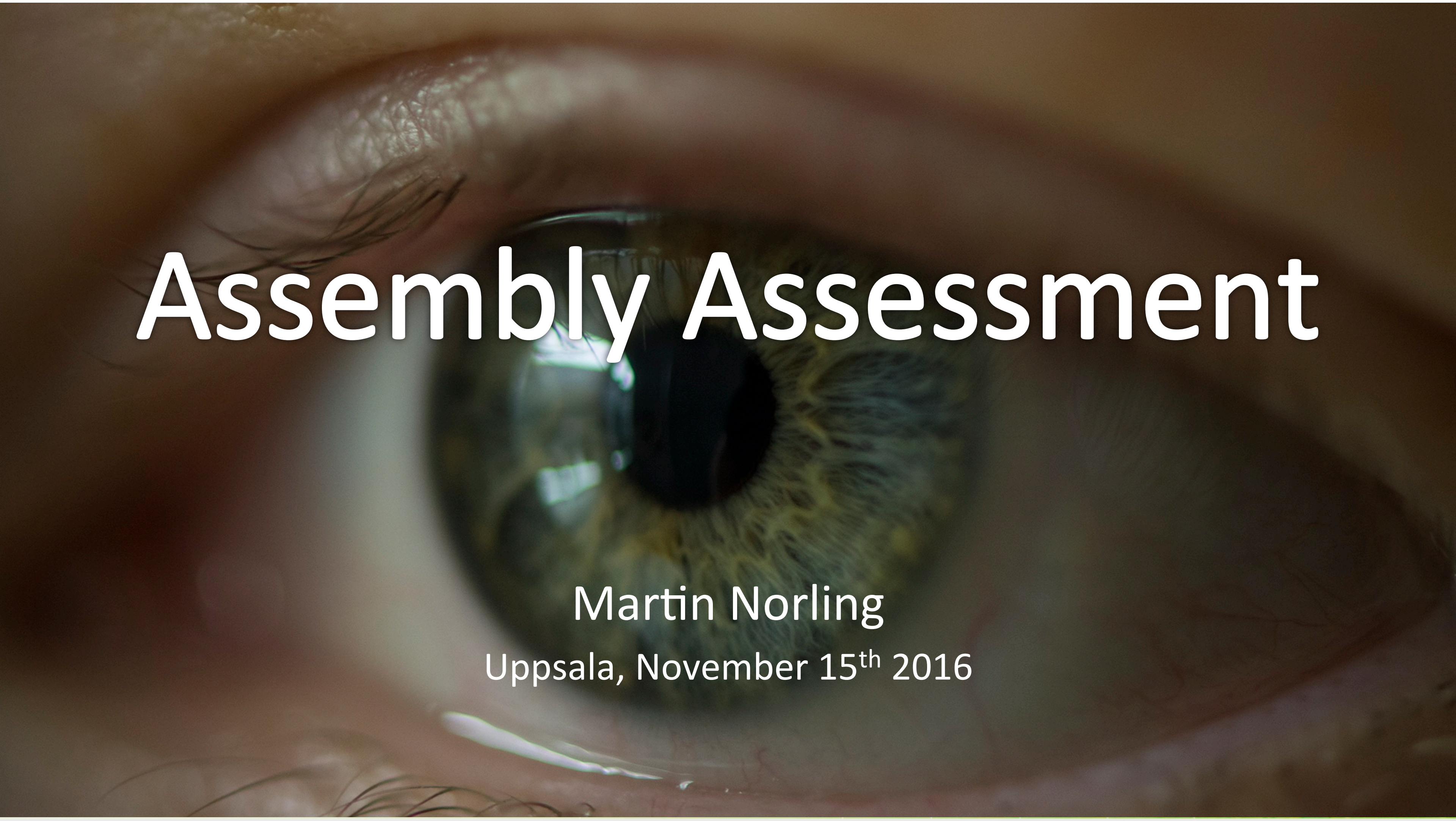


Assembly Assessment

Martin Norling

Uppsala, November 15th 2016

What can we do with an assembly?

Since we can never know the actual sequence, or its variations, validating an assembly is tricky.

But once you’ve used all the assemblers, which assembly should you choose?
Should you trust it?

Is it good enough to start annotating?

