



De Novo Illumina Assemblies with Velvet

Velvet

- Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs. D.R. Zerbino and E. Birney. Genome Research 18:821-829. (2008)



- Pebble and Rock Band: Heuristic Resolution of Repeats and Scaffolding in the Velvet Short-Read de Novo Assembler. Zerbino DR, McEwen GK, Margulies EH, Birney E. PLoS One. 2009 Dec 22;4(12):e8407.



- Using the Velvet de Novo Assembler for short-read sequencing technologies. Zerbino DR. Curr Protoc Bioinformatics. Chap 11:Unit 11.5

Overview

- Run velveth
- Background
- Run velvetg
- Details on selecting velvetg parameters
- Post-assembly analyses

Velvet, a short read *de novo* assembler

- Velvet is currently one of the most popular *de novo* assemblers for short (25-150 bp) read data
- Velvet does not use quality scores, instead it uses coverage data for error correction
- You should trim low quality bases from your reads prior to assembly (if necessary after QC).

Getting started

- **CAREFULLY READ THROUGH THE GETTING STARTED BEFORE YOU BEGIN THE ACTIVITY**
- <http://evomics.org/learning/assembly-and-alignment/velvet/>
- Check your disk space! Clean if necessary
- Complete exercise 1 and start exercise 2

Getting started

- Exercise 2:
- velveth auto 31,45,2 -fastq -shortPaired1....
- 1/2 of the class run velveth with 31,37,2 and the other 1/2 with 39,45,2

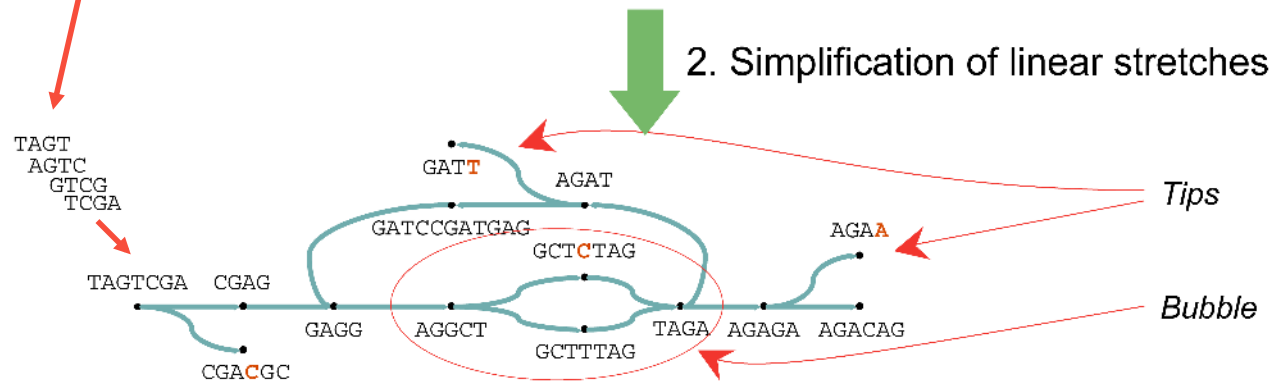
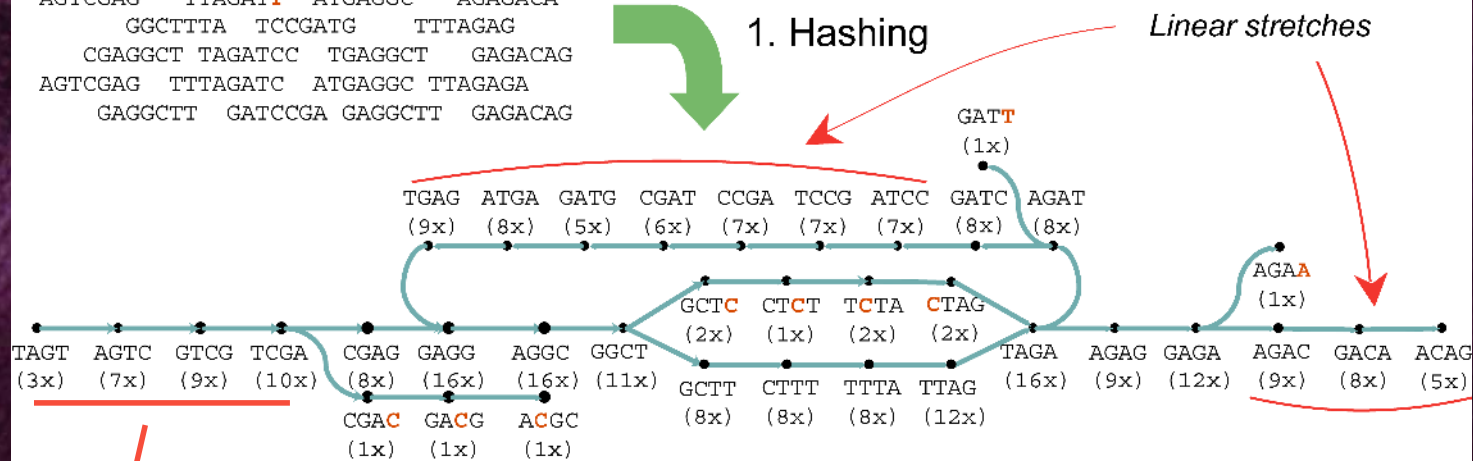
Assembly strategies: Eulerian paths on de Bruijn graphs

