

# *De novo* genome assembly of NGS data

Dr Torsten Seemann

*Victorian Bioinformatics Consortium  
Monash University*

**IMB Winter School 2011**



*Victorian  
Bioinformatics  
Consortium*



**MONASH**  
University

# Outline

- Concepts
  - genome, read, graph, k-mer
- Genome assembly
  - OLC and Eulerian method, software
- Assembly metrics
  - N50 explained
- If we have time...
  - simple example of running “Velvet”
- Contact

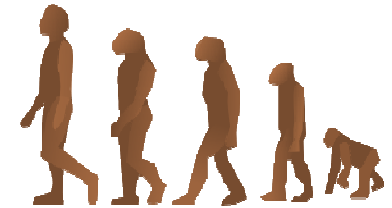


# What is a genome?



- The entire set of DNA that makes up a particular organism
  - Chromosomes
  - Organelles: mitochondria, chloroplast, ...
  - Plasmids
  - Viruses (some are RNA not DNA)
  - Bacteriophage
- Essentially just a set of strings
  - Using the 4 letter DNA alphabet { A,G,C,T }

# Genome sizes



- Virus, Plasmid, Phage
  - 1 kbp to 100 kbp ... *HIV 9181 bp*
- Bacteria, Archaea
  - 1 Mbp to 10 Mbp ... *E.coli 4.6 Mbp*
- Simple Eukaryotes
  - 10 Mbp to 100 Mbp ... *Malaria 23 Mbp*
- Animals, Plants
  - 100 Mbp to >100 Gbp ! ... *Human 3.2 Gbp*

# Genome sequencing



- Hierarchial (“Old School”)
  - Restriction frags, vectors, exo deletion, ...
  - Labour intensive but some advantages
- Whole Genome Shotgun (“WGS”)
  - Shear DNA to appropriate size
  - Do some library preparation
  - Put in sequencing machine
  - Get some big text files with your reads
  - Panic when `NOTEPAD.EXE` won't load them

# Genome assembly

DNA copies  
of the genome



Sequence Reads



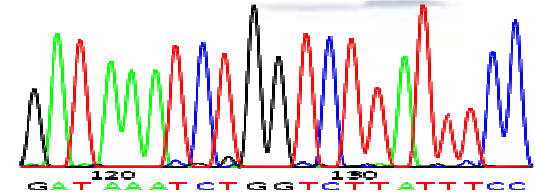
Assembled genome



Previous talk

This talk

# Read technologies



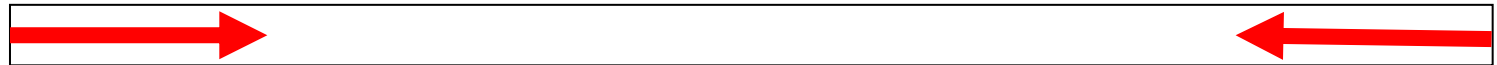
Method	bp	Attributes
<i>Sanger</i>	1200	low yield, low quality at both ends
<i>Roche 454</i>	700	Homo-polymer errors
<i>Illumina</i>	150	Low quality at 3' end
<i>SOLiD</i>	75	Difficult to work in colour-space
<i>IonTorrent</i>	100	High error rate, homo-polymers
<i>PacBio</i>	3000	Very high error rate, can iterate

# Read types

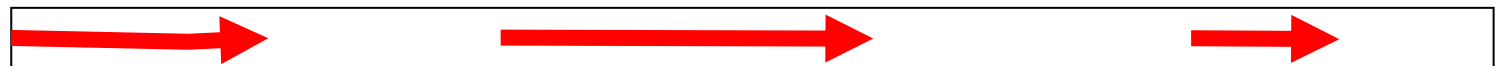
- Single end read
  - Only have sequence from one end of fragment



- Paired / Mated read
  - Have sequence from both ends of fragment



- Strobe / Jumping read
  - Have a series of segments of the fragment





# Reads from genomes



- Short sub-sequences of the genome
  - Don't know where in genome they originate
  - Don't know their orientation (strand)
- Overlap each other
  - Assuming we over-sampled the genome
- Contain errors
  - Wrong base calls, extra/skipped bases
- Represent all of the genome
  - You get most, but coverage is not uniform

