Genome Assembly Tutorial

Deb Triant

21 October 2019

Programming for Biology

Cold Spring Harbor, NY

Genome Assembly Workshop

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
- Well maintained and documented

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

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.

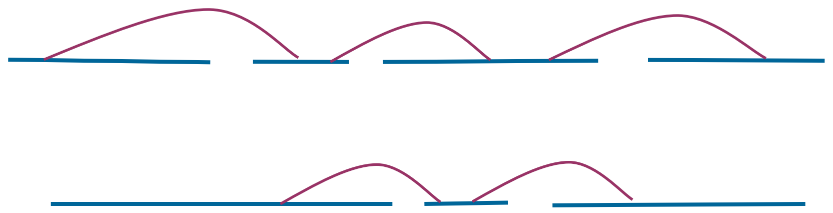
Canu

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

MaSuRCA

https://academic.oup.com/bioinformatics/article/29/21/2669/195975

Linear assemblies



contigs in scaffolds

contig: a contiguous sequence of bases….

```
CTGCCCCCTGTGCCAATGGGTTTGAGGCTCTTCCCACTTCCTTTTCTATTAGATTCAATGTATCTGGTTTTATGTTGAGG
TCCTAGATCCACTTGGACTTGAGCTTTGTACAAGATGACATATATAGGTCTGTTTTTATTCTTCTACATACAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
```

scaffold: a sequence of contigs, separated by gaps….

```
TCCTAGATCCACTTGGACTTGAGCTTTGTACAAGATGACATATATAGGTCTGTTTTTATTCTTCTACATACAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
NNNNNNNNAGTTTTTACCACAATTGCTCTATAGTAAAGCTTGAGGTCAGGGTTGGTGATCCCTCCAGCCATTCTTTCATT
ATTAAGAATTGTTTTCCCTAGTCTGGGTTTTTTGCTTTTCCAGGCGAATTTGAGAATTGCTCTTTCCATGTCTTTGAAGA
ATTGTGTTNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNGGGATTTTGATGGGGTTTGCATTGAATCTGTAGATT
GTCTTTGGTAAGATGGTTAGTTTTACTATGTTAATTCTGCCAATCCACAAGCATGGGAGCGCTCTCCATTTTCTGAGATC
TTCTTCAATTTCTTTCTTGAGAAACTTGAAGTTATTGTCATACA
```