- Pavel Pevzner 2001 (PNAS)
- k-mers instead of whole reads



TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG
 AGTCGAG CTTTAGA CGATGAG CTTTAGA
 GTCGAGG TTAGATC ATGAGGC GAGACAG
 GAGGCTC ATCCGAT AGGCTTT GAGACAG
 AGTCGAG TAGATCC ATGAGGC TAGAGA
 TAGTCGA CTTTAGA CCGATGA TTAGAGA
 CGAGGCT AGATCCG TGAGGCT AGAGACA
 TAGTCGA GCTTTAG TCCGATG GCTCTAG
 TCGACGC GATCCGA GAGGCTT AGAGACA
 TAGTCGA TTAGATC GATGAGG TTTAGAG
 GTCGAGG TCTAGAT ATGAGGC TAGAGAC
 AGGCTTT ATCCGAT AGGCTTT GAGACAG
 AGTCGAG TTAGATT ATGAGGC AGAGACA
 GGCTTTA TCCGATG TTTAGAG
 CGAGGCT TAGATCC TGAGGCT GAGACAG
 AGTCGAG TTTAGATC ATGAGGC TTAGAGA
 GAGGCTT GATCCGA GAGGCTT GAGACAG

Sequencing
(e.g. Illumina, SOLiD, ..)



3. Error removal



From Zerbino and Birney (2008)

TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG

AGTCGAG CTTTAGA CGATGAG CTTTAGA

GTCGAGG TTAGATC ATGAGGC GAGACAG

GAGGCTC ATCCGAT AGGCTTT GAGACAG

AGTCGAG TAGATCC ATGAGGC TAGAGA A

TAGTCGA CTTTAGA CCGATGA TTAGAGA

CGAGGCT AGATCCG TGAGGCT AGAGACA

TAGTCGA GCTTTAG TCCGATG GCTCTAG

TCGACGC GATCCGA GAGGCTT AGAGACA

TAGTCGA TTAGATC GATGAGG TTTAGAG

GTCGAGG TCTAGAT ATGAGGC TAGAGAC

AGGCTTT ATCCGAT AGGCTTT GAGACAG

AGTCGAG TTAGAT T ATGAGGC AGAGACA

GGCTTTA TCCGATG TTTAGAG

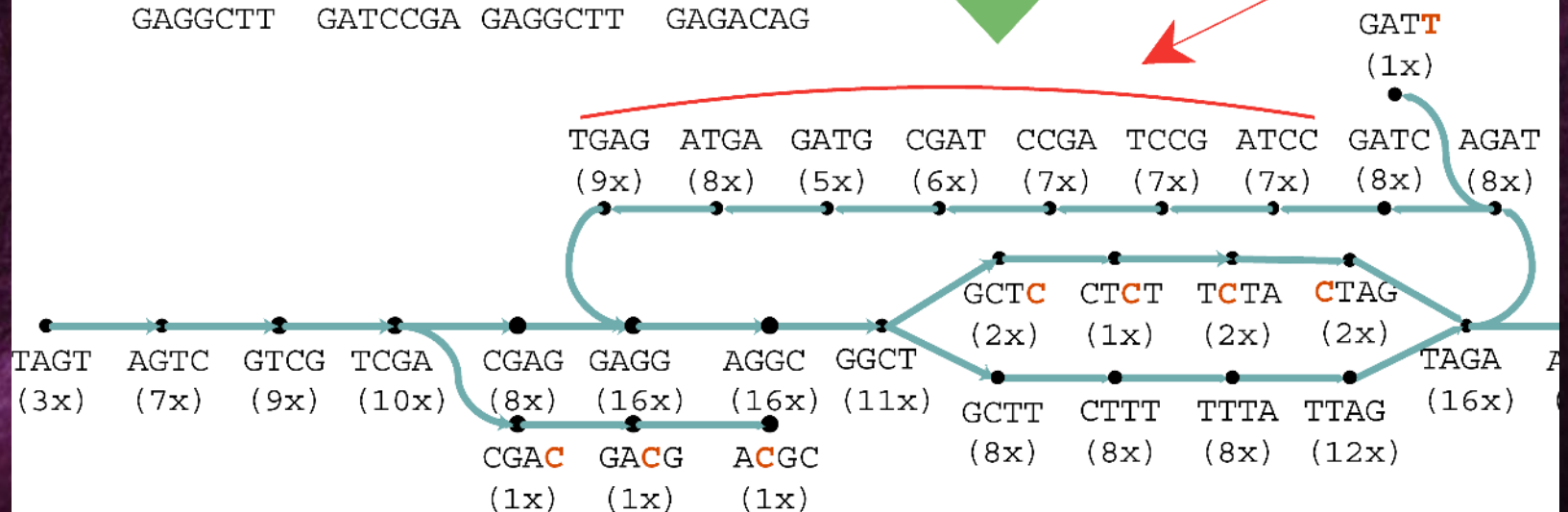
CGAGGCT TAGATCC TGAGGCT GAGACAG

AGTCGAG TTTAGATC ATGAGGC TTAGAGA

GAGGCTT GATCCGA GAGGCTT GAGACAG

Sequencing
(e.g. Illumina, SOLiD,

1. Hashing



Creating a hashtable

- $k=4$, read length=7

Reads: **AGTCGAG**

kmers: AGTC

GTCG

TCGA

CGAG

CTTTAGA

CTTT

TTTA

TTAG

TAGA

...

