Next Generation Sequencing Workshop
– De novo genome assembly –

Tristan Lefébure

TNL7@cornell.edu

Stanhope Lab
Population Medicine & Diagnostic Sciences
Cornell University

April 14th 2010

De novo assembly methods and concepts

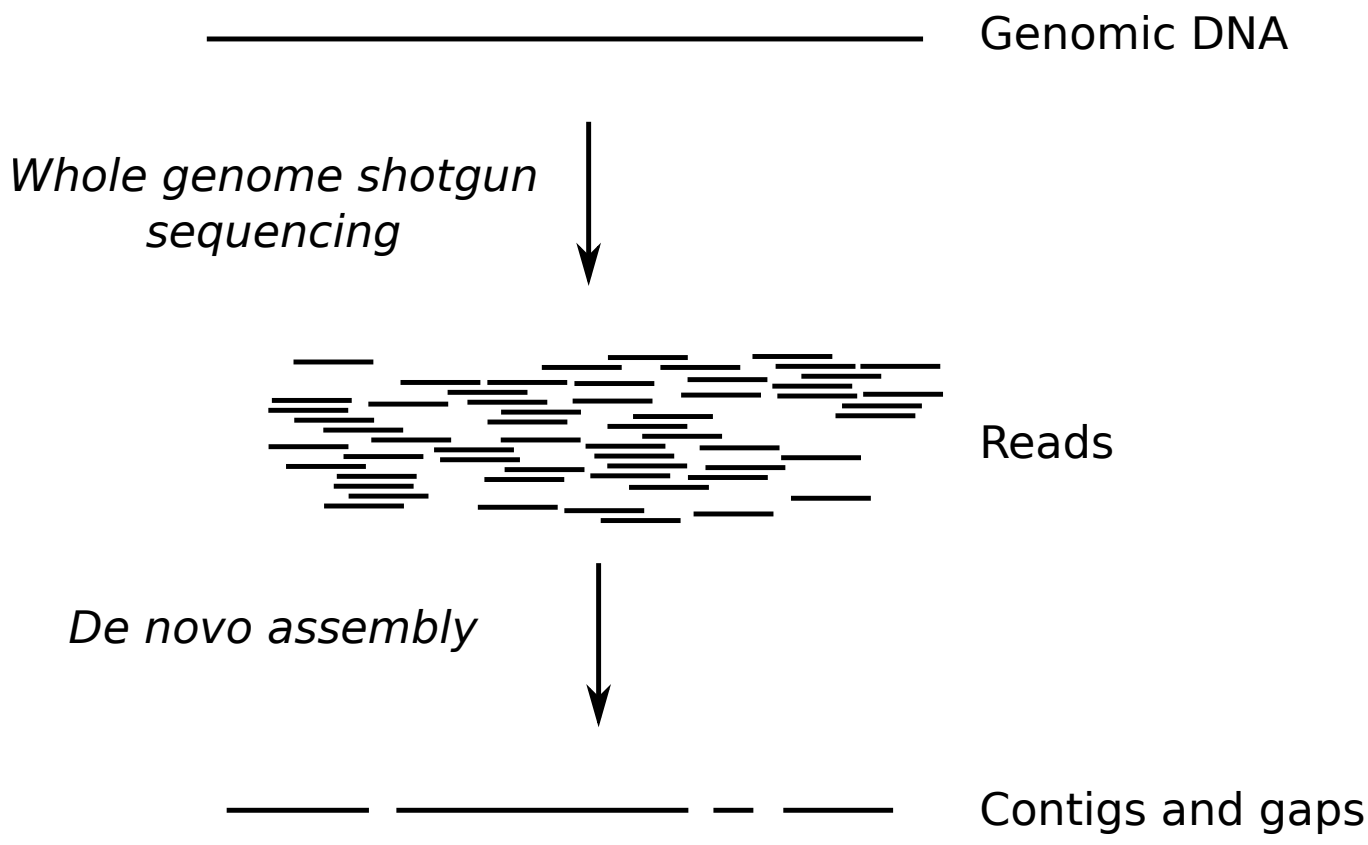
Assembling short reads

454 assembly with Newbler

Illumina assembly with Velvet

Documentation

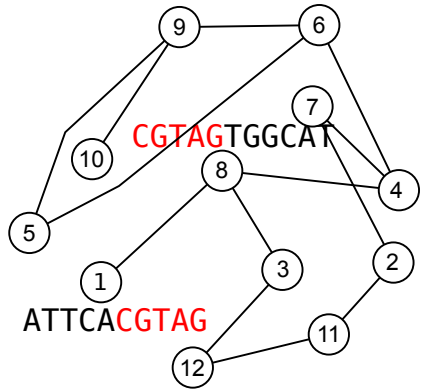
de novo?


Genomic DNA
Whole genome shotgun sequencing
Reads
De novo assembly
Contigs and gaps

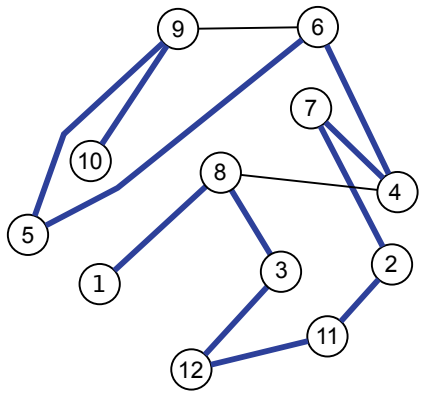
The Overlap-layout-consensus (OLC) approach

1. Pairwise alignments and overlap graph
2. Graph Layout: search of a single path in the graph (i.e. the Hamiltonian path)
3. Multiple sequence alignments and consensus

Examples: Newbler, Celera, Arachne...



Overlap graph



Graph layout


Consensus

The Eulerian path/de Bruijn graph approach

1. kmer hash table
2. de Bruijn graph
3. simplification of the graph and Eulerian path search

Examples: Euler, Velvet, Allpath, Abyss, SOAPdenovo...