# What is genome assembly?

- *De novo* genome assembly is the process of reconstructing the DNA sequences of an organism from its read sequences alone.
- Ideal world
  - Reads long and error-free (unambiguous)
  - Simple deduction problem
- Real world
  - Reads too short and error-prone (ambiguity)
  - Complicated inference problem

# Draft vs Finished genome

- *De novo* assemble your reads into contigs
  - Unique chunks of contiguous DNA
- Join contigs into scaffolds
  - Only if you have paired or strobed reads!
  - Scaffolds contain gaps of ambiguous Ns
- Close gaps
  - Design primers around gaps
  - Sequence the resultant PCR product
- Validate
  - Align your reads back onto the draft genome
  - Check against optical maps (up to 100 MBp)

# Assembly confusion

- Reference assembly
  - We have sequence of very similar genome
  - Reads are aligned to the reference
  - Can guide, but can also mislead
  - Used a lot in human genomics
- *De novo* assembly
  - No prior information about the genome
  - Only supplied with read sequences
  - Necessary for novel genomes eg. Coral
  - Or where it differs from reference eg. Cancer

# Repeats. Repeats. Repeats.

- A repeat is a segment of DNA that occurs more than once in the genome
  - SINEs, LINEs eg ALU repeats
  - transposons, insertion seqs, paralogous genes
- It is impossible to resolve a repeat that is longer than your effective read length
  - Small scale: paired-end reads, < 1 kbp
  - Medium scale: mate-pair reads, 2-20 kbp
  - Large scale: fosmid/BAC libraries, 40-200 kbp
  - Huge scale: optical maps, 10-100 Mbp



# What is a “k-mer” ?

- A k-mer is a sub-string of length k
- A string of length L has  $(L-k+1)$  k-mers
- Example read  $L=8$  has 5 k-mers when  $k=4$

– **AGATCCGT**

– AGAT

– GATC

– ATCC

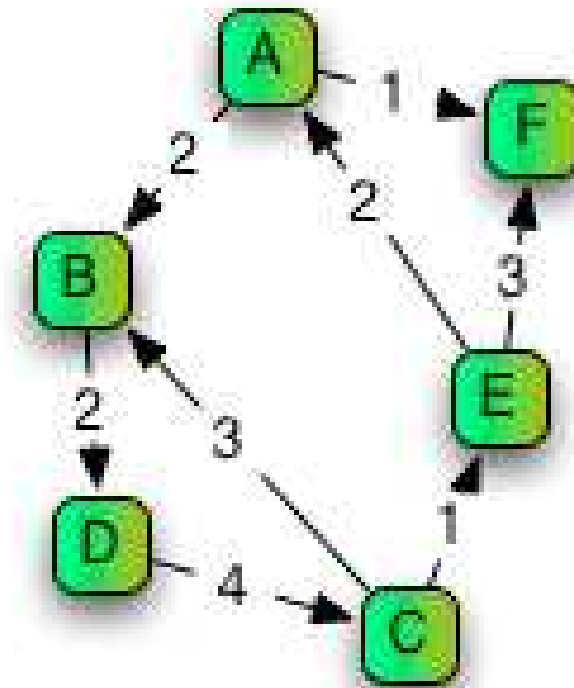
– TCCG

– CCGT



# What is a graph ?

- Not an Excel chart!
- Nodes / Vertices
  - A,B,E,G,H,K,M
- Edges / Arcs
  - (lines between nodes)
- Directed graph
  - Arrow head on edge
- Weighted graph
  - Numerals on edges





# Overlap - Layout - Consensus

- **Overlap**
  - All against all pair-wise comparison
  - Build graph: nodes=reads, edges=overlaps
- **Layout**
  - Analyse/simplify/clean the overlap graph
  - Determine Hamiltonian path (NP-hard)
- **Consensus**
  - Align reads along assembly path
  - Call bases using weighted voting

