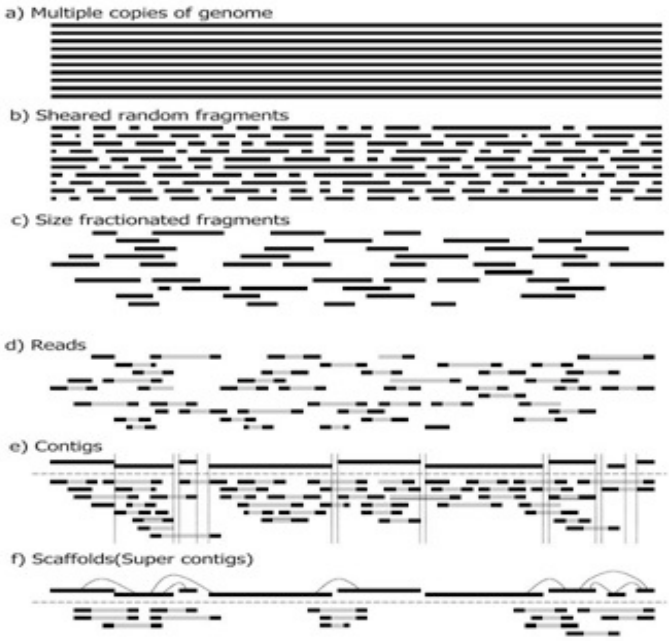
Next generation sequencing: *de novo* assembly

Laurent Falquet, Vital-IT
Helsinki, June 4, 2010

Overview

- What is *de novo* assembly?
- Methods
  - Greedy
  - OLC
  - de Bruijn
- Tools
- Issues
  - File formats
  - Paired-end *vs* mate-pairs
- Visualization
- Discussion

Ultra High Throughput Sequencing (WGS)



- http://www.k.u-tokyo.ac.jp/pros-e/person/shinichi_morishita/shinichi_morishita.htm

Ultra High Throughput Sequencing and Genome Assembly: a Simple Jigsaw Puzzle?

- Yes, but you must deal with
  - Millions of pieces
  - Lots of malformed pieces
  - Often missing pieces
  - Pieces mixed from another puzzle
  - Lots of identical blue sky pieces…
  - If *de novo* you…

Genome assembly, deep blue…

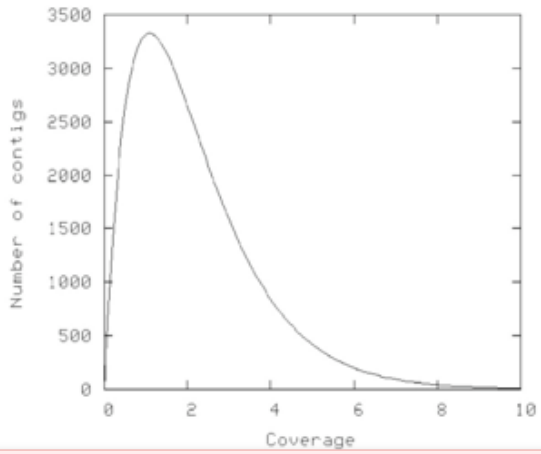
…don't even know the final picture…

Limitations of the techniques

- Sequencing errors (all methods)
- Roche454 long (>8) mononucleotide repeats
- Illumina and SOLiD, very short reads (20-75bp)
- Missing data (sampling/coverage bias)?

Minimal coverage?

- Mathematically, this phenomenon was modeled by Eric Lander and Michael Waterman in 1988. They examined the correlation between the oversampling of the genome (also called coverage) and the number of contiguous pieces of DNA (commonly called contigs) that can be re-constructed by an idealized assembly program.



$P(y) = (C^y * e^{-C}) / y!$

C = coverage
y = nr of time a base is sequenced

If y = 0 (not sequenced) $P(0) = e^{-8}$
Size of gaps = $10^6 * e^{-8} = 300$
Nr of gaps = $16000 * e^{-8} = 4.8$
(read length = 500)