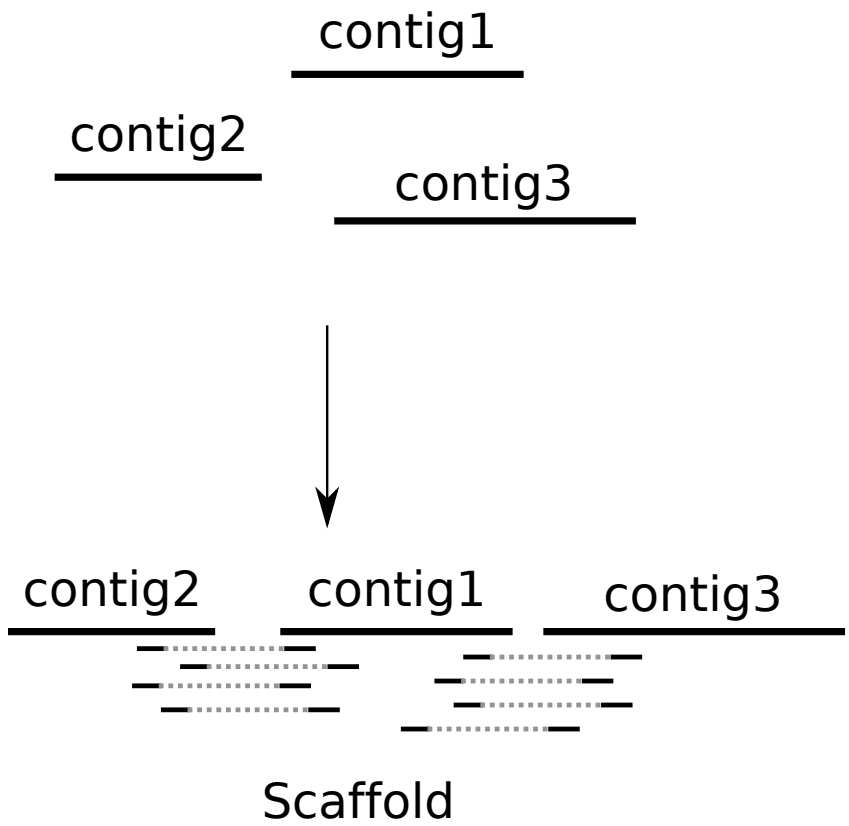
10bp read: ATTCGACTCC

for k=5, 6 kmers:

```
ATTCG
 TTCGA
  TCGAC
   CGACT
    GACTC
     ACTCC
```

de Bruijn Graph

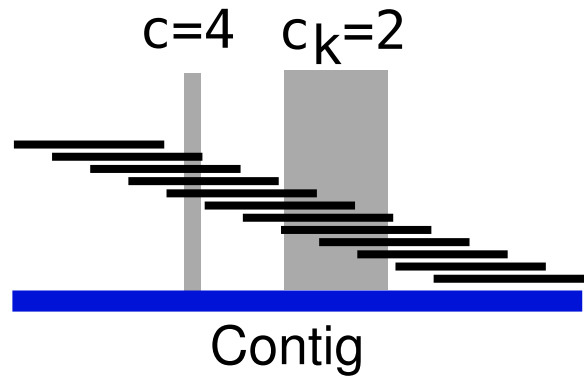
Scaffolding with paired-end/mate reads

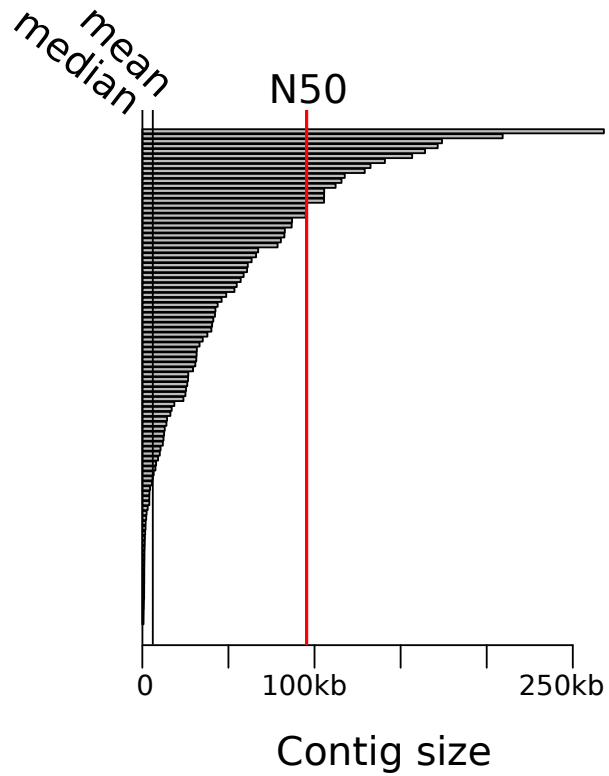


Organization of the contigs into scaffolds

Some vocabulary

- Coverage (or redundancy):
  $$c = \frac{L \times N}{G}$$
  - L: read length
  - N: number of reads
  - G: genome size
- k-mer coverage (c_k)
- N50: weighted median such that 50% of the entire assembly is contained in contigs equal to or larger than this value