Reads

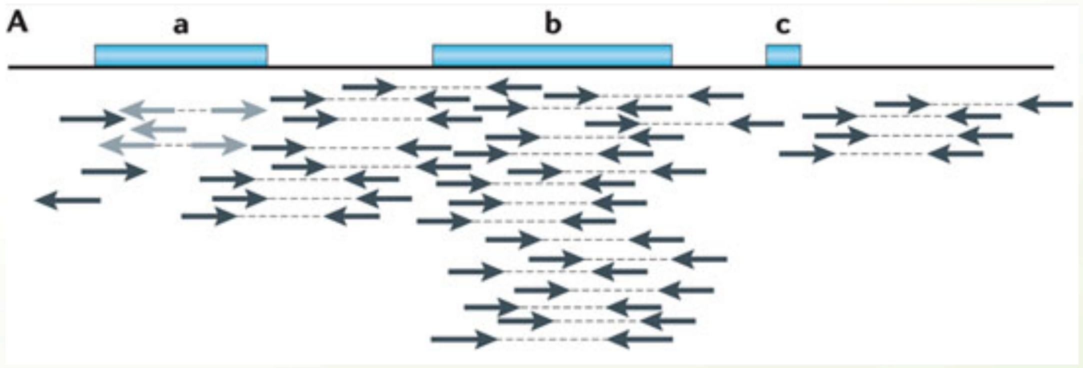
As we discussed earlier, the connection between assembly and reads are commonly lost, as most assemblers are (at least in part) de bruijn graph based.

How well the reads match the assembly is *crucial* for the assembly's reliability though.

Data congruency

Read-pairs in particular are useful when mapped back to an assembly. We can look for things like:

- no read coverage
- paired reads in different contigs
- too long/short pair distances
- reads in wrong direction



How do we map the reads back?

- Many tools available, we commonly use *BWA*, **B**urrows **W**heeler **A**ligner, or *bowtie* (which is also based on the Burrows Wheeler transform).
- Read mapping is a very simple problem compared to de novo assembly, but can still be confused by troublesome genomic regions.

BWA

The *Burrows-Wheeler Transform* is originally a data compression algorithm that reversibly sorts a string of characters into runs of similar characters. This can be used to create a very efficient index of the target sequence.

In short – read mapping becomes a quite efficient operation that is generally always worthwhile.

The result files can be a problem though…

SAMtools and the SAM/BAM/CRAM format(s)

One of the few formats that bioinformatics have (more or less) a standard format for is the SAM format for read mappings.

The SAM format itself is a plain-text format of read-coordinates.

SAM files can be converted to binary BAM files which are more compact, or CRAM files which are compressed even further.

SAM format

SAM format is “readable” in that it looks like this:

CIGAR string

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

There is a lot of good information in there though!

Tags

Feature Response Curves (FRCs)

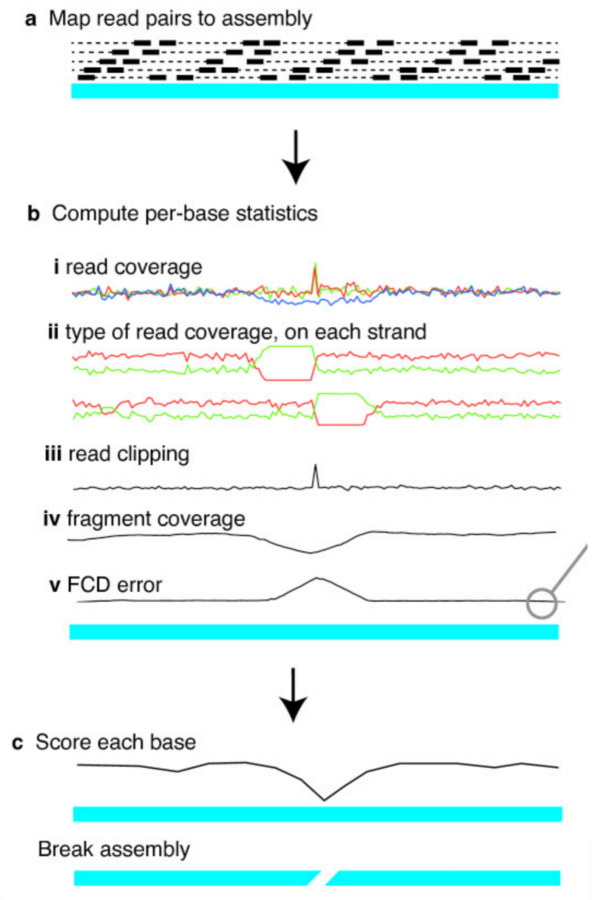
Looks for regions that has suspicious statistics, “features”.

A perfect assembly would have zero features.