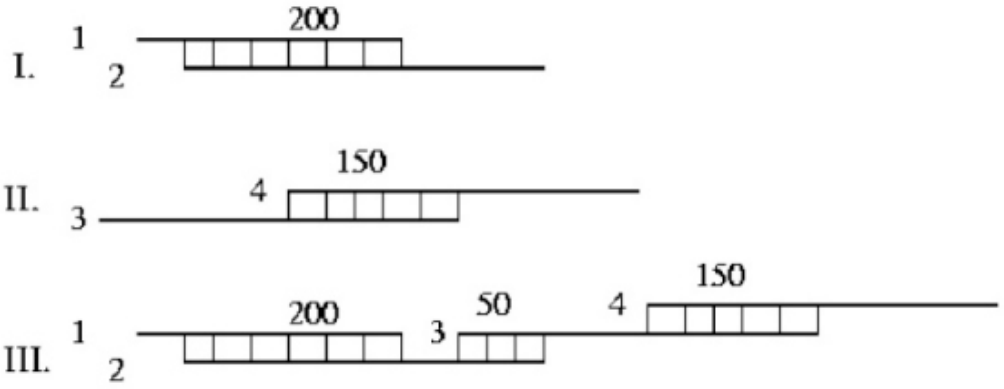
Plot of the Lander-Waterman equation for a genome of 1Mbp (mega base pairs = 1,000,000 base pairs). Between 8 and 10-fold coverage the model predicts that most of the genome will be assembled into a small number of contigs (approx. 5 for a 1Mbp genome).

Algorithms for assembly

- Greedy
  - Phrap, Cap3, TIGR assembler, …
- Overlap-layout-consensus
  - Celera wgs Assembler, Phusion, MIRA3, Edena …
- Eulerian path
  - Euler-SR, Velvet, ABySS, SOAPdenovo, VCAKE, …
- Align-layout-consensus (mapping)
  - Projector2, Mozaik, MAQ, Bowite, BWA, ELAND, MUMmer, …
- Bac-by-Bac
  - Atlas, …

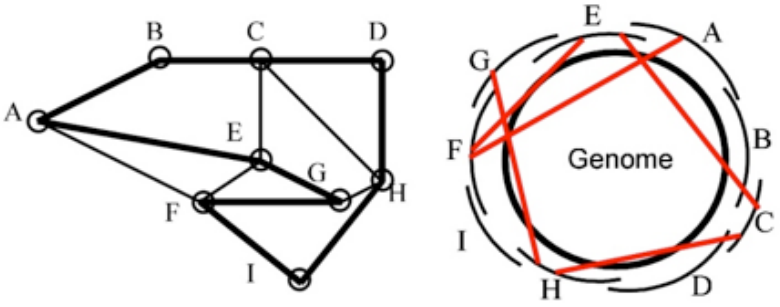
Greedy

- Greedy assemblers - The first assembly programs followed a simple but effective strategy in which the assembler greedily joins together the reads that are most similar to each other.
- An example is shown below, where the assembler joins, in order, reads 1 and 2 (overlap = 200 bp), then reads 3 and 4 (overlap = 150 bp), then reads 2 and 3 (overlap = 50 bp) thereby creating a single contig from the four reads provided in the input. One disadvantage of the simple greedy approach is that because local information is considered at each step, the assembler can be easily confused by complex repeats, leading to mis-assemblies.



Overlap-layout-consensus

- Overlap-layout-consensus - The relationships between the reads provided to an assembler can be represented as a graph, where the nodes represent each of the reads and an edge connects two nodes if the corresponding reads overlap. The assembly problem thus becomes the problem of identifying a path through the graph that contains all the nodes - a Hamiltonian path (Figure below). This formulation allows researchers to use techniques developed in the field of graph theory in order to solve the assembly problem.
- An assembler following this paradigm starts with an **overlap stage** during which all overlaps between the reads are computed and the graph structure is computed. In a **layout stage**, the graph is simplified by removing redundant information. Graph algorithms are then used to determine a layout (relative placement) of the reads along the genome. In a final **consensus stage**, the assembler builds an alignment of all the reads covering the genome and infers, as a consensus of the aligned reads, the original sequence of the genome being assembled.



Overlap graph for a bacterial genome. The thick edges in the picture on the left (a Hamiltonian cycle) correspond to the correct layout of the reads along the genome (figure on the right). The remaining edges represent false overlaps induced by repeats (exemplified by the red lines)

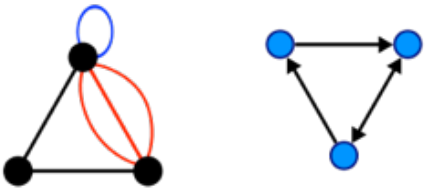
Leonhard Euler
1707 - 1783

- Swiss mathematician
- Euler's identity, the most famous formula!

$$e^{i\pi} + 1 = 0$$

- Graph theory
  - In 1736, Euler solved the problem known as **the Seven Bridges of Königsberg**. The city of Königsberg, Prussia was set on the Pregel River, and included two large islands which were connected to each other and the mainland by seven bridges.
  - The problem is to decide whether it is possible to follow a path that crosses each bridge exactly once and returns to the starting point. It is not: there is no Eulerian circuit. This solution is considered to be the first theorem of graph theory, specifically of planar graph theory.