# OLC : Pairwise Overlap

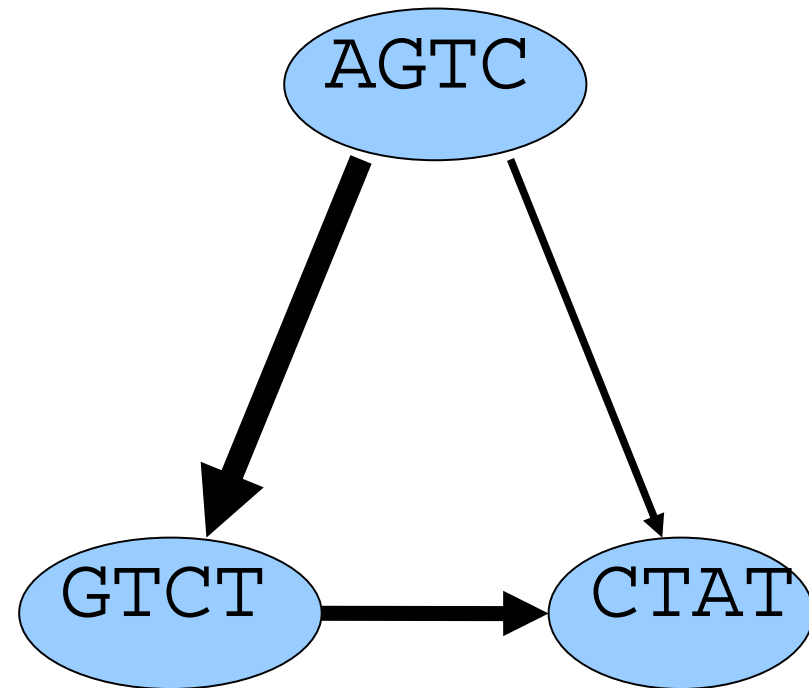
- All against all pair-wise comparison
  - $\frac{1}{2} N(N-1)$  alignments to perform [N=no. reads]
  - Each alignment is  $O(L^2)$  [L=read length]
- In practice, use smarter heuristics
  - Index all k-mers from all reads
  - Only check pairs that share enough k-mers
  - Similar approach to BLAST algorithm
- Both approaches parallelizable
  - Each comparison is independent

# OLC: Overlap Example

- True sequence (7bp)
  - AGTCTAT
- Reads (3 x 4bp)
  - AGTC, GTCT, CTAT
- Pairs to align (3)
  - AGTC+GTCT, AGTC+CTAT, GTCT+CTAT
- Best overlaps
  - AGTC-                      AGTC---                      GTCT--
  - -GTCT                      ---CTAT                      --CTAT
  - (*good*)                      (*poor*)                      (*ok*)