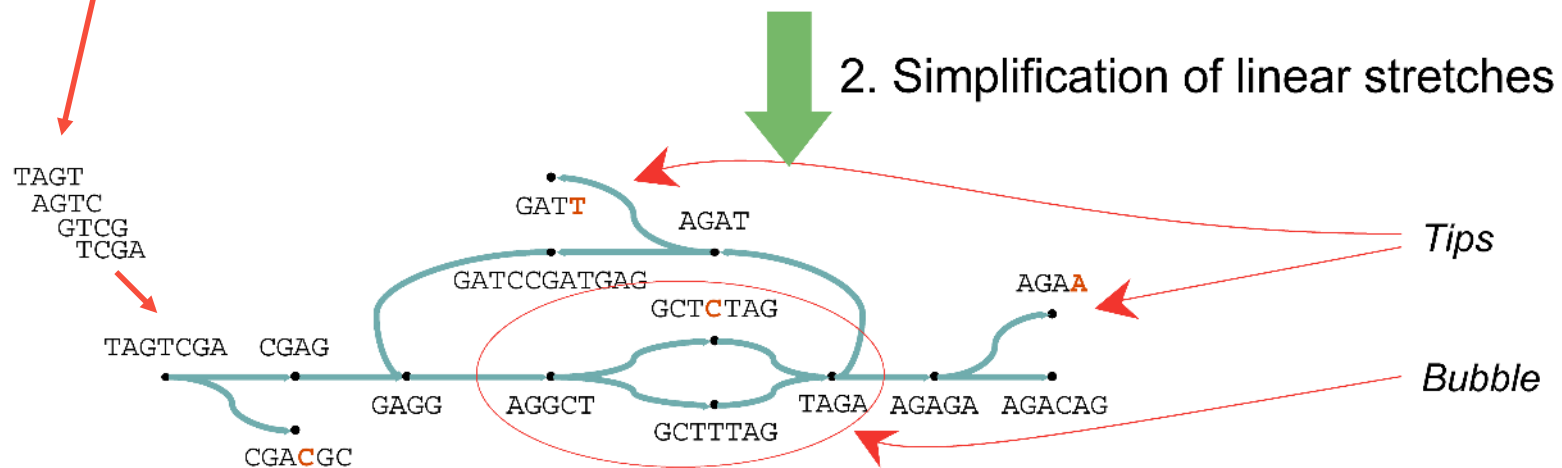
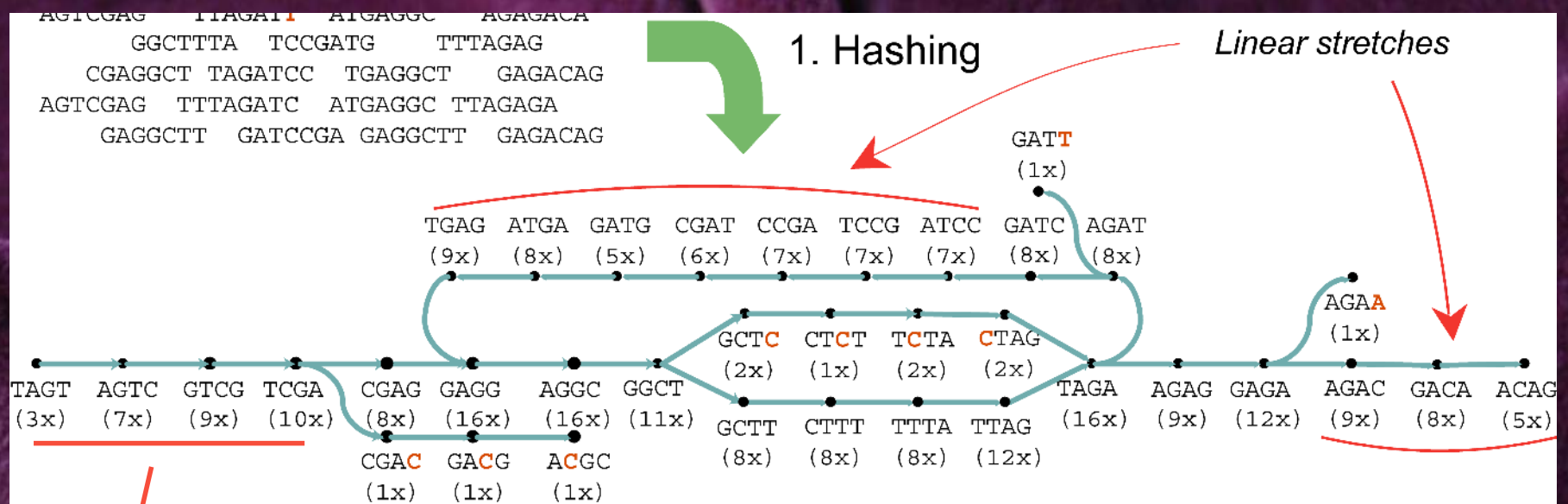
Creating a hashtable

- AAAA: 8
- AAAC: 10
- AACT: 9
- ...

Building the graph



- Choose a kmer (educated guess...): AGTC
- Find its extensions: kmers that start with GTC (4 at most)
- Iterate



3. Error removal



From Zerbino and Birney (2008)

Continue exercises

- Finish exercise 2 and start exercise 3.