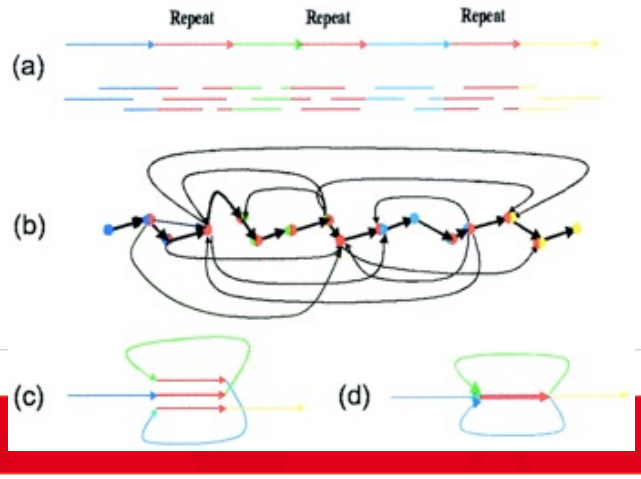
Graph theory



- A graph refers to a collection of **vertices** (or '**nodes**') and a collection of **edges** (or '**vectors**') that connect pairs of vertices.
- A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another (digraph).

Eulerian path

- **Eulerian path** approaches are based on early attempts to sequence genomes through a technique called sequencing by hybridization. In this technique, instead of generating a set of reads, scientists identified all strings of length k (k-mers) contained in the original genome.
- This approach, also based on a graph-theoretic model, breaks up each read into a collection of overlapping k-mers. Each k-mer is represented in a graph as an edge connecting two nodes corresponding to its k-1 bp prefix and suffix respectively. It is easy to see that, in the graph containing the information obtained from all the reads, a solution to the assembly problem corresponds to a path in the graph that uses all the edges - an Eulerian path.
- One advantage of the Eulerian approach is that repeats are immediately recognizable while in an overlap graph they are more difficult to identify.



Eulerian *vs* Hamiltonian path ?

- Both definitions are very similar:
  - a **Hamiltonian** path visits every vertex exactly once.
  - an **Eulerian** path visits every edge exactly once.
  - a **de Bruijn** graph is Eulerian **and** Hamiltonian.

- In practice, however, it is much more difficult to construct a Hamiltonian path or determine whether a graph is Hamiltonian, as that problem is NP-complete.

http://en.wikipedia.org/wiki/Hamiltonian_path
http://en.wikipedia.org/wiki/Eulerian_path

Limitations of the sequence

- Repeats
  - transposases, IS-elements, retroviruses, duplications, etc.
- Polymorphisms
  - SNPs, CNV, multiploid, sample mixture, etc.
- Sequence bias
  - %GC

Repeats are a major issue for all assemblers

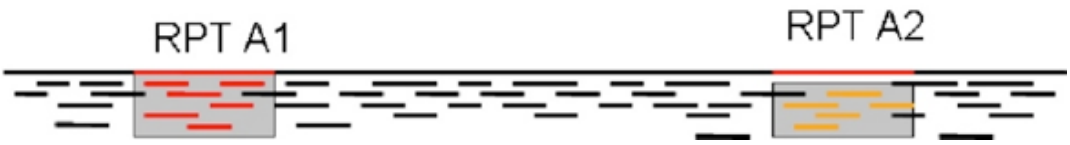


Figure top. Two copies of a repeat along a genome. The reads colored in red and those colored in yellow appear identical to the assembly program.

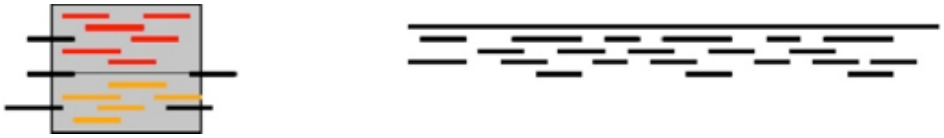


Figure bottom. Genome mis-assembled due to a repeat. The assembly program incorrectly combined the reads from the two copies of the repeat leading to the creation of two separate contigs.