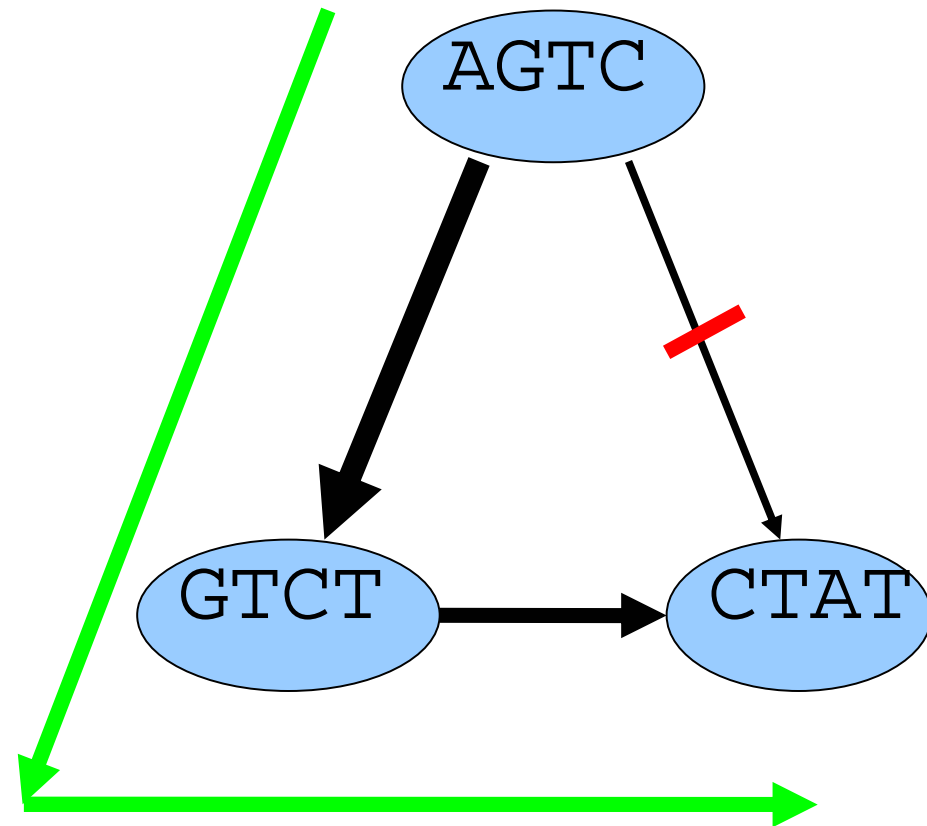
# OLC: Overlap Graph

- Nodes are the 3 read sequences
- Edges are the overlap alignment with orientation
- Edge thickness represents score of overlap



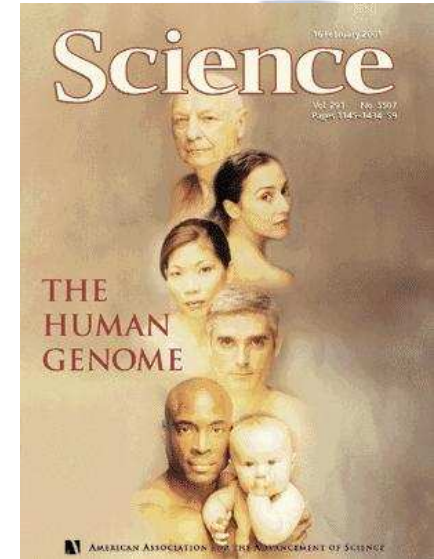
# OLC: Layout - Consensus

- Optimal path shown in green
- Un-traversed weak overlap in red
- Consensus is read by outputting the overlapped nodes along the path
- aGTCTCTat



# OLC : Software

- Phrap, PCAP, CAP3
  - Smaller scale assemblers
- Celera Assembler
  - Sanger-era assembler for large genomes
- Arachne, Edena, CABOG, Mira
  - Modern Sanger/hybrid assemblers
- **Newbler (gsAssembler)**
  - Used for 454 NGS “long” reads
  - Can be used for IonTorrent flowgrams too

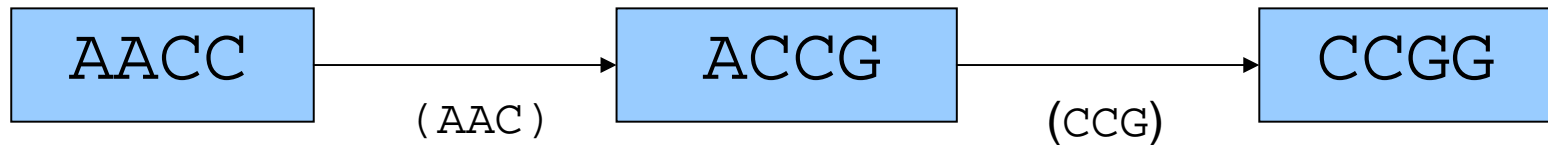


# Eulerian approach

- Break all reads (length  $L$ ) into  $(L-k+1)$   $k$ -mers
  - $L=36, k=31$  gives 6  $k$ -mers per read
- Construct a *de Bruijn* graph (DBG)
  - Nodes = one for each unique  $k$ -mer
  - Edges =  $k-1$  exact overlap between two nodes
- Graph simplification
  - Merge chains, remove bubbles and tips
- Find a Eulerian path through the graph
  - Linear time algorithm, unlike Hamiltonian