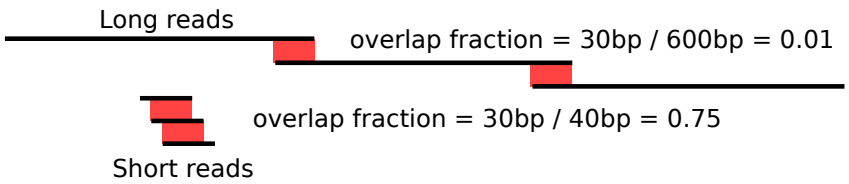


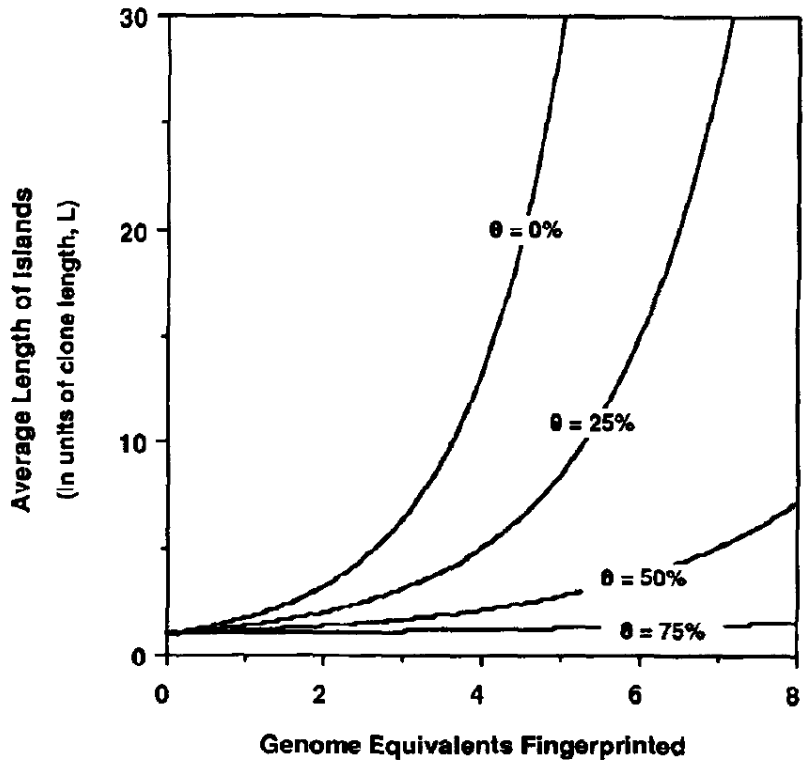
NGS de novo assembly

Problem: Shorter reads and higher error rate

- $\Rightarrow$ Fraction of overlap between reads is high ($\Theta = \frac{25bp}{36bp} \sim 0.7$) $\Rightarrow$ Need higher coverage
- $\Rightarrow$ Discriminate sequencing errors $\Rightarrow$ Need higher coverage
- $\Rightarrow$ Difficulty to resolve small repeats

Consequence: Larger data-set, more heuristics and specific hardware (lots of memory)





(Lander & Waterman 1988)

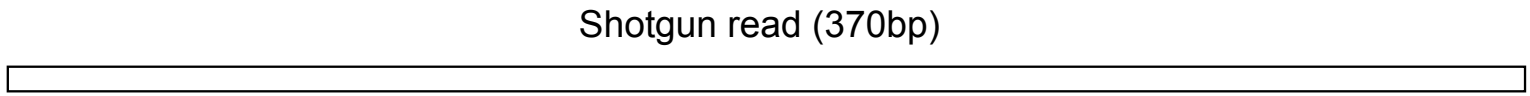
Main short reads de novo assemblers

Data	Program	Link
454	Newbler	distributed with instrument
454	CABOG	`http://wgs-assembler.sf.net/`
Illumina	Velvet	`http://www.ebi.ac.uk/~zerbino/velvet/`
Illumina	ALLPATH	http://www.broadinstitute.org/[...]
Illumina	ABySS	`http://www.bcgsc.ca/platform/bioinfo/software/abyss`
Illumina	SOAPdenovo	`http://soap.genomics.org.cn/`
Mixed	Mira	`http://www.chevreux.org/projects_mira.html`

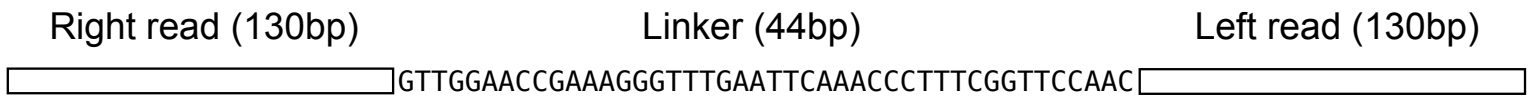
454 data assembly with Newbler

FLX 50/50 mix of regular and paired-end reads of *E. coli* K12:

- 454 “regular“ reads (EcoliRL.sff, 107,769 reads):



- 454 paired-end reads (ecoPEhalfSet8kb.sff, 199,197 reads):



Assembly with Newbler:

1. All the reads are used to build contigs using the OLC
2. Paired-end information is used to build scaffolds

Memory usage (Gb):

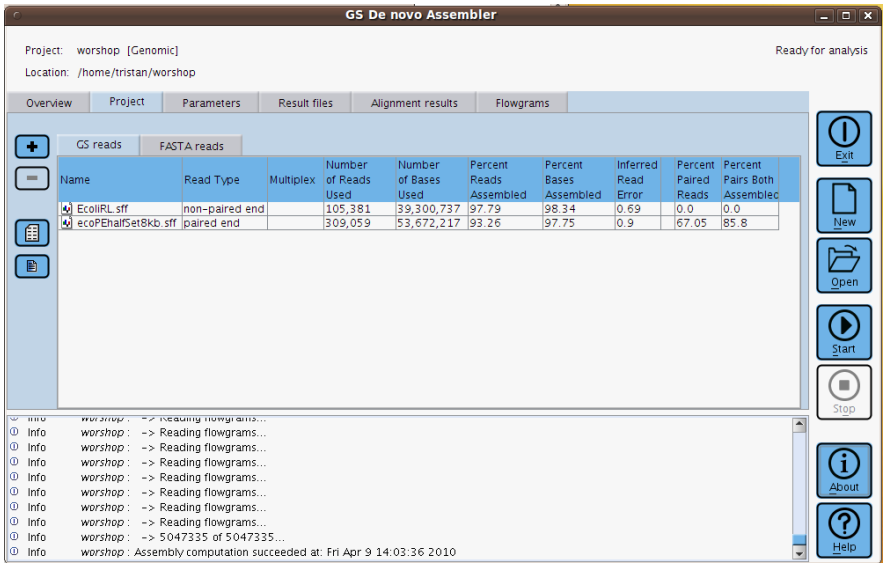
$$
\begin{aligned}
M &= N \times L \times 3/1073741824 \\
&= (107769 \times 370 + 199197 \times 2 \times 130) \times 3/1073741824 \\
&= 0.26GB
\end{aligned}
$$

Running Newbler

The GUI way:

Listing 1: shell script to load Newbler GUI

```
1 gsAssembler
```



The command-line way:

Listing 2: shell script to run Newbler

```
1 #Set up an assembly project
2 newAssembly Ecoli_k12_newbler
3 #Add SFF files to the project
4 addRun Ecoli_k12_newbler EcoliRL.sff
5 addRun Ecoli_k12_newbler ecoPEhalfSet8kb.sff
6 #Run the assembly
7 runProject Ecoli_k12_newbler
```

The outputs: 454NewblerMetrics.txt (1)