Helping the assembly with linked reads

- When the **distance** and the **orientation** between 2 reads is known
- First proposed by
  - Edwards, A; Caskey, T (1991). "Closure strategies for random DNA sequencing". Methods: A Companion to Methods in Enzymology 3 (1): 41–47. doi:10.1016/S1046-2023(05)80162-8.
- Also called
  - Double-barreled
  - Mate-pairs
  - Paired-ends

Mate pairs validation example

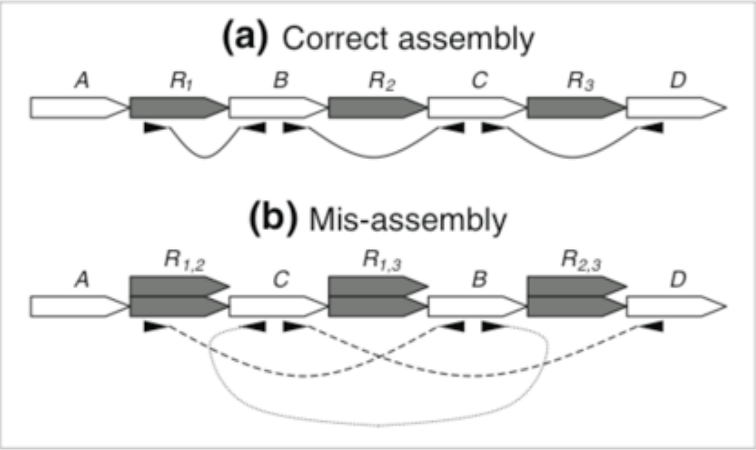
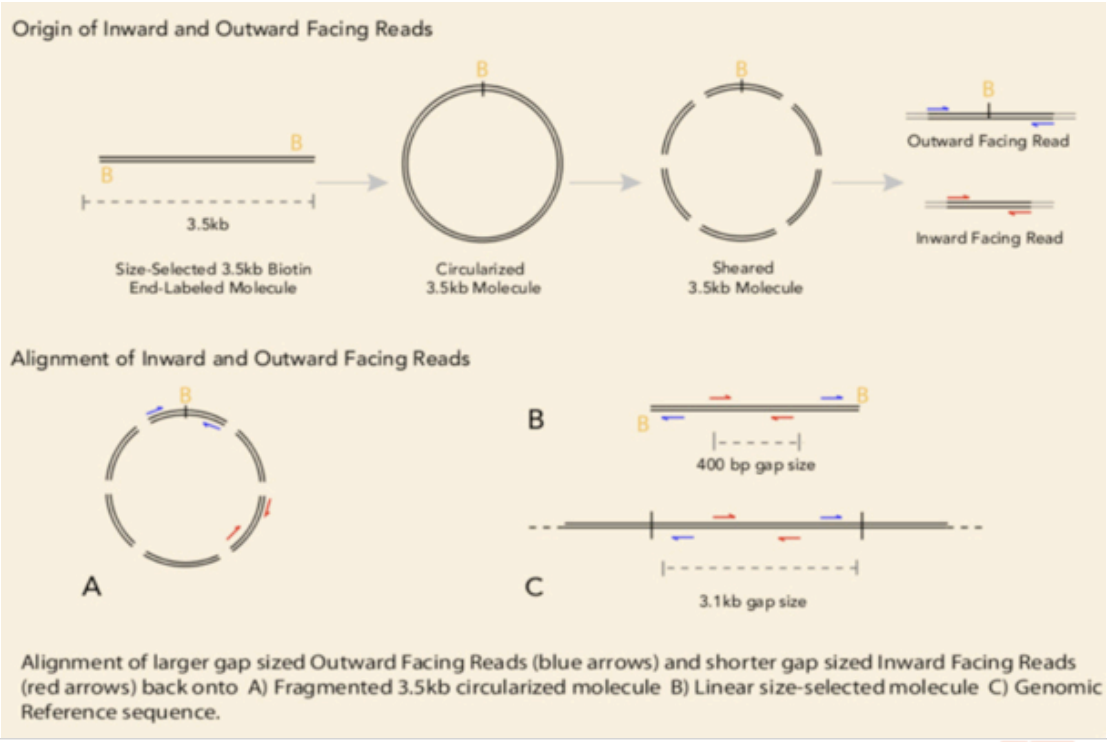