# DBG : simple

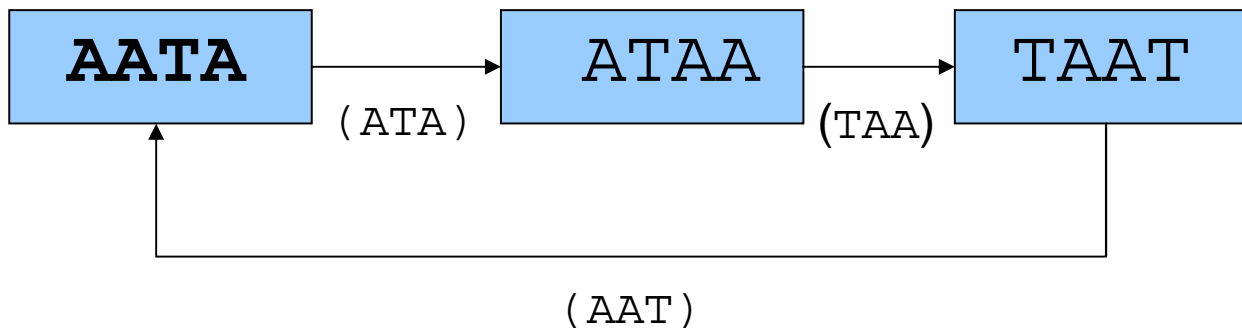
- Sequence
  - AACCGG
- K-mers (k=4)
  - AACC ACCG CCGG
- Graph





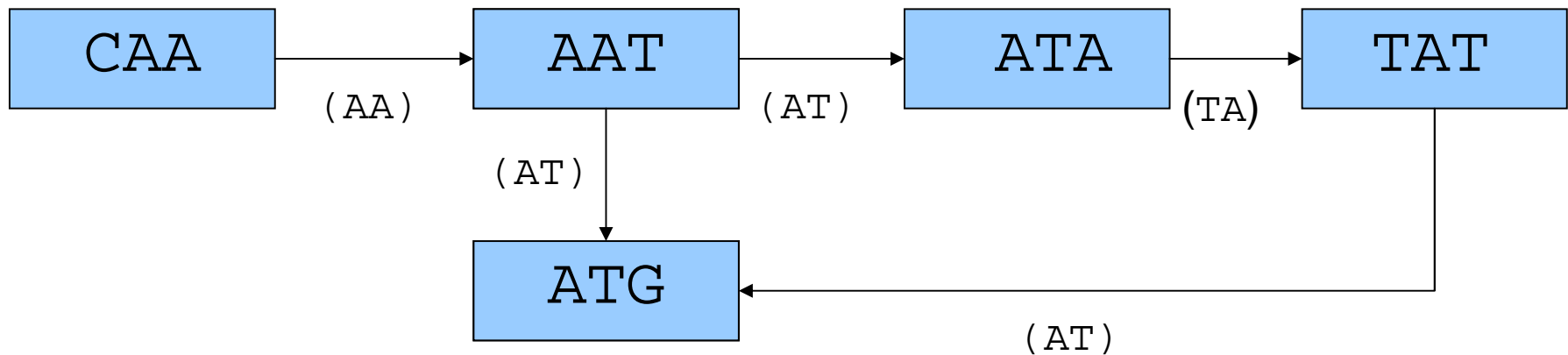
# DBG : repeated k-mer

- Sequence
  - AATAATA
- K-mers (k=4)
  - AATA ATAA TAAT AATA (repeat)
- Graph



# DBG: alternate paths

- Sequence
  - CAATATG
- K-mers (k=3)
  - CAA AAT ATA TAT ATG
- Graph



# DBG: graph simplification

- Remove tips or spurs
  - Dead ends in graph due to errors at read end
- Collapse bubbles
  - Errors in middle of reads
  - But could be true SNPs or diploidity
- Remove low coverage paths
  - Possible contamination
- Makes final Eulerian path easier
  - And hopefully more accurate contigs