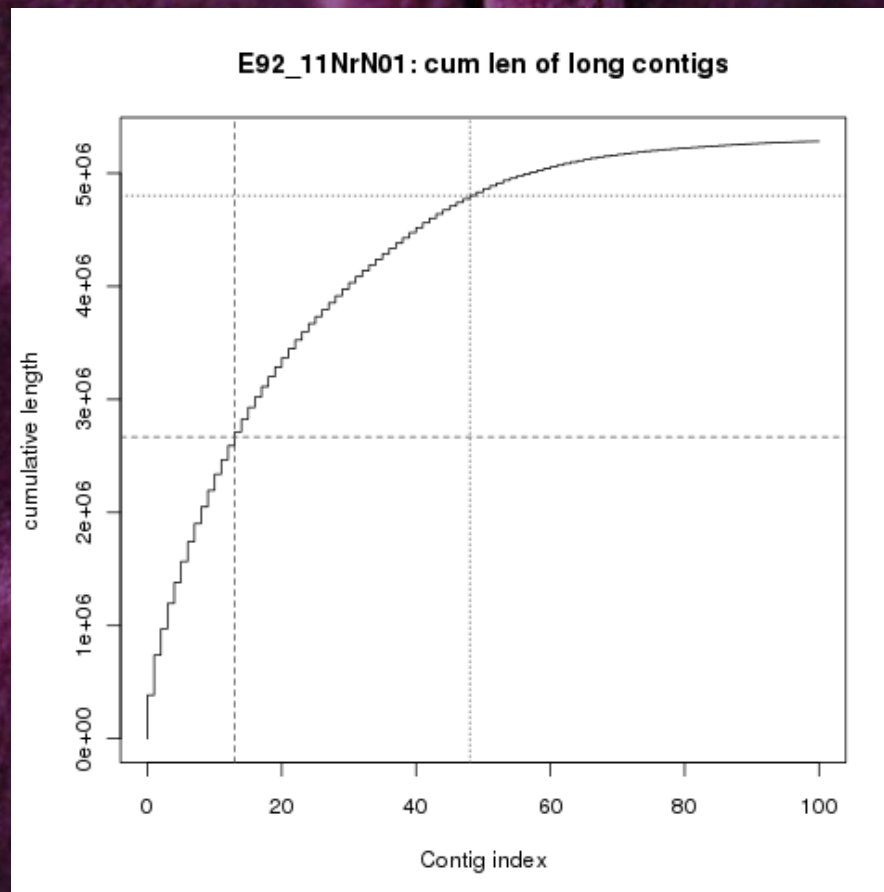
Primary velvetg parameters

- k
 - Short k-mer high accuracy, short contigs
 - Long k-mer lower accuracy, longer contigs
- exp_cov
 - = in k-mer coverage !!!
- ins_length

Assembly statistics: n50 and n90

- "Contig or scaffold N50 is a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value"
-seqanswers.com
- Especially helpful compared to the maximum contig size

Assembly statistics: n50 and n90



k-mer coverage (C_k)

- All coverage values in Velvet are in *k*-mer coverage, not nucleotide coverage
- $C_k = C * (L - k + 1) / L$
 - k = *kmer* length, L = read length, C = nucleotide coverage
- *k*-mer coverage is always lower than the actual bp coverage!
- For example: When $k=31$ and $L=36$, your *k*-mer coverage is 1/6 of the actual coverage
- The longer the *kmer*, the lower the coverage

Continue activity

- Finish exercises 4-7
- Report results in the Excel spreadsheet on display

VelvetOptimiser (Perl script)

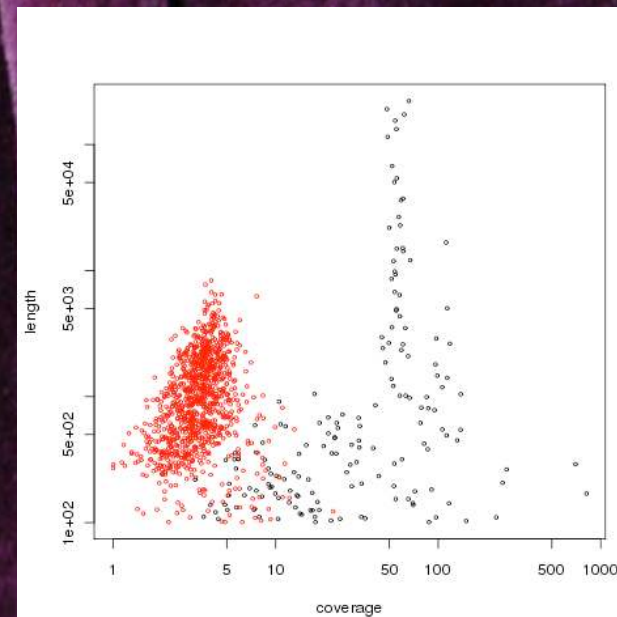
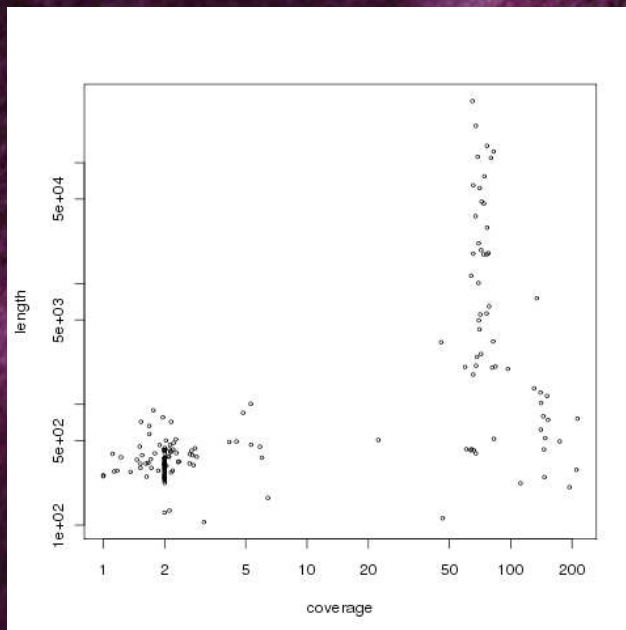
- Simon Gladman
- Runs velvet on a *k-mer* range you specify to determine optimum
- Estimates optimum *exp_cov* and then searches for optimum *cov_cutoff*
- Pros:
 - Optimization functions (n50, max contig, tbp)
 - Good job of guessing amount of RAM required
- Cons:
 - Can't specify most velvetg parameters directly
- It is recommended to re-run velvetg with VelvetOptimiser parameters and try out additional parameters