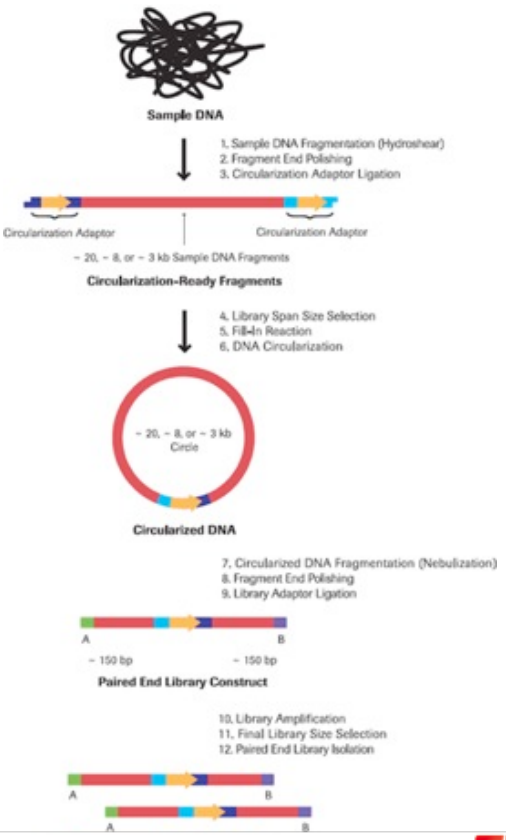
Figure 3
Mate-pair signatures for rearrangement style mis-assemblies. **(a)** Three copy repeat *R*, with interspersed unique sequences *B* and *C*, shown with properly sized and oriented mates. **(b)** Mis-assembled repeat shown with mis-oriented and expanded mate-pairs. The mis-assembly is caused by co-assembled reads from different repeat copies, illustrated by the stacked repeat blocks.

- 4 main criterias
  - Mates too close to each other
  - Mates too far from each other
  - Mates with same orientation
  - Mates pointing away from each other
- Other criterias
  - Mates not present on the assembly (singletons)
  - Mates on different contigs

Illumina mate-pair (long paired-end by circularization)



454 paired-end and mate pairs



Paired-end summary

- Illumina: paired-end 200-500bp
- Illumina: mate-pair 3Kbp
- 454: mate pair 140bp, insert any size (3Kbp, 8Kbp, 20Kbp)
- SOLiD: paired-end 50+25bp, insert 200-600bp
- SOLiD: mate pair 50+25bp, insert 600bp-10Kbp

Tools for *de novo* sequence assembly (non-exhaustive list)

- ABySS
- Velvet
- SOAPdenovo
- Euler
- Edena
- Newbler
- MIRA
- WGS(Celera)
- Amos
- Phusion
- Phrap
- Cap3

- Mummer
- SAMtools
- BreakDancer
- Eagleview
- Hawkeye
- Tablet

Software issues

- File formats jungle
  - Each software has its own internal formats, few comply with the emerging standards
- Parameters tuning
  - Several parameters must be tuned, in particular the Kmer
- Large memory requirements
  - Some software might require hundreds of Gbytes
- Often single threads
  - Few of the software are multithreaded
- Unfinished beta software
- Poor visualization

File formats jungle

- **.fasta**
- .qual (phred quality file)
- .fastq
- .sff (454 binary data file, Standard Flowgram Format)
- .srf (sequence read format) plateform independent format
- .txt (Illumina/Solexa files) (FastQ-like)
- .csfasta (SOLiD color space)

- Paired-end
  - 2 files
  - crossbow style

- Output
  - fasta
  - SAM/BAM
  - afg
  - Other files (stats)