# DBG : Software

- Velvet
  - Fast, relatively easy to use, multi-threaded
- AllPaths-LG
  - Designed for larger genomes, robust
- AbySS
  - Runs on cluster to get around RAM issues
- Ray
  - Designed for MPI/SMP clusters

# OLC vs DBG

- **DBG**
  - More sensitive to repeats and read errors
  - Graph converges at repeats of length  $k$
  - One read error introduces  $k$  false nodes
  - Parameters: `kmer_size cov_cutoff ...`
- **OLC**
  - Less sensitive to repeats and read errors
  - Graph construction more demanding
  - Doesn't scale to voluminous short reads
  - Parameters: `minOverlapLen %id ...`

# Assembly metrics

- Number of contigs/scaffolds
  - Fewer is better, one is ideal
- Contig sizes
  - Maximum, average, median, “N50” (next slide)
- Total size
  - Should be close to expected genome size
  - Repeats may only be counted once
- Number of “N”s
  - N is the ambiguous base, fewer is better



# The “N50” metric

- The N50 of a set of contigs is the size of the largest contig for which half the total size is contained in that contigs and those larger.
  - The weighted median contig size
- Example:
  - 7 contigs totalling 20 units: 7, 4, 3, 2, 2, 1, 1
  - N50 is 4, as  $7+4=11$ , which is  $> 50\%$  of 20
- Warning!
  - Joining contigs can increase N50 eg.  $7+4=11$
  - Higher N50 may mean more mis-assemblies

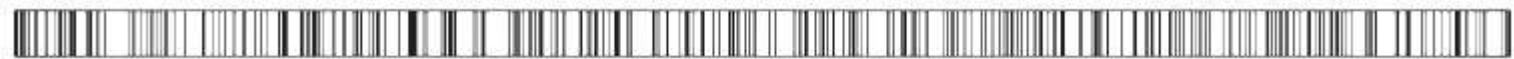
# Scaffolding : method

- Scaffolding algorithm
  - constraint-based optimization problem
- Most assemblers include a scaffolding module
  - Velvet, Arachne, COBOG, AbySS
- Standalone scaffolder: Bambus
  - Part of AMOS package
  - Can handle various types of constraints
  - Uses some heuristics to find solutions



# Optical mapping : overview

- A restriction digest map on a genome scale!
  - OpGen USA (Prok), Schwartz Lab UWM (Euk)
- Choose suitable enzyme restriction site
  - eg. XbaI8 : AACGTT
- Get back a map of all locations of AACGTT
  - Accurate to about 200bp



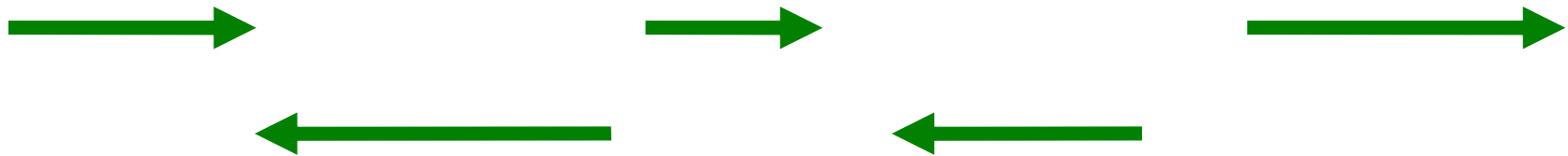
- Align contigs/scaffolds to optical map
  - Use MapSolver or SOMA software

# Optical Mapping: example

- Optical map



- Mapped contigs



- Unmapped contigs



- Need good number of sites to be mappable

# Optical mapping: benefits

- Gives global overview of molecule
  - Aids in genome finishing
- Validates correctness of assembly
  - Identifies mis-assemblies
  - *eg. M.avium* paratb. K10 - found inversion
- Becoming routine for bacterial genomes
  - Cost US\$3000
- Can do 2+ optical maps of same genome
  - More mappability