Post-assembly analyses

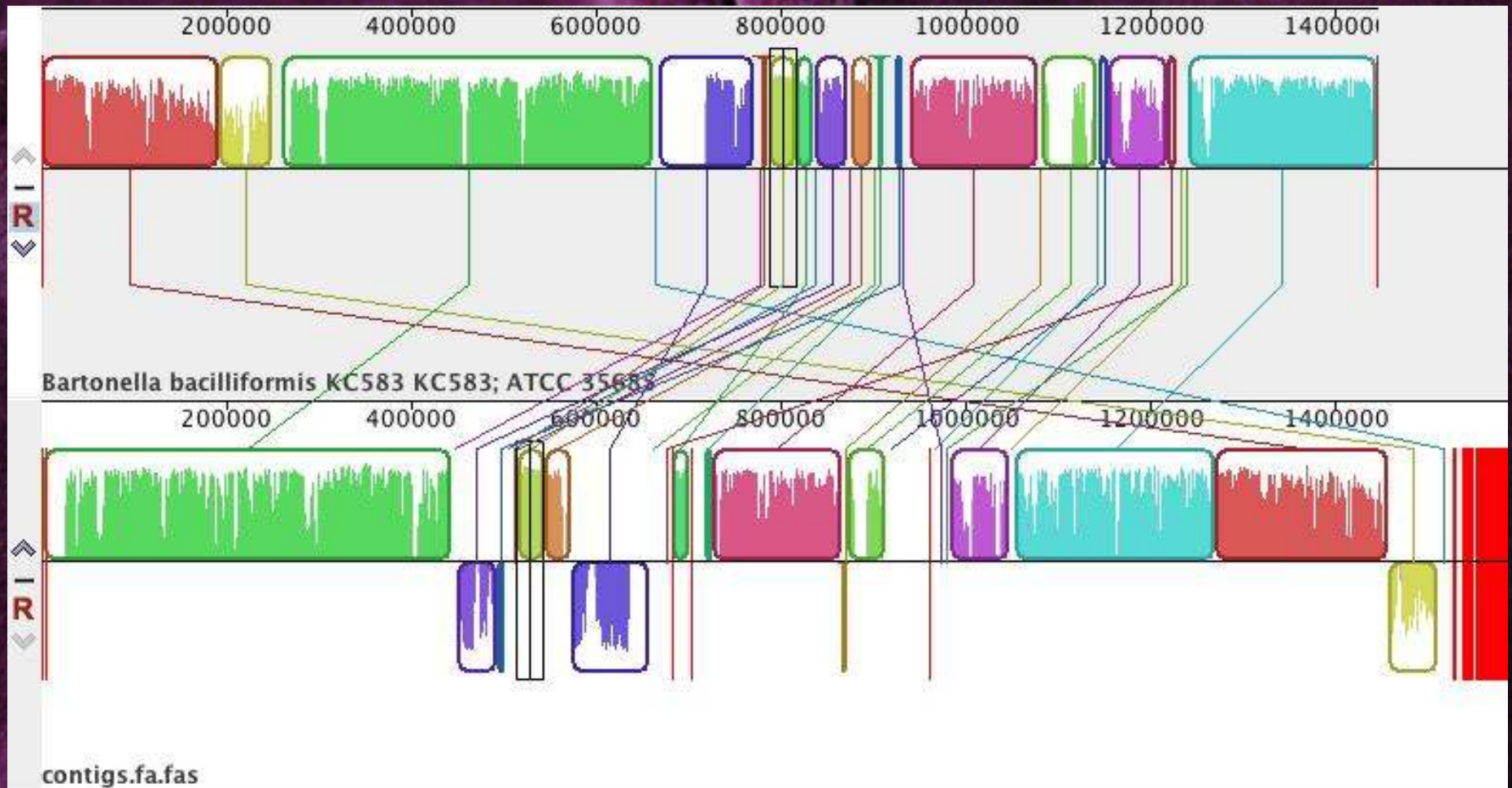
- Always check your assembly for obvious misassemblies
 - Use tablet and/or a whole genome alignment to reference genome (Mauve, nucmer+ACT, ...)
- Contigs that represent adjacent genomic regions may have overlap of $< 2k$, and sometimes can be merged (larger contigs!)