FASTA example

```
>contig_6 length=320 nReads=87 !529472 ] !2294037  ]
TAACGGTAGGCTTTTTTGACCGCTTCATCGTCGGGTGGTTCAACATTTTCTAATTGATAT
GGGATGCCTAAATTTTTCCACTTATACACGCCGAGTTGGTGATAGGGTAAGATTTCAAAT
TTTTCAACGTTATCAAGAGAATTAATAAATTCTCCAAGTTGAATGAGATCTTCTTTATCA
TCTGAGATACCTGGCACTAGGACGTGACGAATCCATACAGGTTGTTTCATATCTGACAAT
TTACGGGCAAATTTGAGTATATGTGTATTGGGTTTGCCTGTTAATTGAATATGTTTTTCA
TTATTAATATGCTTTATGTC
>contig_7 length=140 nReads=45 60537 ] 1378182  ]
TCGTTTTATAACTGAAGAAGAACTATCAAAATATATGAACGCCGATCAAAAACAACCTGA
AGAACCTGCAGCTCAAGAAATTAAACAACATCAAAATGTCGATAACCCGCGTGGTATTGA
ACAATTTAATACACACAATA
>contig_8 length=212 nReads=59 1604937 ] 1907084  ]
ATAAGTTGAATCTGTTTGATTAGCTTGAGTGATGGCATTACCATTCGACTGATGGTTAAA
ACCTTGGTCTACTTGATTATTTTCTATAGTTGCAGCTGAAGCCTCGTGATGTGATGTAAG
AAATAAAGCAGAAGTAGTGATAGTTGCGCCGATTAAGTATTTGATAGAATGATGAGTCAA
AAAAATCTCCCCTTGAATATATTTATTTATAC
```


Quality file example

```
>contig_6 base quality
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 21 24 23 23 22 22 22 22 30 31 20 20 20 20
20 20 30 30 31 30 33 33 34 34 34 34 31 33 31 34 34 30 30 25 25 25 26 30 30
33 37 30 33 36 35 23 23 24 24 24 24 30 30 30 30 30 30 30 30 30 33 33 33 33
30 30 33 33 33 30 33 33 29 29 29 29 33 34 33 33 21 21 21 21 23 23 23 23 23
23 24 24 26 26 26 26 26 26 26 40 33 30 30 30 30 33 33 33 33 33 30 33 33 33
33 33 33 33 31 31 31 31 34 37 40 40 45 37 52 52 52 52 52 52 55 55 59 64 65
68 58 57 49 49 49 49 49 49 49 56 60 53 60 53 49 49 49 49 49 34 45 45 45 45
30 25 25 25 28 30 45 45 49 49 49 49 49 70 60 60 59 59 53 56 53 53 53 55 53
53 60 45 49 49 42 42 42 45 45 36 33 34 49 46 46 46 54 54 59 53 49 49 42 41
43 40 44 40 49 49 49 54 58 53 52 57 51 51 41 39 15 15 35 35
>contig_7 base quality
…
```

Example of FASTQ Illumina 1.5

```
@C3PO_0001:2:1:17:1499#0/1
TGAATTCATTGACCATAACAATCATATGCATGATGCAAATTATAATATCATTTTTAGTGACGTCGTGAATCGTTT
+C3PO_0001:2:1:17:1499#0/1
abaaaaaaaaaaa`a`aa_aaaaaaaaaaaaaaaa_a__aaa`aaaaa^aaaaa`a]^`a__YZYZ^`NJDJ\_Z
@C3PO_0001:2:1:17:1291#0/1
TGTTTGAGCAAATGATTCATAATAATGTATTTCAATATTTTTAGGAATATCTCCCAATATTGCGCGTGCTGAATT
+C3PO_0001:2:1:17:1291#0/1
a`_`_\a_aaaa_a^Z^^a[a^aa]a_^_a_``aa__`aa`X^X^^`aa_\_]VR`\a_]W\_`_a]a]][\RZV
@C3PO_0001:2:2:1452:1316#0/1
GTCCATCCGCAGCAGCGAATTTTTGACGTCCCCCCCCCGAANGGANGNGANNNNGNNGNNNTNTNNAAANGNNNNN
+C3PO_0001:2:2:1452:1316#0/1
_U__a\__`]_`ZP\\_Z^[]aa^a_]XNBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
…
```


Warning: various FASTQ formats…

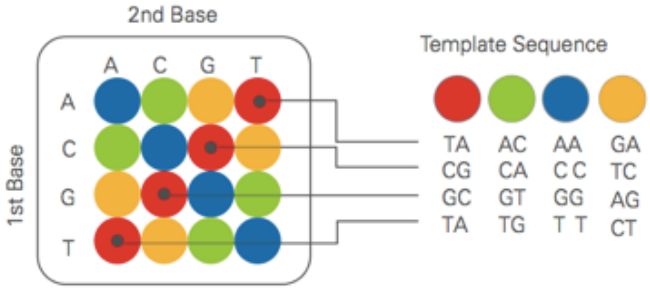
```
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.....................
...............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.....................
................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ.....................
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
|                         |    |        |                              |                     |
33                                     59   64       73                               104
126
S - Sanger        Phred+33,  raw reads typically (0, 40)
X - Solexa        Solexa+64, raw reads typically (-5, 40)
I - Illumina 1.3+ Phred+64,  raw reads typically (0, 40)
J - Illumina 1.5+ Phred+64,  raw reads typically (3, 40) with 0 and 1=unused, 2=Read Segment
Quality Control Indicator (bold)
```