- Low/High coverage
- Low/High paired coverage
- High singleton count
- High span (pair on other contig)
- High outie
- Compression/Stretch

https://github.com/vezzi/FRC_align

REAPR


a Map read pairs to assembly
b Compute per-base statistics
i read coverage
ii type of read coverage, on each strand
iii read clipping
iv fragment coverage
v FCD error
c Score each base
Break assembly

Uses same principle of FRCurve:

- Identifies suspicious/erroneous positions
- **Breaks assemblies in suspicious positions**
- The “broken assembly” is more fragmented but hopefully more corrected (REAPR cannot make things worse…)

REAPR (Hunt et al. 2013)

How was the sequence produced?

Many library preparation techniques will affect the sequencing output. As shown by this figure from a virology paper, whole genome amplification (WGA) can severely alter the coverage profile.

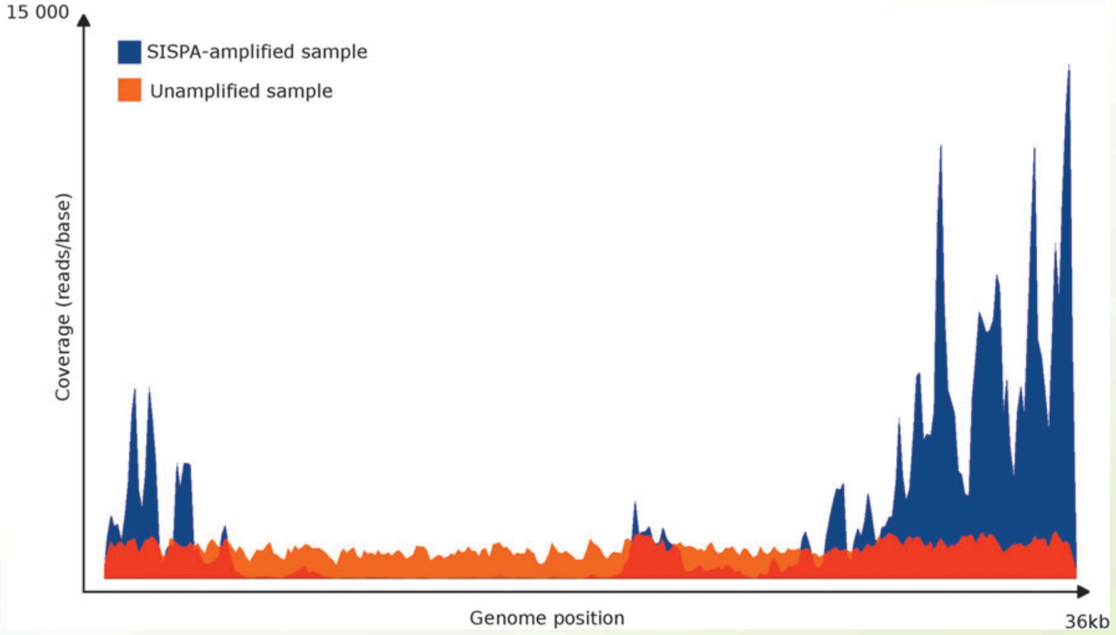
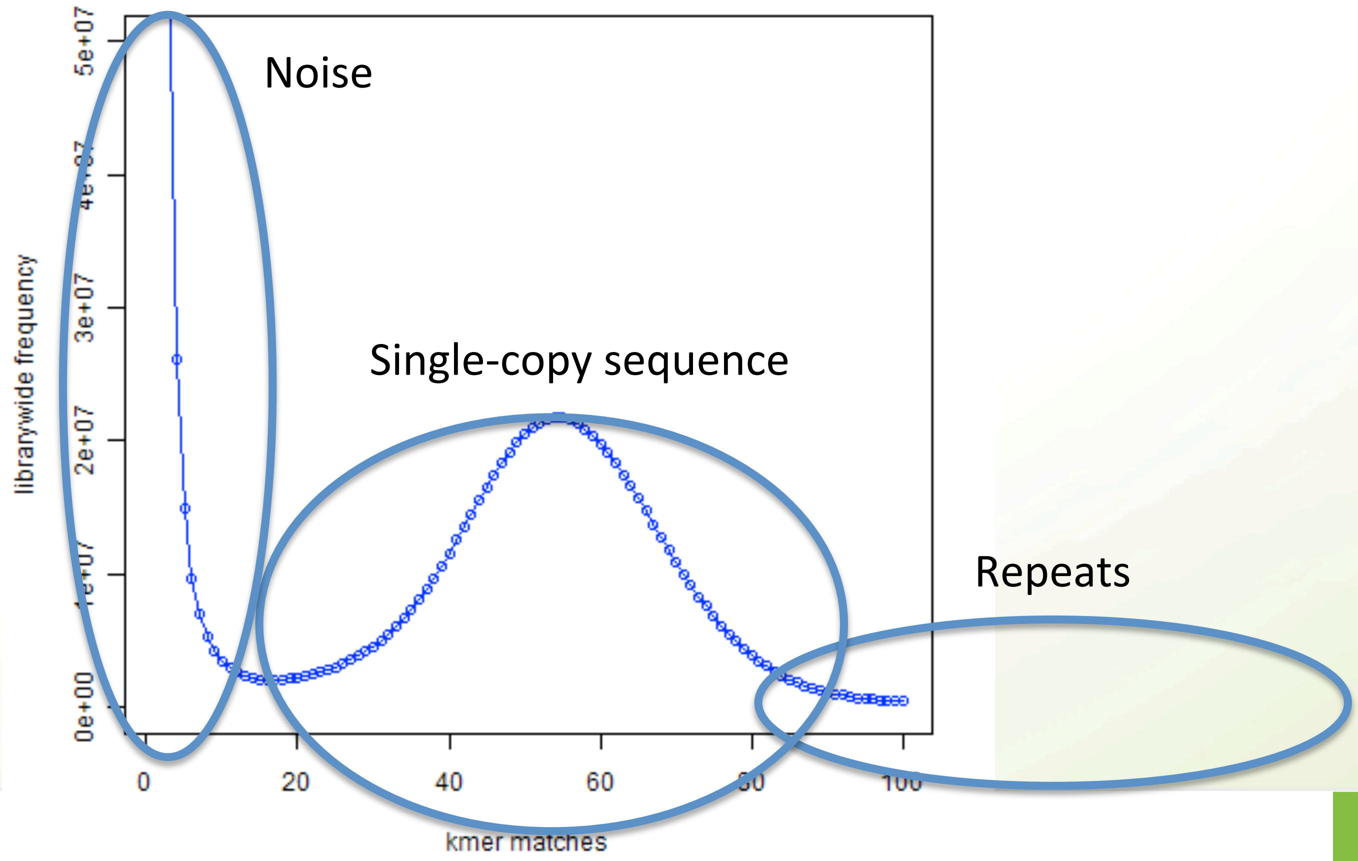