Post-assembly analyses

- Things to consider:
 - Check contamination (BLAST to nr, “metagenomics” approach)
 - Coverage vs contig length

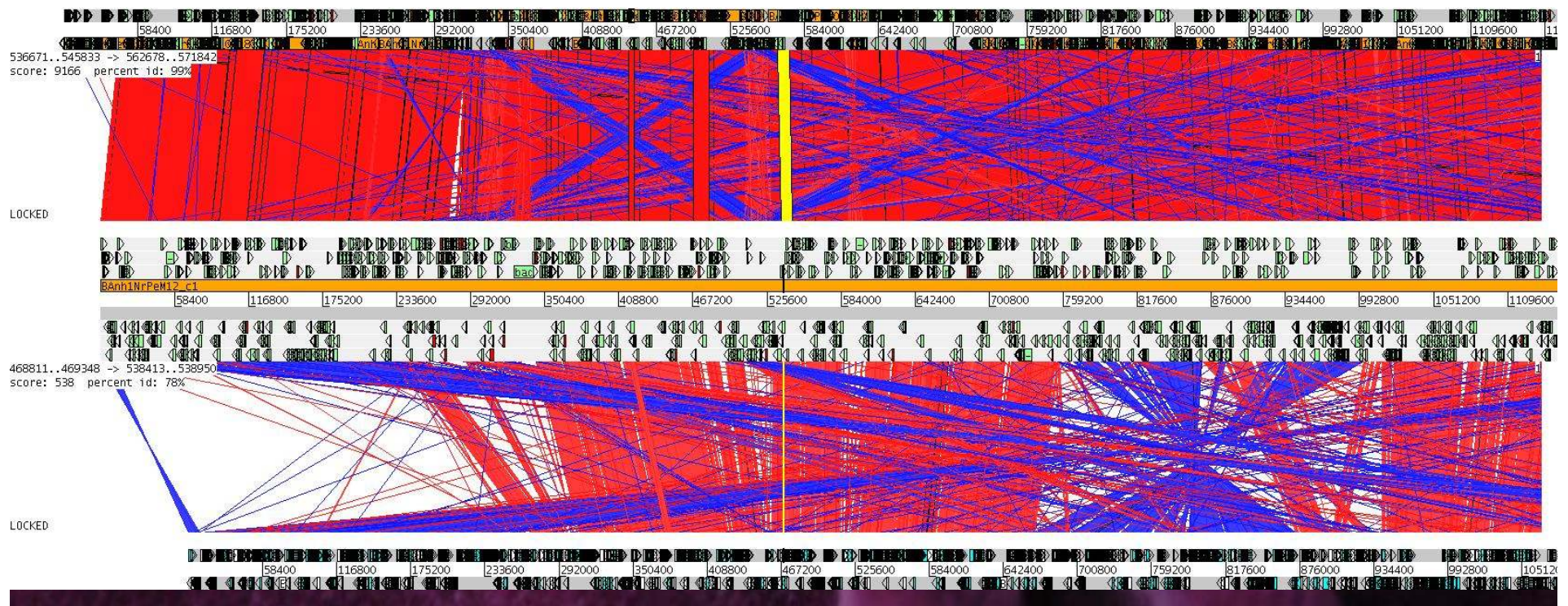


Post assembly analyses (Mauve)