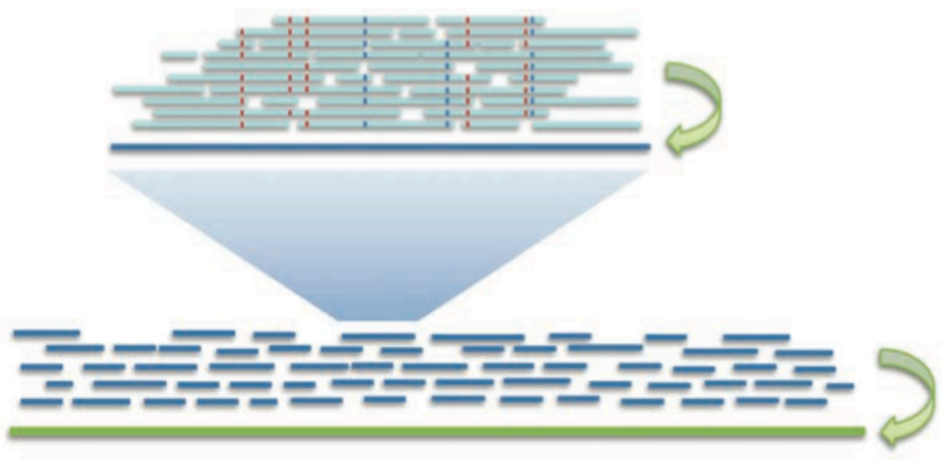
Number of Ns = predicted gap size

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

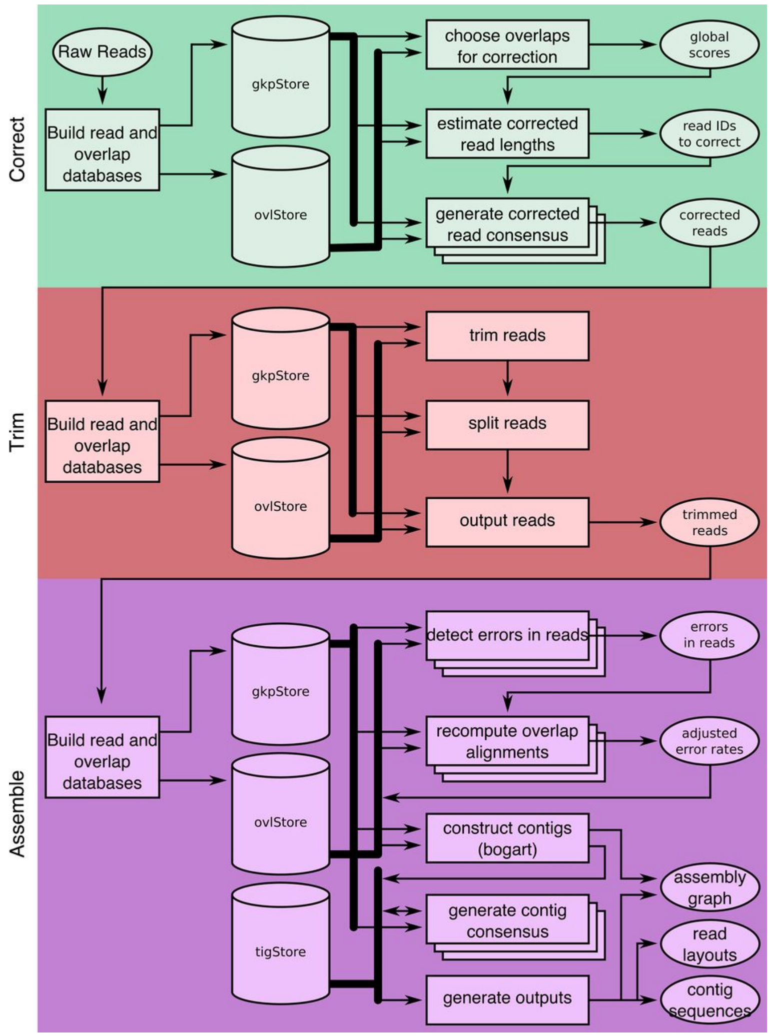
- Canu is an assembler that specializes in PacBio or Oxford Nanopore long-read data
- Derived from Celera assembler
- Corrects reads, trims suspect regions (e.g., adaptors) then assembles the corrected reads
- Can be run with one command to do all steps or each step can be run separately (correcting, 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



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

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.

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"

For eukaryotic genomes, coverage more than 20x sufficient but 30-60x is recommended minimum.

More coverage, more longer reads for Canu to assemble resulting in better assemblies.

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.

Found on canu quickstart:

https://canu.readthedocs.io/en/latest/quick-start.html#quickstart

"no experience or data required, download and assemble *Escherichia coli* today"!

Installing Canu

To install Canu:

1. git option - not using today!

   - compiler conflict

Make a directory for installation

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

.... Mac compiler error!!

-j = number of threads

Installing Canu

To install Canu:

2. Download current release - Canu v1.8

- released 22 October 2018

https://github.com/marbl/canu/releases

- click on green "Latest release" tab

```
$ wget
https://github.com/marbl/canu/releases/downlo
ad/v1.8/canu-1.8.Darwin-amd64.tar.xz
```

```
$ tar -xJf canu-1.8.Darwin-amd64.tar.xz
```

```
    -x extract
    -J specific to .xz files
    -f file
```

Installing Canu

Where is it installed?

Executables found in:

canu-1.8/

You can run the assembler with:

canu-1.8/Darwin-amd64/bin/canu

from within bin: ./canu

Installing Canu

- Executables found in: canu-1.8/Darwin-amd64/bin/canu
  - 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

Need gnuplot for plotting: Thanks Jessen!

```
$ which gnuplot
$ brew install gnuplot
```

using homebrew - package manager for macOS:

https://brew.sh

Installing Canu

- If we try to run will receive a java error. Again, specific to the course computers:

```
$ brew cask install java
```
– from previous workshop

```
error:ERROR:  mhap overlapper requires java version at least 1.8.0; you have unknown (from 'java').
ERROR:  'java -Xmx1g -showversion' reports:
ERROR:    'No Java runtime present, requesting install.'
```

```
$ java --version
```

- click on "more info" pop-up button to bring you to Java SE Downloads page

Installing Java:

https://www.oracle.com/technetwork/java/javase/downloads/jdk13-downloads-5672538.html

https://www.oracle.com/technetwork/java/javase/downloads/jdk13-downloads-5672538.html