Listing 3: snippet of 454NewblerMetrics.txt

```
runData
{
      file
       {
          path = "/assembly/workshop/454/EcoliRL.sff";

          numberOfReads = 107769, 107768;
          numberOfBases = 40000224, 39963011;
       }
}

pairedReadData
{
     file
       {
          path = "/assembly/workshop/454/ecoPEhalfSet8kb.sff";

          numberOfReads = 199197, 331398;
          numberOfBases = 62500134, 54908933;
          numWithPairedRead = 133565;
       }
}
```

Expected Coverage is:

$$C = \frac{39963011 + 54908933}{4639675}$$

$$C \sim 20X$$

The outputs: 454NewblerMetrics.txt (2)

Listing 4: snippet of 454NewblerMetrics.txt

```
pairedReadStatus
    {
        numberWithBothMapped   = 114957;
        numberWithOneUnmapped  = 2121;
        numberMultiplyMapped   = 15962;
        numberWithBothUnmapped = 525;

        library
          {
             libraryName     = "ecoPEhalfSet8kb.sff";
             pairDistanceAvg = 8087.8;
             pairDistanceDev = 2022.0;
          }
     }
```

$\Rightarrow$ Estimated linker size is 8087bp (real is 8kb)

The outputs: 454NewblerMetrics.txt (3)

Listing 5: snippet of 454NewblerMetrics.txt

```
scaffoldMetrics
    {
          numberOfScaffolds    = 6;
          numberOfBases        = 4656961;
          avgScaffoldSize      = 776160;
          N50ScaffoldSize      = 4643557;
          largestScaffoldSize = 4643557;
    }

largeContigMetrics
    {
          numberOfContigs    = 99;
          numberOfBases      = 4554063;
          avgContigSize      = 46000;
          N50ContigSize      = 105518;
          largestContigSize = 267981;
          Q40PlusBases       = 4548397, 99.88%;
          Q39MinusBases      = 5666, 0.12%;
      }

allContigMetrics
      {
           numberOfContigs = 122;
           numberOfBases    = 4558538;
      }
```

Assembly organized in:

- 6 scaffolds, including one giant (4.6MB)
- 99 large contigs (N50 = 105kb)
- Summing 4.55MB (vs 4.64MB): $\sim$ 98% of the genome is assembled

The outputs: 454Scaffolds.fna

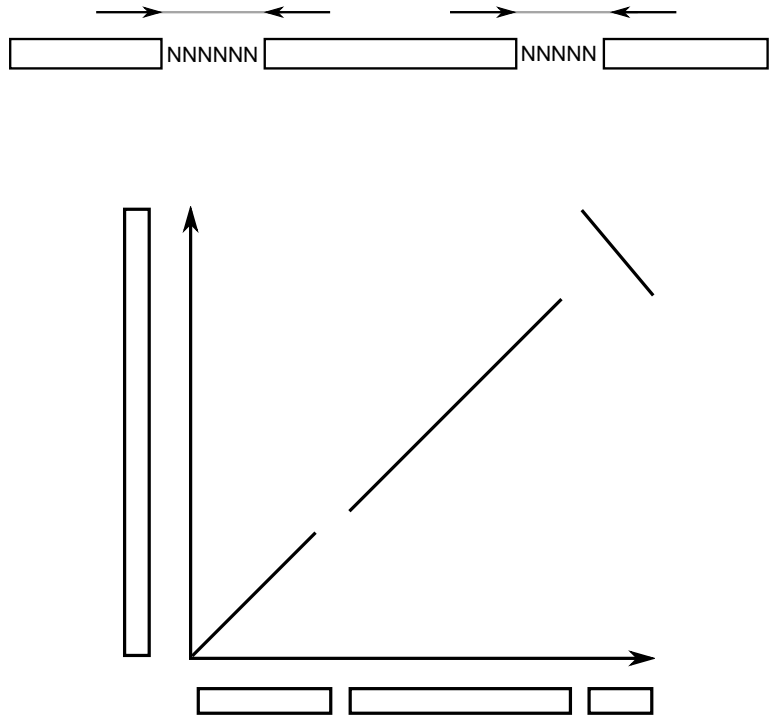
Listing 6: snippet of 454Scaffolds.fna

```
>scaffold00005   length=4643557
GAAACAGAATTTGCCTGGCGGCCGTAGCGCGGTGGTCCCACCTGACCCCATGCCGAACTC
AGAAGTGAAACGCCGTAGCGCCGATGGTAGTGTGGGGTCTCCCCATGCGAGAGTAGGGAA
CTGCCAGGCATCAAATTAAGCAGTAAGCCGGTCATAAAACCGGTGGTTGTAAAAGAATTC
 [......]
GGTTGTTGGTGGAAATTGTCGTGATATGGTGCGATATCGGCGTCATCCAGGCGTAGCGTC
AGGTTGCCGCCGTTGCGCTCATCCCAGCCTTTCAGCCAGGCGTCGGTGGTGGCTTTGATC
ATTCCCtGGACAAACCAGGACTGAGTAATGTTTTGCATGTTCTGTGTTCCTGTAAATTCG
GTGTTGTCGGATGCACGACCCGTAGGCCGGATAAGGCGCTCGCNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNCCCGTAGGCCGGATAAGGCGCTCGCGCCGCATCCGGCAG
TGTTTACCCGCGGCGACTCAAAATTTCTTTCTCATAAGCCCGCACGCTCTCCAGCCATTC
GCTACCTGCTGGCGTATCGTGACGTTGGCAATACATTTCCCAGACCGCCTGCCACGGCAA
CGATTTCTGCTCTTCCAGCAGTGCCAGACGCGCAGTGTAATCGCCCGCCGCTTCCAGCTT
GCGCAGCTCAGCGGTAGGTTCCAGCAACGCACGCAGCAGGGCTTTTTTCATATTGCGTGT
```

Fasta file where contigs are organized in scaffolds, with gaps filled with Ns

Genome fisnishing

- **Within scaffold gaps**: design primers using 454Scaffolds.fna
- **Between scaffold gaps**:
  - Alignment against a closely related taxa
  - Alignment against a physical map (e.g. optical map)
  - Genome walking



Illumina data set

NCBI SRA, accn SRR001665:

- 36 bp reads, PE with 200bp insert
- 20,816,448 reads

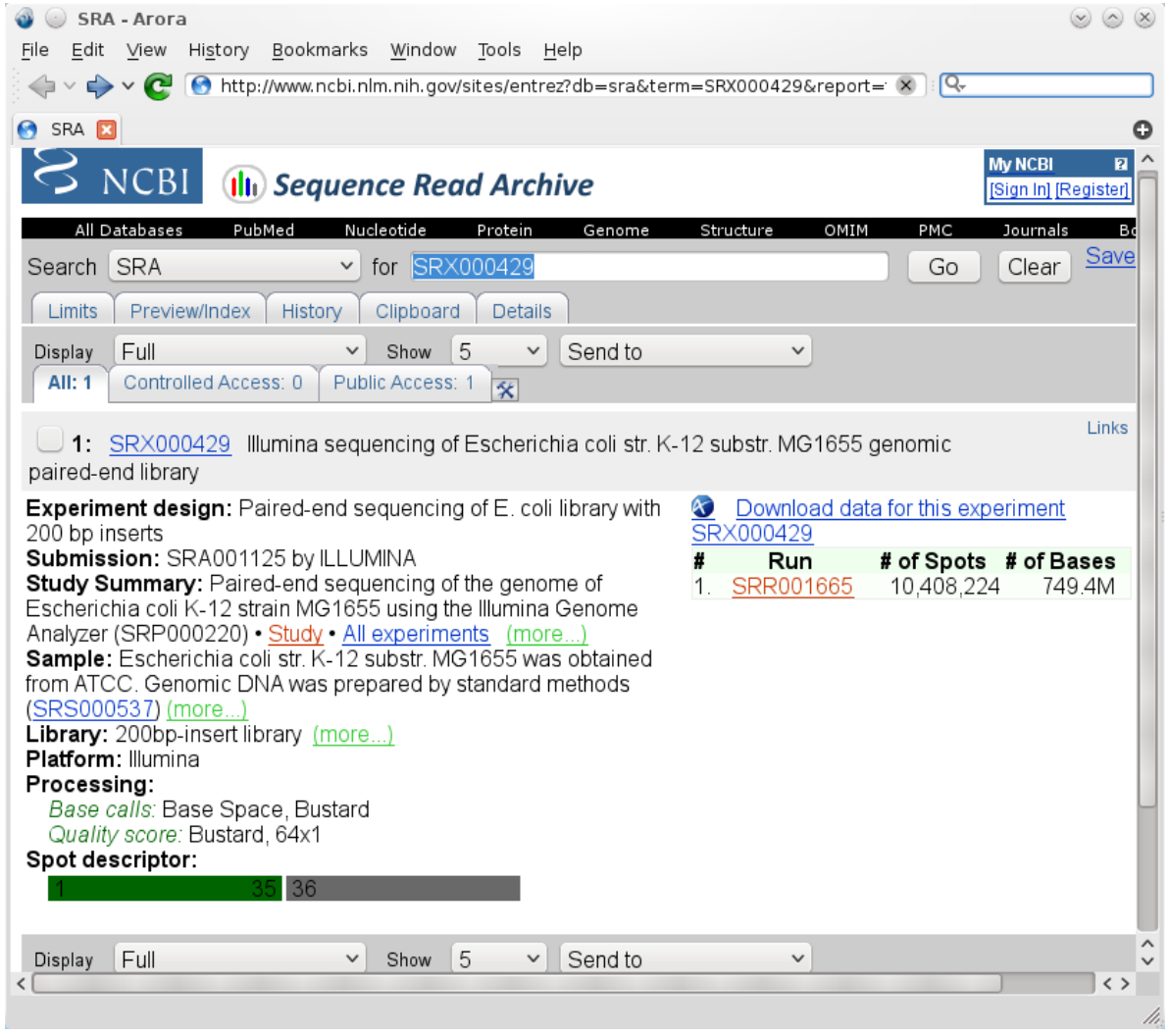
$$C = \frac{20816448 \times 36}{4639675}$$

$$C \sim 160X$$

- 2 files, one for each mate:

