X	1	1	1,014	4.5
0.75	18X	1	3	714	4.5
0.5	12.5X	31	35	735	4.3
0.25	6.3X	67	75	1,188	1.9

https://rtsf.natsci.msu.edu/_rtsf/assets/File/depth%20and%20coverage.pdf

Genome coverage

SEQUENCE

- **ecoli*.contigs.fasta** - Everything which could be assembled and is part of the primary assembly, including both unique and repetitive elements.
- **ecoli*.unitigs.fasta** - Contigs, split at alternate paths in the graph
- **ecoli*.unassembled.fasta** - Reads and low-coverage contigs which could not be incorporated into the primary assembly.

The header line for each sequence provides some metadata on the sequence:

tig # len=<integer> reads=<integer> covStat=<float> gappedBases=<yes|no> class=<contig|bubble|unassm> suggestRepeat=<yes|no> suggestCircular=<yes|no>

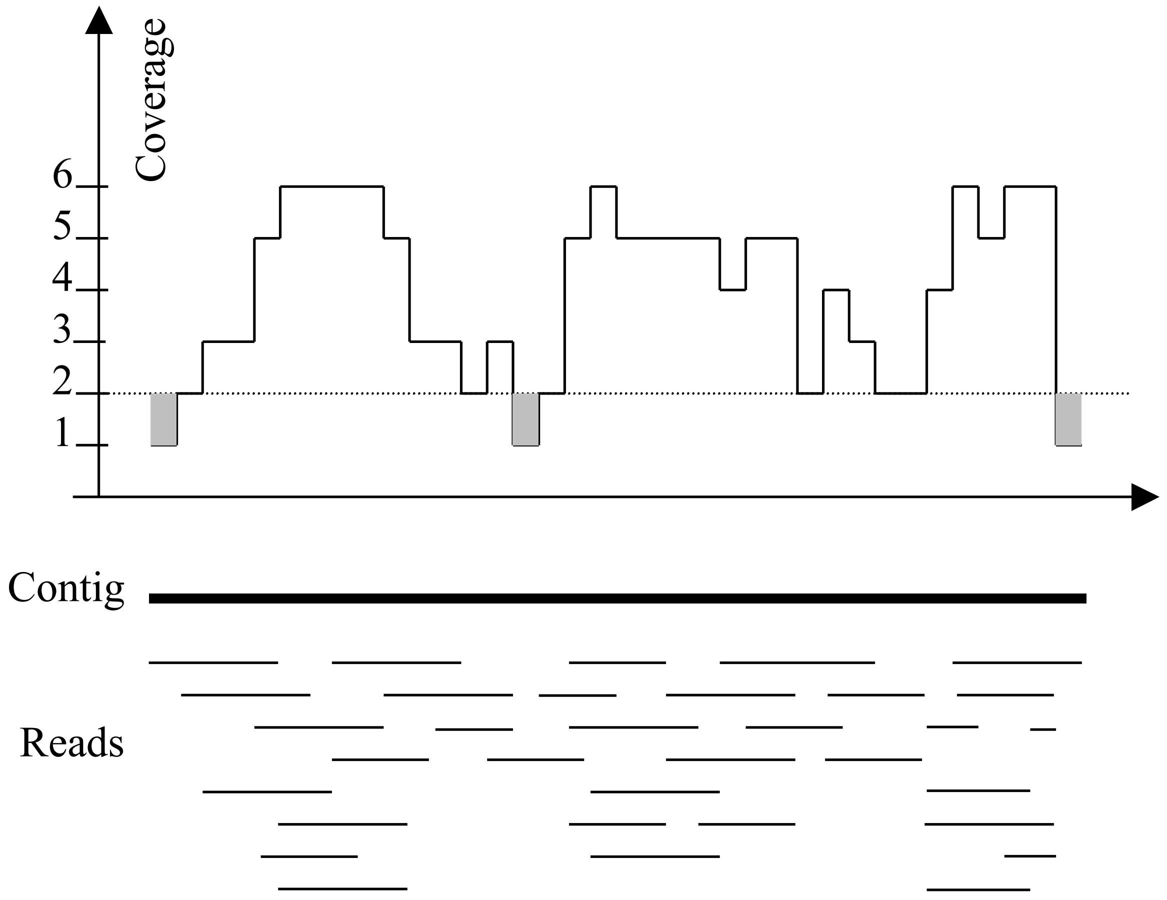
ecoli-0.25.contigs.fasta:

>tig00000001 len=102939 reads=69 covStat=1.36 gappedBases=no class=contig suggestRepeat=no suggestCircular=no

ecoli-full.contigs.fasta:

>tig00000001 len=4639282 reads=10263 covStat=3780.63 gappedBases=no class=contig suggestRepeat=no suggestCircular=no