- `jdk-13.0.1_osx-x64_bin.dmg -double-click`
- `should see JDK 13.0.1.pkg`
- `installation window`

```
$ java --version
java 13.0.1 2019-10-15
Java(TM) SE Runtime Environment (build 13.0.1+9)
Java HotSpot(TM) 64-Bit Server VM (build 13.0.1+9,
mixed mode, sharing)
```

Installing Canu

Getting Help

/canu/documentation/source - help docs

`$ canu` command line options

`$ canu -options` parameter options

https://canu.readthedocs.io/en/latest/

Installing Canu

Download canu test E.coli dataset:

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

- 25X subset (223Mb)

Where do you want them to download?

```
$ curl -L -o pacbio.fastq
http://gembox.cbcb.umd.edu/mhap/raw/ecoli_p6_
25x.filtered.fastq
```

- o naming downloaded file pacbio.fastq

- L if page has moved to a new location, curl will redo at the request at the new site

How many reads within the file? How many lines?

FASTQ

```
@m141013_011508_sherri_c100709962550000001823135904221533_s1_p0/5451
9/26233_32397
TCGATCGAGTAACTCGCTGCTGTCAGACTGGTTTTTGGTCGATCGACTATTGTTTCAGTCGCAAGAAT
ATTGTGTCCAGTCGATCGACTGAATTCTGCTGTACGGCCACGGCGGATGCACGGTACAGCAGGCTCAG
ACGGATTAAACTGTT
+
5=9=9<=9,-5@<<55>,6+8AC>EE.88AE9CDD7>+7.CC9CD+++5@=-FCCA@EF@+**+*--
55--AA---AA-5A<9C+3+<9)4++=E=+===<D94)00=9)))2@624(/(/2/-
(.(6;9(((((.(.'((6-66<6(///
@m141013_011508_sherri_c100709962550000001823135904221533_s1_p0/5453
2/2817_4395
GTAAAATTGAGGTAAATTGTGCGGAATTTAGCAATACCGTTTTTTTATTATCACCGGATATCTATTC
TGCTGTACGGCCAAGGAGGATGTACGGTACAGCAGGTGCGAACTCACTCCGACGCTCAAGTCAGTGAC
TTAATGATAAGCGTG
+
?????<BBBBBB5<?BFFFFFFECHEFFECCFF?9AAC>7@FHHHHHHFG?EAFGF@EEDEHHDGHHC
BDFFGDFHF)<CCD@F,+3=CFBDFHBD++??DBDEEEDE:):CBEEEBCE68>?))5?**0?:AE*A
*0//:/*:*:**.0)
@m141013_011508_sherri_c100709962550000001823135904221533_s1_p0/5455
1/25910_41116
GCTAGTCTTGTGTTTAGTTTTATGTTTTGCATGTTGTAACGGATTCATAAACATAGGTGTTTGTTTCT
TTTTATGGTTGTACAATTTGGCCCTAAGGCCCTACACTTACTTGTTTGTTTCTTTTATGGTACGACAT
TTGAGTGGTGGTTGA
+
```

Running Canu

screen - unix command to run programs in multiple windows or "screens"

- If your computer breaks connection, your program will continue run on server

$ `screen` - to open a new screen

$ `ctrl-a d` to suspend screen

-Program will continue to run

$ `screen -r` - to re-open existing screen session

$ `screen -r ####` - use number to re-open if multiple sessions running

Running Canu

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

```
$ canu -p ecoli -d ecoli-pacbio  genomeSize=4.8m  -pacbio-raw \
your_path/canu/pacbio.fastq saveReads=true \
2>CanuRun_20191021.log **to save output

*where are reads....include pathway
-p assembly-prefix
-d output directory
input types:
 -pacbio-raw -pacbio-corrected -nanopore-raw -nanopore-corrected fastq or
fasta format
```

****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-full  genomeSize=4.8m  -pacbio-raw \
your_path/canu/pacbio.fastq saveReads=true \
2>full_out_20191021.log **to save output
```

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 2>correct_out_20191021.log
```

Then trim the output of the correction:

```
canu -trim -p ecoli -d ecoli-manual  genomeSize=4.8m \
-pacbio-corrected ecoli-manual/ecoli.correctedReads.fasta.gz \
saveReads=true 2>trim_out_20191021.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   \ 2>assemble-
0.013_out_20191021.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_20191021.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.

Running Canu

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

```
$ canu -p ecoli -d ecoli-pacbio  genomeSize=4.8m  -pacbio-raw \
your_path/canu/pacbio.fastq saveReads=true \
2>CanuRun_20191021.log **to save output