http://en.wikipedia.org/wiki/FASTQ_format

SOLiD color space FASTA format

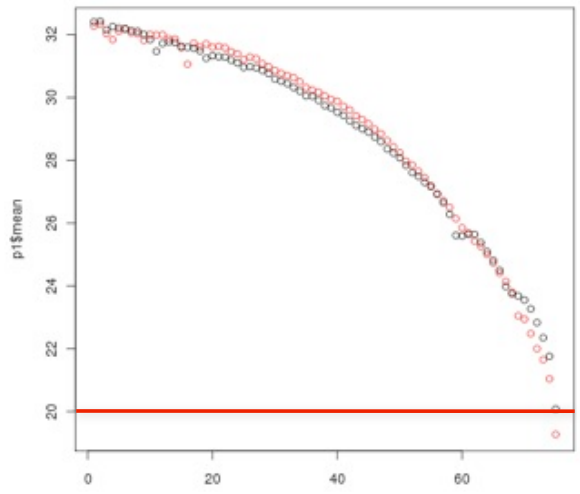
```
>1_51_64_F3
T103010312303332332033330000211222 23
>1_51_127_F3
T201032323320313231011010020031031 02
```

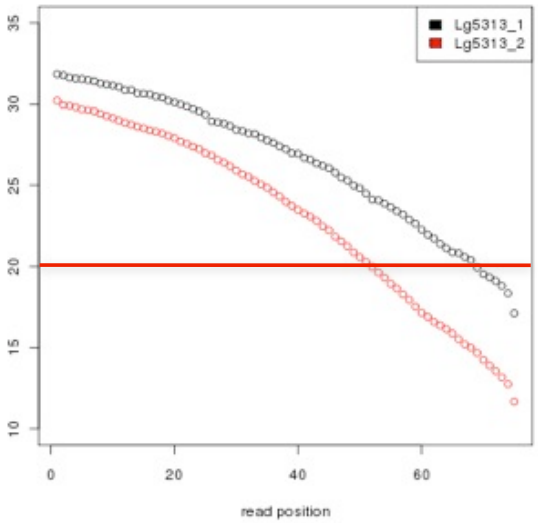
Each number can be replaced according to this table



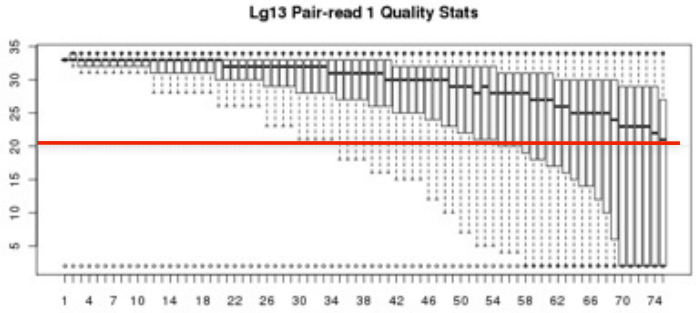
Variability in the quality (mean value per position)

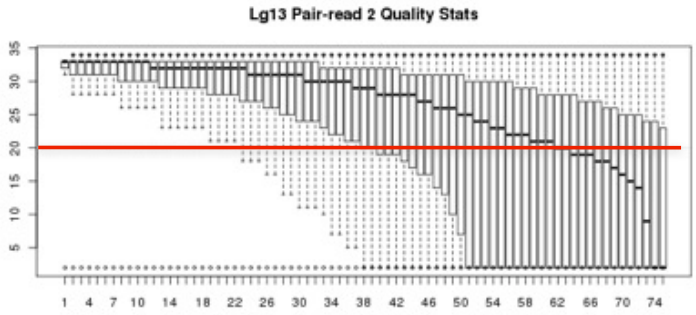
- Good example
- Less good example…





Variability in the quality (boxplot)





Filtering data can help

http://pathogenomics.bham.ac.uk/blog/2009/09/tips-for-de-novo-bacterial-genome-assembly/