Typical sequencing coverage



Imagine raindrops on a sidewalk

*Slide from Michael Schatz

Poisson Distribution

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

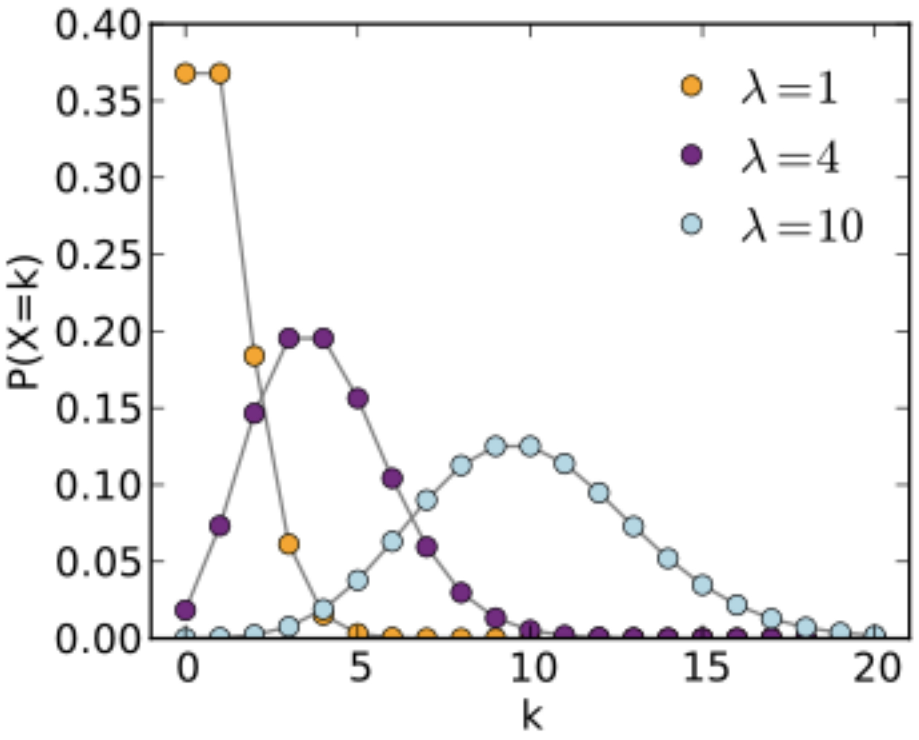
The probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event.

Formulation comes from the limit of the binomial equation

Resembles a normal distribution, but over the positive values, and with only a single parameter.

Key property:

- ***The standard deviation is the square root of the mean.***



*Slide from Michael Schatz

Genome Assembly Problem Set

Using ecoli-0.25.contigs.fasta, write a script that reports:

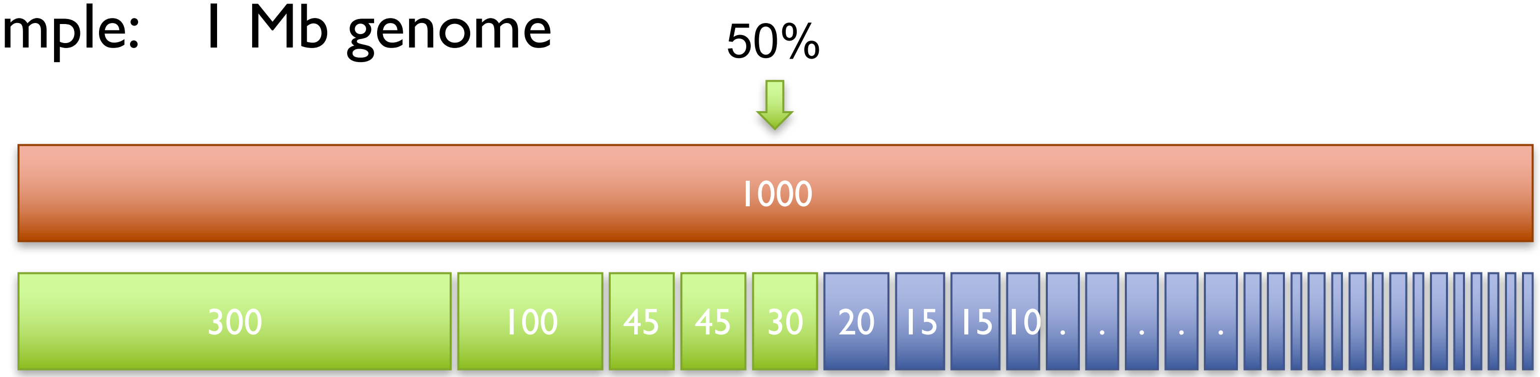
1. The number of contigs in the file
2. The shortest contig.
3. The longest contig.
4. The L50 size

The fasta.file is listed on course github as ecoli_6x.fasta.

L50 size

Def: 50% of the genome is in contigs as long as the L50 value

Example: 1 Mb genome



L50 size = 30 kbp

(300k+100k+45k+45k+30k = 520k >= 500kbp)

A greater L50 is indicative of improvement in every dimension:

- Better resolution of genes and flanking regulatory regions
- Better resolution of transposons and other complex sequences
- Better resolution of chromosome organization
- Better sequence for all downstream analysis

References

- Illumina data:

w2rap-contigger

https://bioinfologics.github.io/the-w2rap-contigger

Allpaths-lg

http://www.broadinstitute.org/software/allpaths-lg/blog/

- PacBio data:

Falcon & Falcon-unzip

https://github.com/PacificBiosciences/FALCON

Canu

http://canu.readthedocs.io/en/latest/index.html