  SRR001665_1.fastq.gz

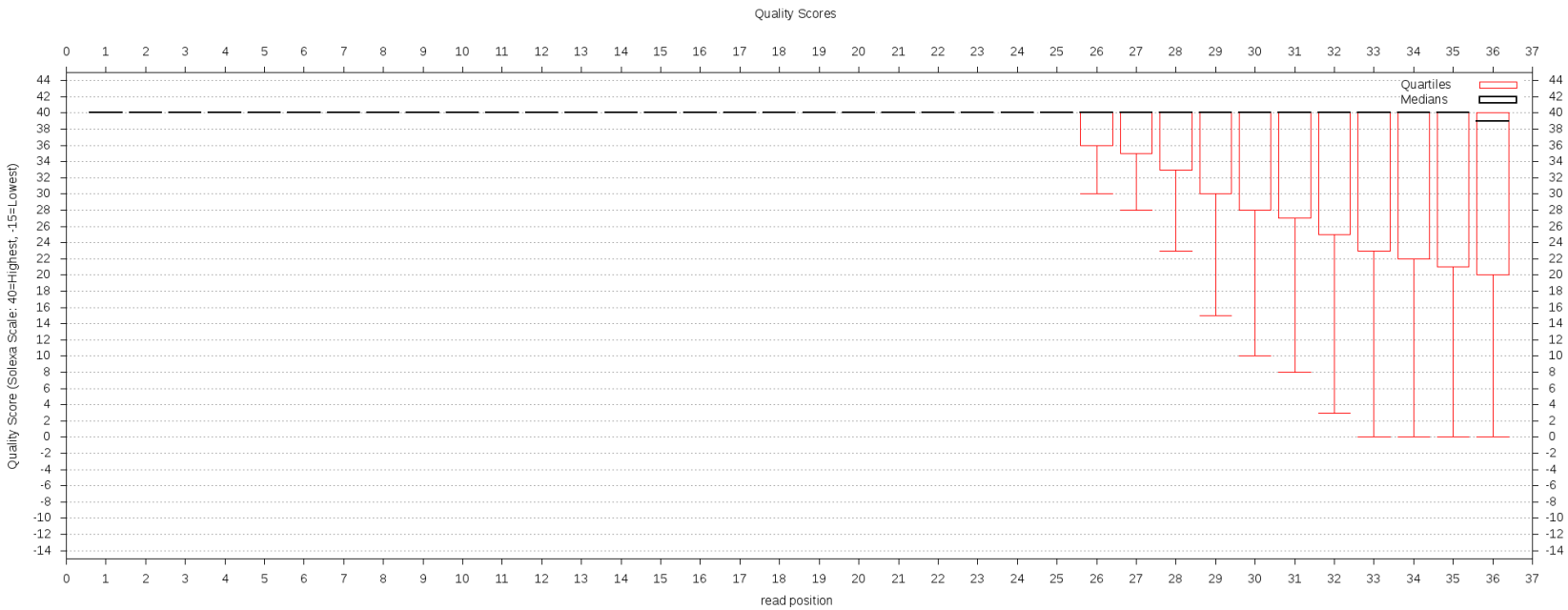
  SRR001665_2.fastq.gz



Quality check

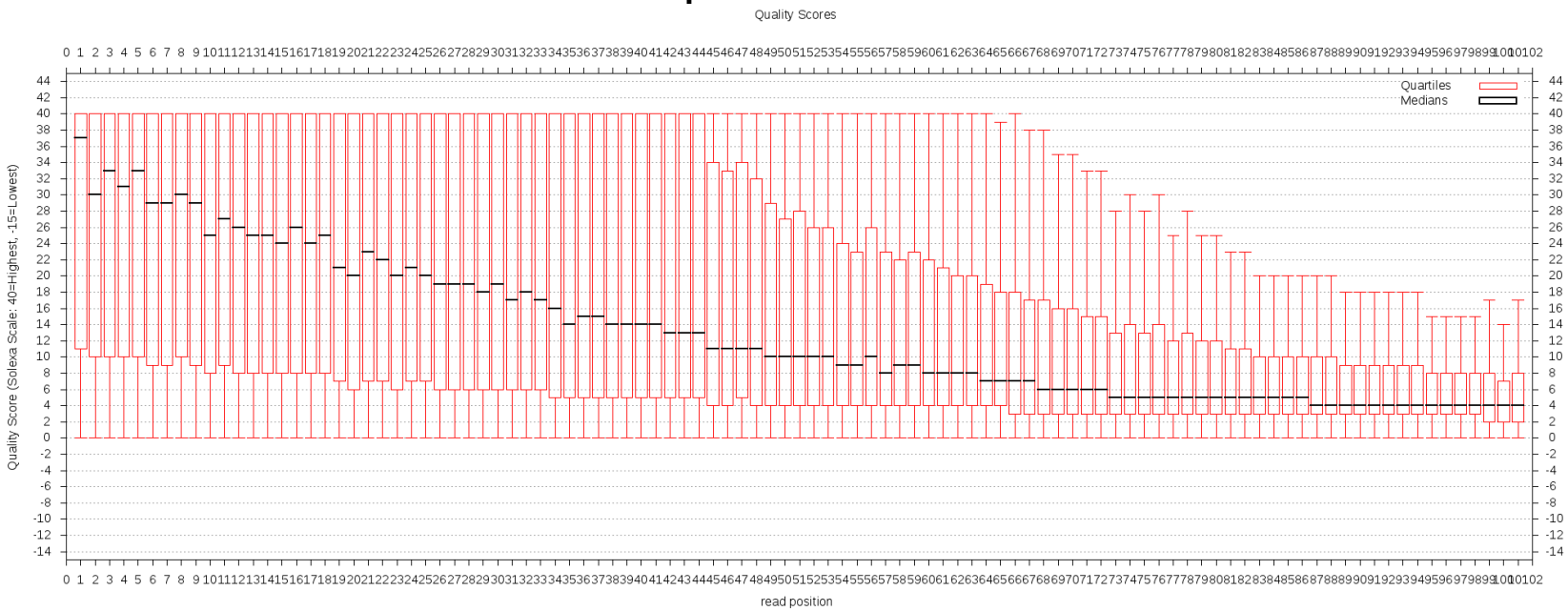
Listing 7: shell script running FastX

```
1 #Fastx quality (to do for each file)
2 fastx_quality_stats -Q 33 -i SRR001665_1.fastq -o SRR001665_1.fastq_qual_stat
3 fastq_quality_boxplot_graph.sh -i SRR001665_1.fastq_qual_stat -o
      SRR001665_1.fastq_qual_stat.png
4 #Trim reads of low quality
5 fastq_quality_trimmer -Q 33 -t 25 -i SRR001665_1.fastq -o SRR001665_1_trim25.fastq
```



A good looking, but bad run...

SRR034509: 10E6 PE 101bp reads



⇒ Do not use for de novo assembly!

Velvet

1. Preparing the paired-end data set

Listing 8: shell script to merged the paired-end fastq files

```
1 shuffleSequences_fastq.pl SRR001665_1.fastq SRR001665_2.fastq shuffled_seq.fastq
```

2. Build the kmer hash table (velveth)
3. Build the de Bruijn graph (velvetg)
4. Incorporate the PE information (option ins_length and exp_cov)
5. Simplify the graph (option cov_cutoff)

Max memory usage (Gb):

$$
\begin{aligned}
M &= (-109635 + 18977 \times ReadSize(bp) + 86326 \times GenomeSize(Mb) \\
&\quad + 233353 \times NumReads(M) - 51092 \times K(bp))/1048576 \\
&= (-109635 + 18977 * 36 + 86326 * 4.6 + 233353 * 20 - 51092 * 31)/1048576 \\
&\sim 3.9Gb
\end{aligned}
$$

Running Velvet, the manual way

Choosing the kmer-length using (must be odd):

$$C_k = C \times (L - k + 1)/L$$

$$k = L + 1 - \frac{C_k \times g}{n}$$

where k is the hash length, L the read length, C the coverage, g the genome size and n the number of reads.
If we target a C_k of 20, we get a k of 32.5, so we will use $k = 31$

Listing 9: shell script to manually run Velvet

```
#!/bin/bash
#choosing a kmer length (here 31) and building the kmer index
velveth ecolik12_31 31 -fastq -shortPaired shuffled_seq.fastq
#build the graph
velvetg ecolik12_31/
#manually estimate the k-mer coverage using stats.txt
#incorporate coverage and PE information
velvetg ecolik12_31/ -ins_length 200 -exp_cov 25 -cov_cutoff 10
```