Figure 1. Comparison of coverage by mapping the reads to the Ad2 reference genome using Bamview. Amplified and unamplified samples are mapped together overlapping. Shown is the discrepancy between the amplified approach and the unamplified approach.

Kmer content

As you remember from earlier, the kmer-spectra tells us what information is in the READS.



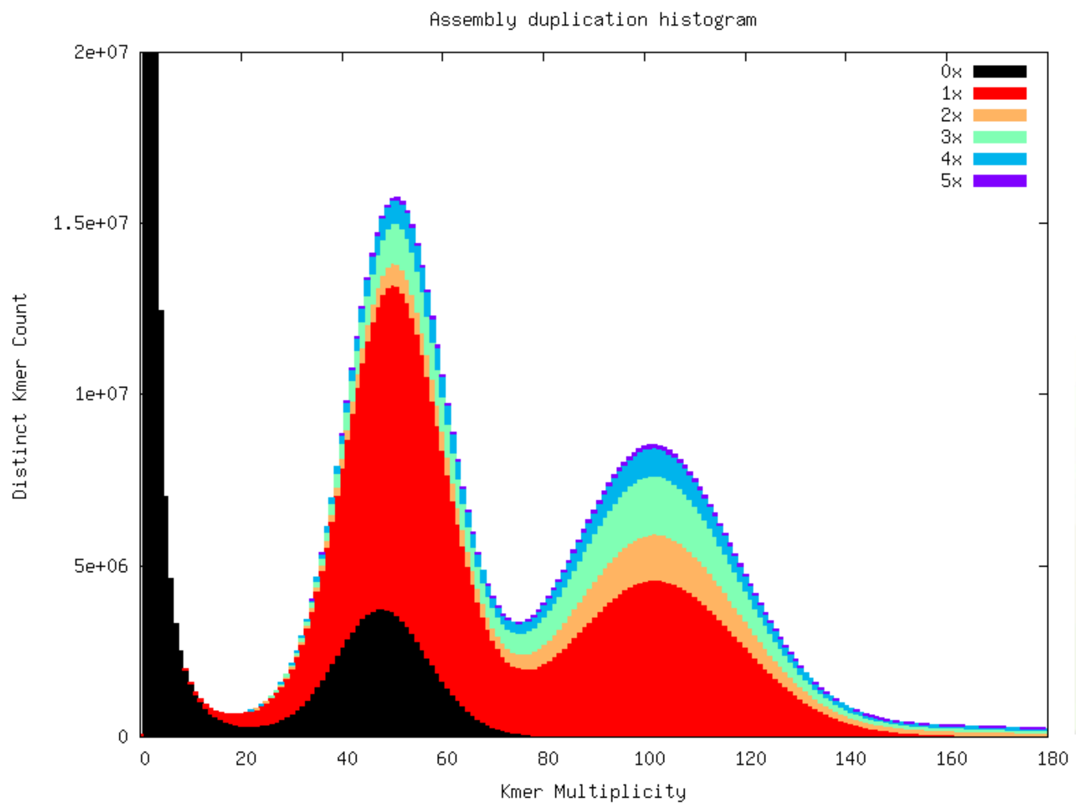
KAT

Going back to KAT (<u>K</u>mer <u>A</u>nalysis <u>T</u>oolkit), we can extract the kmer-content of the *assembly* as well as the *reads*.

With this information we can compare if the kmer information in the reads correspond well to that of the assembly.