*where are reads....include pathway
-p assembly-prefix
-d output directory
input types:
 -pacbio-raw -pacbio-corrected -nanopore-raw -nanopore-corrected fastq or
fasta format
```

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

Running Canu - job control

`Ctrl – z` - stops job running at command line

`bg` - moves to background

`jobs` - see what is running

`fg` -job number - move job to foreground

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:

>tig00000001 len=576432 reads=573 covStat=249.02 gappedBases=no class=contig suggestRepeat=no suggestCircular=no

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

*covStat: Is the log of the ratio of the contig being unique vs it being two-copy, based on the number of reads Positive means more likely to be unique, negative means more likely to be repetitive

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?

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

Try some assemblies with filtered data sets.

Using canu - split reads

- Count and split files:

```
$ wc -l pacbio.fastq
$ split -l
```
output number of lines - full: 52,496

- 60%   31,498

```
$ split -l
```
31498 pacbio.fasta

```
$ mv xaa ecoli_filtered_0.60.fastq
$ rm
```
the rest (xa*)

- confirm number of lines and reads

Genome coverage

- How much coverage is enough?

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

Try some assemblies with filtered data sets:

0.80 - reads, 10,499 reads, 180M, ~22X

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

0.60 - 7,874 reads, 133Mb, ~15X

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

0.40 - 5250 reads, 89Mb, ~10X

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

Genome coverage

- How much coverage is enough?

  Full data set:

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

Percent Reads	Coverage	Contigs	Unitigs	Unassembled Sequences	Unassembled Length (Mb)	Genome Size (Mb)
1	25X	1	1	1506	7.1	4.6
0.8	22X	1	1	1234	6.4	4.6
0.75	18X	1	1	1109	5.8	4.6
0.6	15X	8	8	959	5.5	4.6
0.5	12.5X					
0.25	6.3X					

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

Genome coverage

- How much coverage is enough?

  Full data set:

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

Percent Reads	Coverage	Contigs	Unitigs	Unassembled Sequences	Unassembled Length (Mb)	Genome Size (Mb)
1	25X	1	1	1,506	7.1	4.6
0.8	22X	1	1	1,234	6.4	4.6
0.75	18X	1	1	1,109	5.8	4.6
0.6	15X	8	8	959	5.5	4.6
0.5	12.5X	31	35	735	4.6	4.3
0.25	6.3X	67	75	1,188	7.5	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>

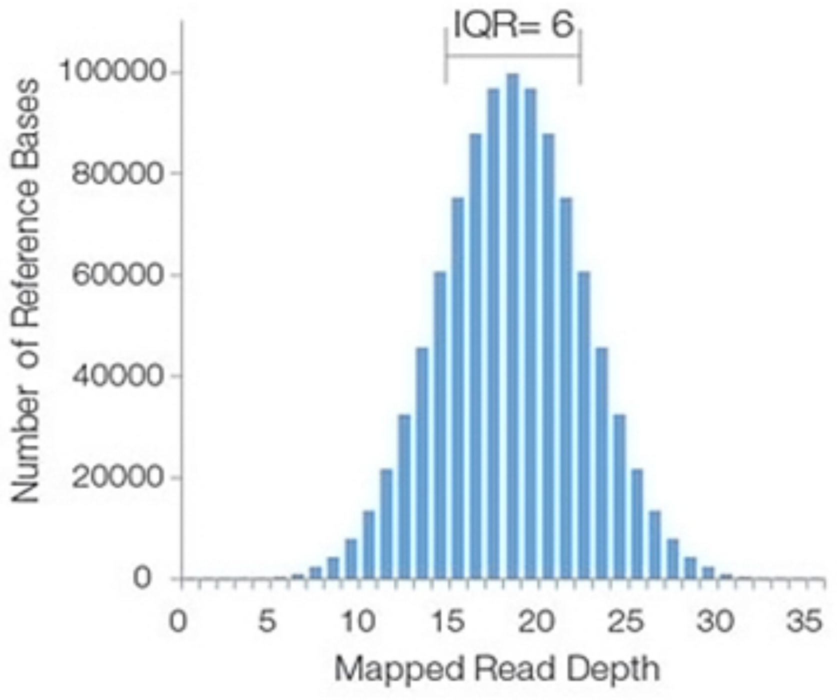
ecoli-0.25.contigs.fasta:

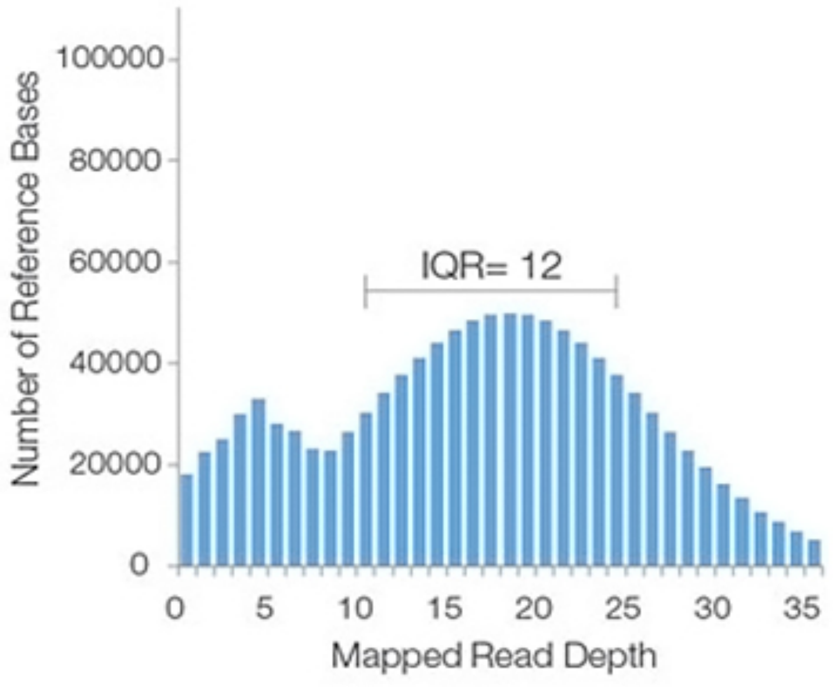
>tig00000001 len=102939 reads=69 covStat=1.36 gappedBases=no class=contig suggestRepeat=no suggestCircular=no