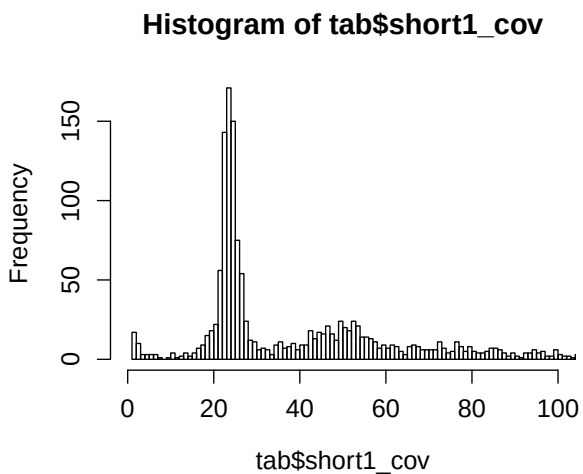
Manually estimating the kmer coverage

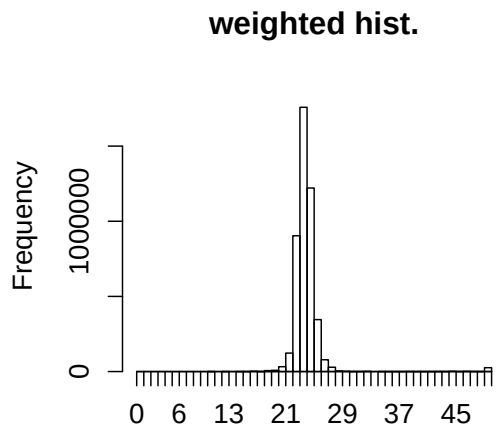
Listing 10: snippet of the stats.txt file

```
ID  lgth   out in  long_cov  short1_cov  short1_Ocov  short2_cov  short2_Ocov
1   32937  1   1   0.000000  25.170234   25.170234    0.000000    0.000000
2   23596  1   1   0.000000  23.367223   23.367223    0.000000    0.000000
3   48701  1   1   0.000000  24.246566   24.246566    0.000000    0.000000
```

Listing 11: R script to manually estimate the coverage

```
tab <- read.table("stats.txt", h=T)
hist(tab$short1_cov, breaks=50000, xlim=c(0,100))
library(plotrix)
weighted.hist(tab$short1_cov, tab$lgth, breaks=0:50)
```





Running Velvet with VelvetOptimiser

VelvetOptimiser: heuristics to get the best assembly by optimizing the kmer lenght and coverage cutoffs

```
1 #!/bin/bash
2 #Run VelvetOptimiser
3 #CAC V4 nodes have 16GB RAM, don't run more than 3 jobs in
      parallele (-t)
4 #kmer length: from 21 to 31
5 VelvetOptimiser.pl -s 21 -e 31 -t 3 -f '-fastq -shortPaired
      shuffled_seq.fastq' -o '-ins_length 200'
```

- Best parameters: kmer length: 31; exp. cov.: 23; cov. cutoff: 2
- N50: 95,399
- Number of contigs: 331
- Longest contig: 268,048 bp
- Total assembly: 4,566,467 bp

Running Velvet, the brute force way

Try them all...

Listing 12: bash script to run Velvet over many parameters

```
#!/bin/bash
# 2 nested loops
for i in {21..31..2}
  do
    velveth velveth_$i $i -fastq -shortPaired
        shuffled_seq.fastq
    velvetg velveth_$i
    for j in {0..100..10}
      do
        k=‘expr $j / 2‘
        velvetg velveth_$i -ins_length 200 -exp_cov $j
            -cov_cutoff $k
        cp velveth_$i/contigs.fa contigs_h${i}_cov${j}.fa
        cp velveth_$i/stats.txt stats_h${i}_cov${j}.txt
      done
  done
```

Choosing an assembly

Listing 13: R function to get Velvet assembly statistics

```
#A function that returns N50, total length, longuest contig,
#nbr of contigs (excluding the small one)
#Usage: asb_stat("stats.txt", 31)
asb_stat <- function(file, kmer, small.cut = 100) {
 tab <- read.table(file, h=T)
 nnodes <- nrow(tab)
 length.nt <- tab$lgth + kmer -1
 ncontig <- sum(length.nt > 100)
 biggest <- max(length.nt)
 total <- sum(length.nt)
 length.nt.sorted <- sort(length.nt, decreasing = T)
 cum.sum.length <- cumsum(length.nt.sorted)
 n50 <- length.nt.sorted[cum.sum.length > total/2][1]
 return(data.frame(file=file,n50=n50, total.length=total, longuest=biggest, ncontig=ncontig))
}
```

Listing 14: R script to select the best assembly after a brute force run of Velvet

```
#list of stat files
list.stat <- dir(pattern = 'stats_')
#associated kmer length
km <- sub('.*_h', '', list.stat)
km <- as.numeric(sub('_cov.*', '', km) )

#calculate stats
asb.stats <- NULL
for (i in 1:length(list.stat)) {
  asb.stats <- rbind(asb.stats, asb_stat(list.stat[i], km[i]))
}

#rank assemblies
min_length <- 4E6
filt.asb <- asb.stats[asb.stats$total.length > min_length, ]
ranks <- order(filt.asb$n50, -filt.asb$ncontig, filt.asb$longuest, filt.asb$total.length, decreasing=T)

#best assembly is:
filt.asb[ranks, ][1,]
```

More Velvet tricks...

- The “auto“ option:

  ```
  velvetg <dir> -cov_cutoff auto -exp_cov auto ...
  ```

- Using two PE libraries:

  ```
  velveth <dir> 25 -fastq -shortPaired <file1> -shortPaired2
  velvetg <dir> -ins_length 200 -ins_length2 5000 ...
  ```

- Compilation option:

  ```
  make 'CATEGORIES=3' 'MAXKMERLENGTH=75'
  ```

- Remove genome “parasites”: plasmids, mitochondrial and chloroplastic genomes

  ```
  velvetg <dir> -max_coverage 200 ...
  ```