# Genome finishing : aims

- Produce a single “closed” DNA sequence
  - No gaps or ambiguous bases (only A,G,T,C)
  - No true contigs excluded
- Possible?
  - Yes, for bacteria and virus
  - Troublesome, for larger genomes
- Necessary?
  - Unfinished draft genomes still very useful
  - Advantage is simpler analysis, global structure

# Future trends

- Current reads are “single” or “paired”
  - Relative orientation known eg.  $\rightarrow \dots \leftarrow$
  - Known distance apart eg.  $200 \pm 50$  bp
- Third generation sequencing will change this
  - Strobe reads (PacBio)
  - 3000bp reads interspersed with gap jumps
  - Longer reads, a return to OLC approach?
- Who knows what else!
  - New algorithmic challenges & error models

# Velvet : run through

EMBL-EBI



- Get your reads in suitable format
  - Typically interleaved .fastq or .fasta
- Hash your reads
  - Use “velveth” and choose “k” parameter
- Assemble the hashed reads
  - Use “velvetg” (parameters optional)
- Examine the output
  - Contigs and graph information

# Velvet : read file formats

- Illumina reads are supplied as “fastq”

```
@HWUSI-EAS100R:6:73:941:1273
AGTCGCTTTAGAGTATCTTAGATTTTTCTCCTATGAGGAG
+HWUSI-EAS100R:6:73:941:1273
hhhgggggfdba[[^_Z_ZYXWWWPQQQRNOOHGFBBBBBB
```

- Four lines per read
  1. '@' and unique sequence identifier (id)
  2. Read sequence
  3. '+' with optional duplication of id
  4. Read quality (ASCII encoded)

# Velvet: k-mer size

- Need to choose a “k” the k-mer size
  - Must be odd (avoids palindrome issues)
  - Must be less than or equal to read length
- Small “k”
  - Graph can be overly connected, no clear path
  - More divergence and ambiguity
- Large “k”
  - Less connectivity, more specificity
  - Smaller graph, less RAM, runs faster



# Velvet: hash (index) the reads

```
% ls
```

```
reads.fastq
```

```
% velveth outdir 31 -short -fastq reads.fastq
```

```
Reading FastQ file reads.fastq;
```

```
Inputting sequence 100000 / 142858
```

```
Done inputting sequences
```

```
% ls outdir
```

```
Log Roadmaps Sequences
```



# Velvet: assembly



```
% velvetg outdir  
    -exp_cov auto  
    -cov_cutoff auto
```

```
Writing contigs into 31/contigs.fa...  
Writing into stats file 31/stats.txt...  
Writing into graph file 31/LastGraph...  
Estimated Coverage = 5.894281  
Estimated Coverage cutoff = 2.947140  
Final graph has 436 nodes and n50 of 274, max  
1061, total 92628, using 111913/142858 reads
```

```
% ls outdir
```

```
Graph2  LastGraph  Log PreGraph  Roadmaps  
Sequences  contigs.fa  stats.txt
```

# Velvet : output files

- `contigs.fa`
  - The assembled contigs in .fasta format
- `stats.txt`
  - Intermediate information about each contig
    - Average coverage (in k-mers)
    - Length (in k-mers)
    - How many edges went in/out of this contig node
- `LastGraph`
  - Detailed representation of the de Bruijn graph

# References

J. Miller, S. Koren, G. Sutton (2010)  
*Assembly algorithms for NGS data*  
**Genomics** 95 pp315-327.

M. Pop (2009)  
*Genome assembly reborn: recent  
computational challenges*  
**Briefings in Bioinformatics** 10:4 pp354-366.

E. W. Myers (2005)  
*The fragment assembly string graph*  
**Bioinformatics** 21:2 pp79-85.

# Contact



*Victorian  
Bioinformatics  
Consortium*

- Email
  - [torsten.seemann @ monash.edu](mailto:torsten.seemann@monash.edu)
- GoogleChat
  - [torsten.seemann](#)
- Web
  - [bioinformatics.net.au](http://bioinformatics.net.au)
- Twitter
  - [@torstenseemann](#)
  - [#IMBWS11](#)