ecoli-full.contigs.fasta:

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

Coverage histograms





Assumes reads randomly distributed across the genome

https://www.illumina.com/science/education/sequencing-coverage.html

Poisson Distribution

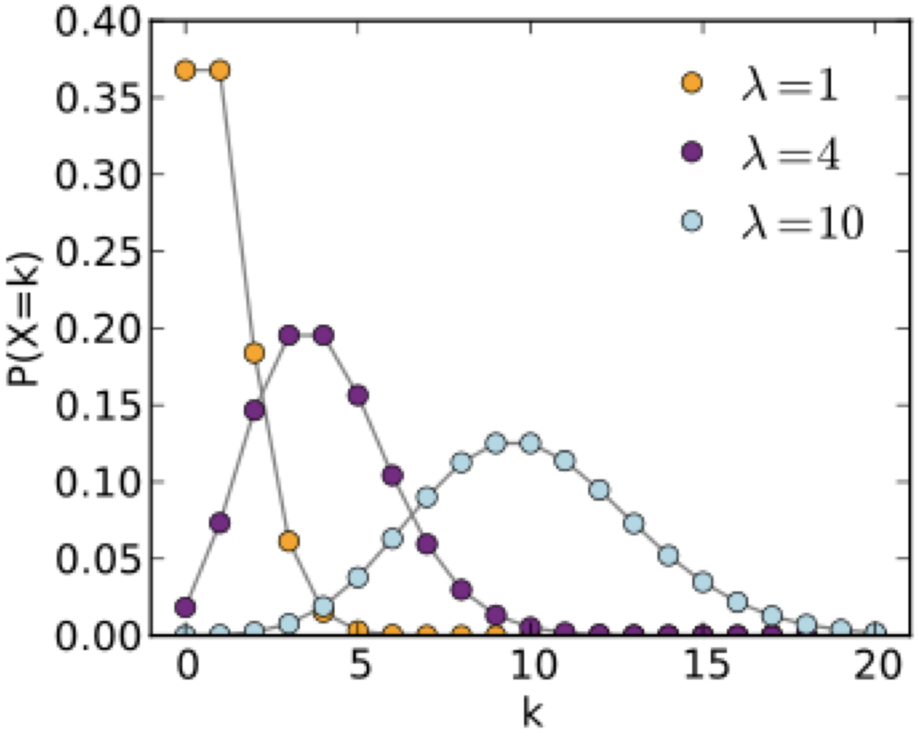
The probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event.

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

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 length
5. The N50 count

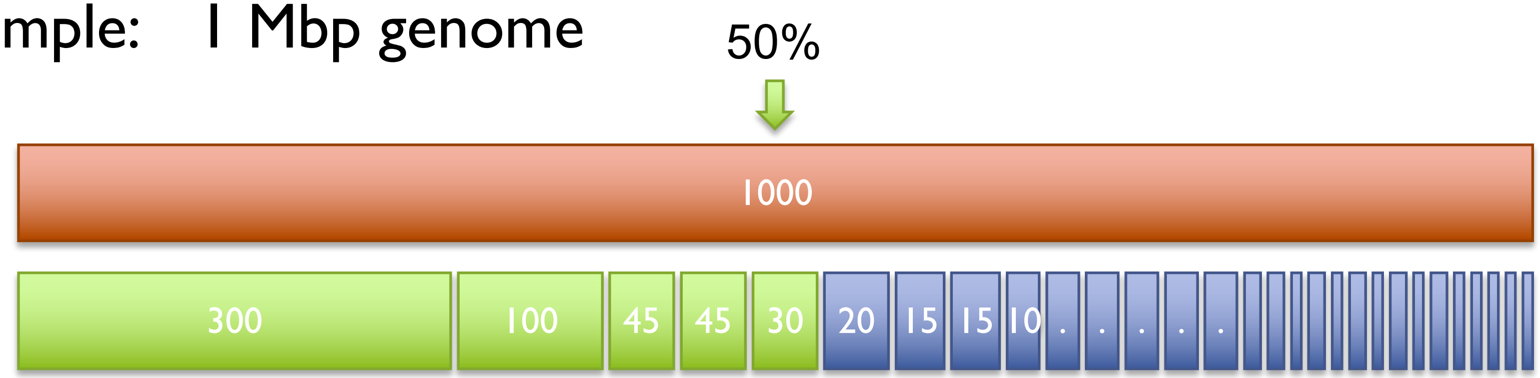
The fasta.file is listed on github workshops/GenomeAssembly:

ecoliPB-filtered-0.25.contigs.fasta.

N50

Number of contigs that are as long or longer than the N50 value

Example: 1 Mbp genome



L50 = 30 kbp

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

N50 - number of contigs that sum to 50% length of assembly

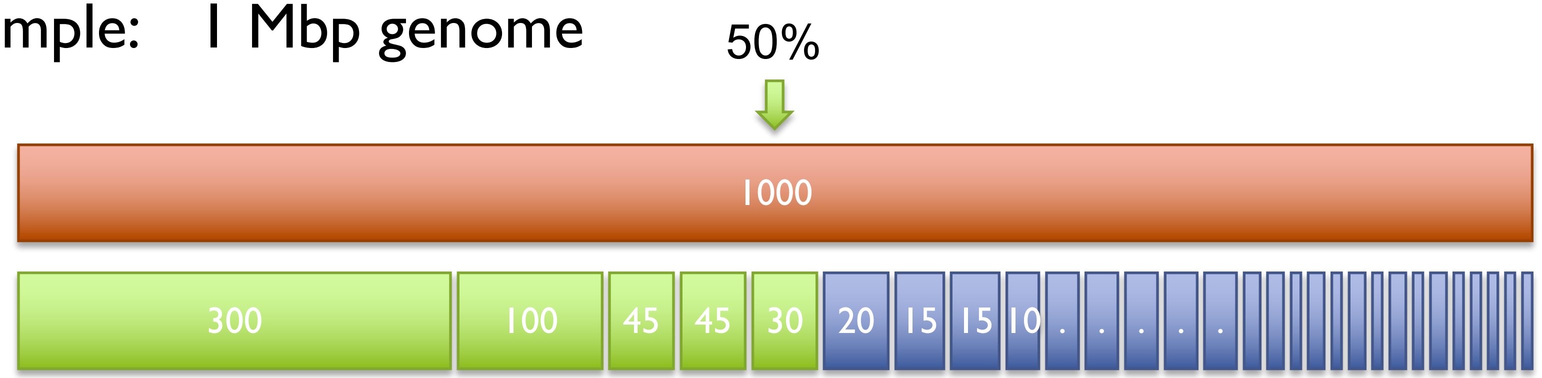
N50 = how many?

- High L50 vs Low N50
  - longer sequences and fewer of them….in theory
  - lower stringency can inflate L50

L50

50% of the genome in contigs as long as or larger than contig at half genome size

Example: 1 Mbp genome



L50 = 30 kbp

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

A greater L50 is usually a sign of assembly improvement

- Comparable with genomes of similar size
- Genome composition can bias comparisons
- High L50 vs Low N50