- Adding long reads or contigs to the assembly

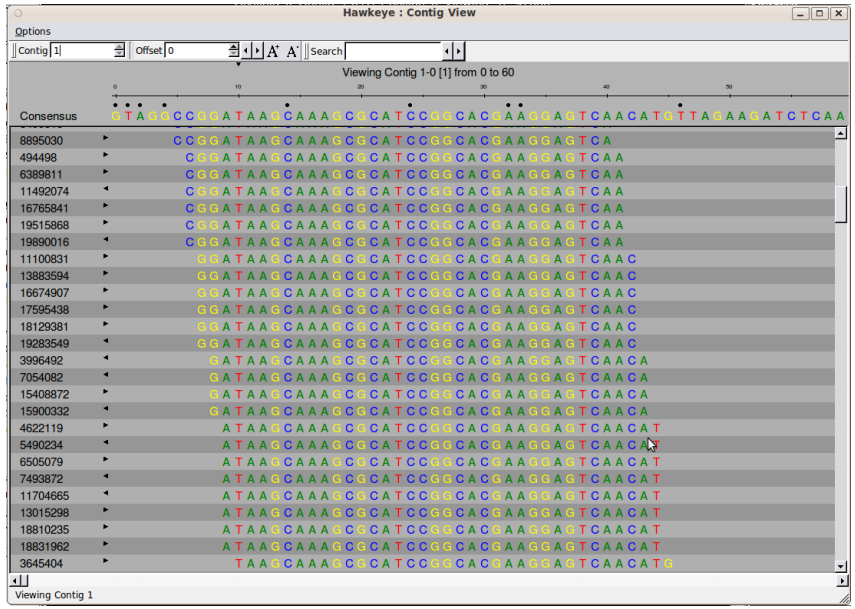
  ```
  velveth <dir> 25 -fasta -log <fasta file> ...
  ```

Visualizing the assembly (1)

Listing 15: Shell script to visualize the assembly with Hawkeye

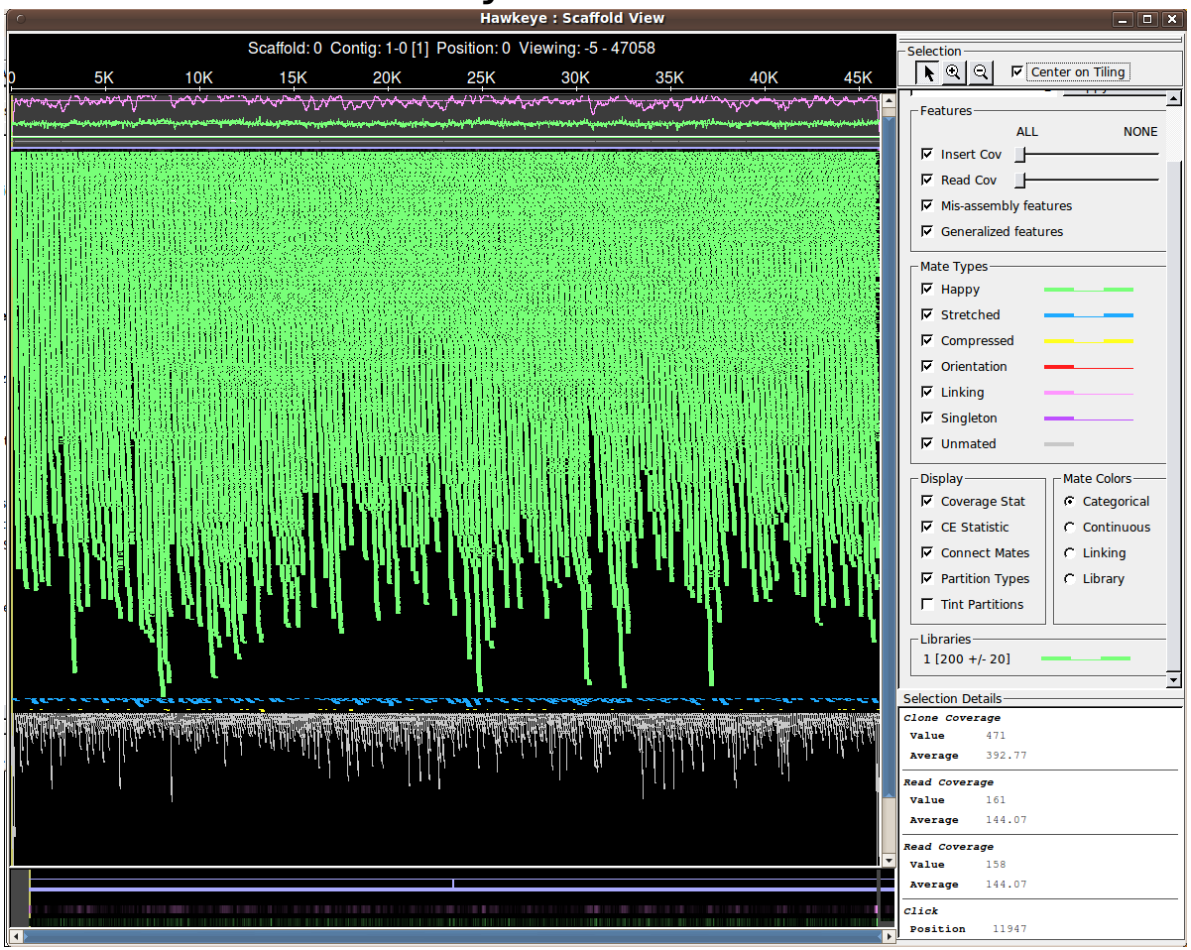
```
1 #run velvet with the AFG option
2 velvetg ecolik12_31/ --ins_length 200 --exp_cov
      25 --cov_cutoff 10 --amos_file yes
3 #To reduce memory usage, extract the first
      contig from the AFG file (the script
      comes with Velvet)
4 asmbly_splitter.pl 1 velvet_asm.afg
5 #Use AMOS tools to prepare a bank and then
      visualise with hawkeye
6 bank-transact -m velvet_asm_1.afg -b
      velvet_asm_1.bnk -c
7 hawkeye -t velvet_asm_1.bnk/
```

Hawkeye, contig view



Visualizing the assembly (2)

Hawkeye, scaffold view



Other visualization tools:

- Tablet
- Consed
- ...

454 vs Illumina

Methods	N50	Total	Max	N cont	N scaf	Cost
454, Newbler	106KB	4.55MB	268KB	122	6	∼$6,700
Illumina, Velvet (manual)	95KB	4.58MB	268KB	178		∼$2,100
(VelvetOptimiser)	95KB	4.58MB	268KB	180		
(brute force)	116KB	4.56MB	356KB	126		

Documentation

de novo assemblies	CBCB assembly starter
	Pop, 2009, Brief. Bioinf.
	Miller et al, 2010, Genomics
	Illumina tech. note on de novo assembly
list of programs	SEQanswers wiki
Velvet	Zerbino & Birney, 2008, Genome.Res.
	Zerbino et al, 2009, PLoS ONE
Newbler	454 technical documentation
	Margulies et al, 2005, Nature