Comparing kmer content of reads vs. assembly

In this graph we see a *diploid* assembly (thus the two peaks), colored by kmer content.

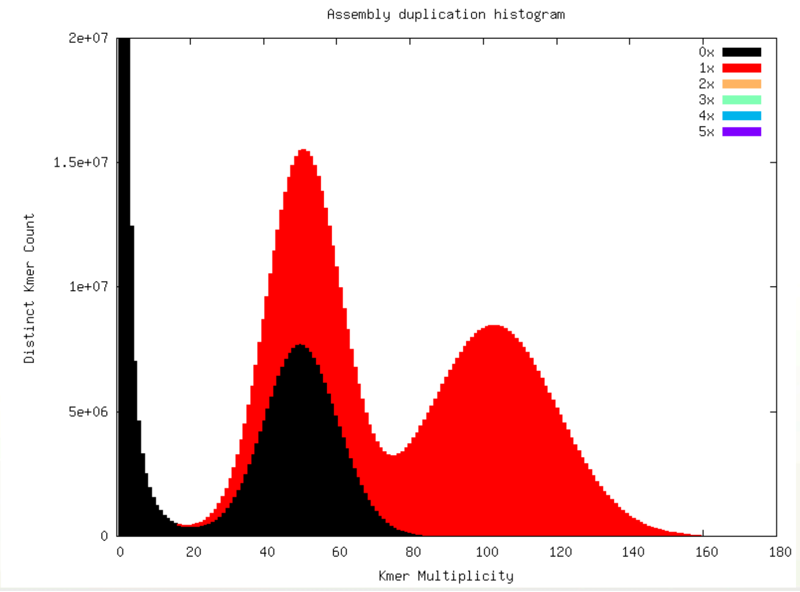


This plot shows several kmers being used multiple times more in the assembly than in the read set.

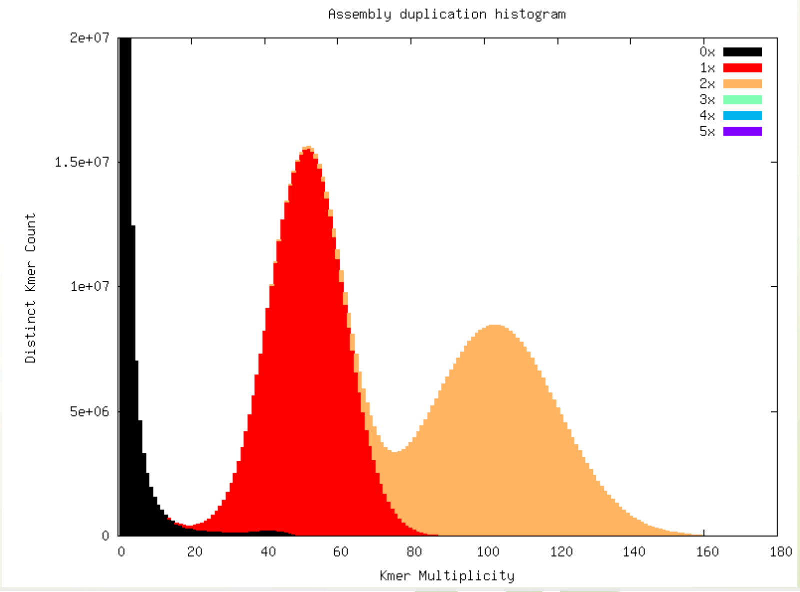
Comparing kmer content of reads vs. assembly

There are two ideal kmer contents for this assembly; a) shows the ideal kmer content from a *haploid* assembler, and b) show the ideal kmer content from a *diploid* assembler.

a)



b)



Questions?

Also, there is coffee before the exercise!