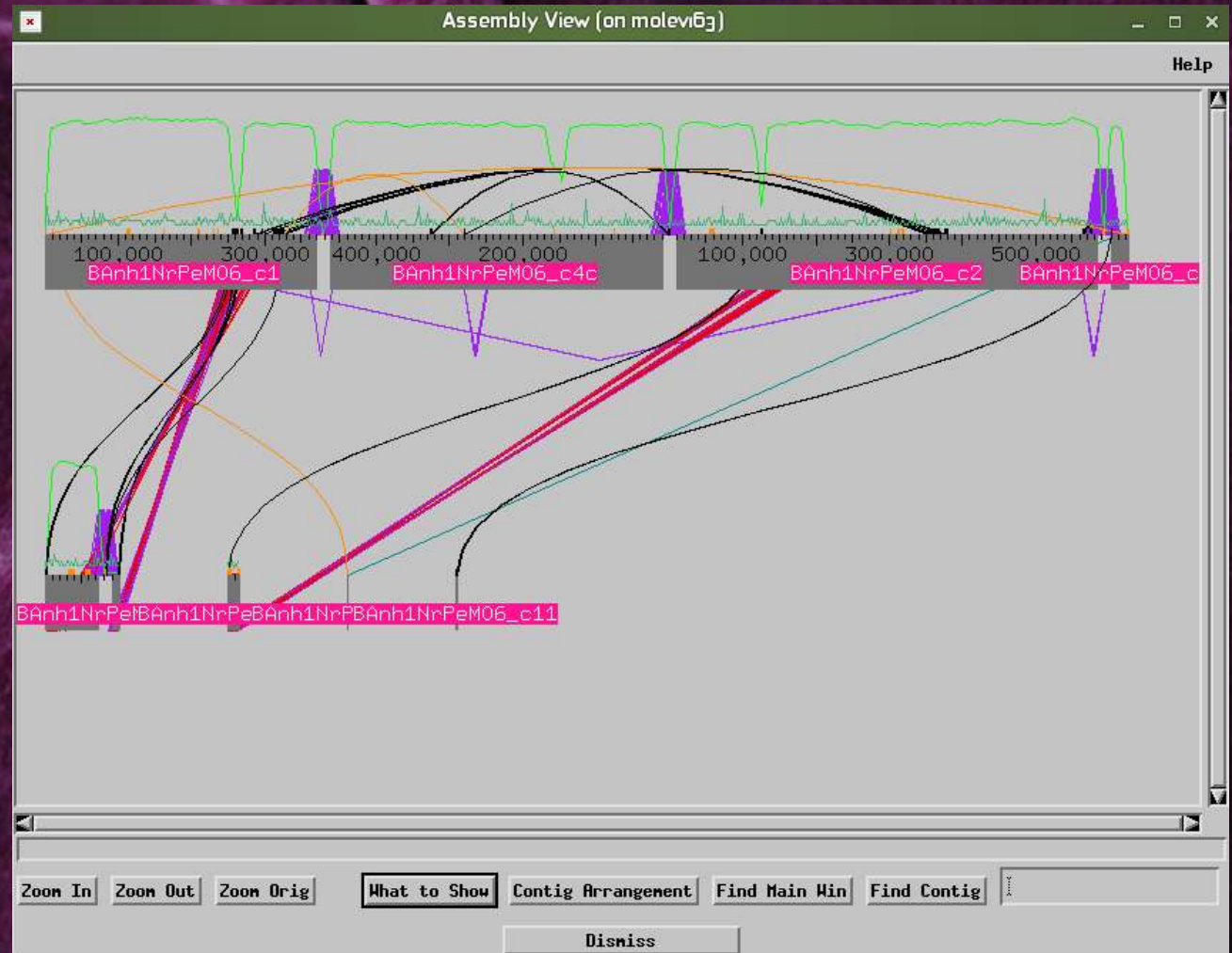
Post-assembly analyses

- Order contigs/scaffolds with respect to a reference assembly or a close genome (nucmer)
- Compare with ACT (Artemis comparison tool)



Post-assembly analyses

- Even coverage?
- Even distribution of good paired-end reads?



Hardware considerations

- A 1GB genome project with a mixture of ~1 billion reads (~73x) is using 300GB+ of RAM
- Some assemblers such as ABySS tend to use less memory
- Opt for a processor with higher L3 cache (24MB) than simply clock speed
- Lots of disk space

Velvet associated software

- Oases: *de novo* transcriptome assembler for short reads
- MetaVelvet: *de novo* assembly of metagenomic data
- Columbus module: allows the assembly process to be assisted by alignment information onto a set of reference sequences
- Curtain: *de novo* assembly of large genomes