- Illumina reads quality decrease with length
  - Trim 3' ends of reads according to quality
  - Remove reads with average low quality
  - If coverage is high, remove orphan reads
- 454 reads
  - Trim 3' ends of reads according to quality
  - Remove reads with average low quality
  - If possible correct for long mononucleotide repeats
- Check contigs by remapping reads

De novo assembly

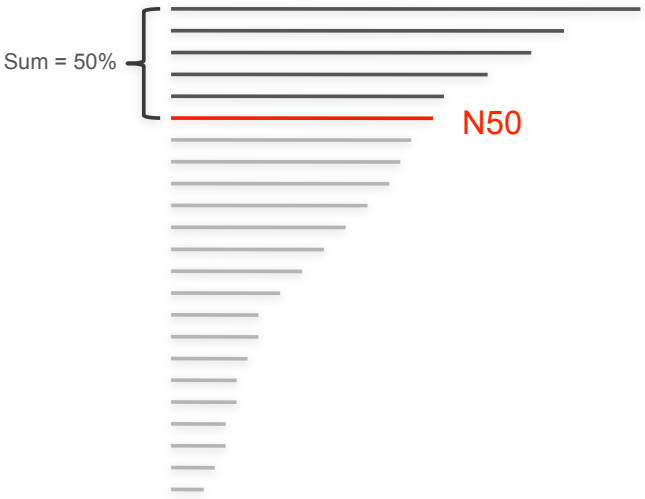
- Current trend: start with small inserts paired-end and add larger inserts sequentially
- Do not mix all reads (454, Illumina, SOLiD, etc…)
- Assemble them separately or sequentially
  - 454 with newbler
  - Illumina with SOAPdenovo, ABySS or Velvet
  - SOLiD with Velvet or ABySS
- Combine assemblies
  - With newbler, SOAPdenovo, CAP3, Phrap, etc…

Assembly quality measurements

- Number of contigs
  - Ideally 1 for a bacterial genome…, but the lower the better
- Contig sizes
  - The larger the better (up to the size of the genome), usually given in maximum, minimum and average lengths.
- Correctness
  - Difficult to assess for a new genome
- N50
  - **The most used quality value for *de novo* assembly**
  - The N50 is the size of the smallest of all the large contigs covering 50% of the genome

N50 what's that?

- Sort the contigs by size
- Sum them starting with the largest until you reach 50% of the estimated genome size
- Last contig added = N50



Velvet for S5

K =	21	23	25	27	29	31
Nr contigs	7012	1906	767	420	325	**252**
Consensus size bp	3103135	2918437	2875773	**2863169**	3619536	2936521
N50	12182	43427	67361	66898	66306	**107440**
Min	41	45	49	53	57	**61**
Max	161171	201664	201396	201389	238872	**369778**

ABySS for S5

K =	21	23	25	27	29	31
Nr contigs	4318	2891	2123	1636	1339	**1113**
Consensus size bp	3220552	3127361	3088161	**3049081**	3078819	3052504
N50	15928	25693	29334	30241	**31596**	29797
Min	21	23	25	27	29	**31**
Max	63797	132812	132816	132992	**132996**	122383

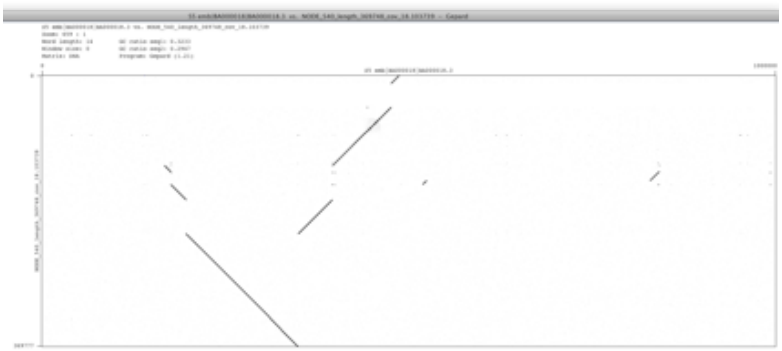
SOAPdenovo for S5

K =	21	23	25	27	29	31
Nr contigs	247	**213**	234	248	280	362
Consensus size bp	2818333	**2825424**	2831502	2833334	2828297	2785031
N50	98956	**99286**	82319	82910	84517	52098
Min	100	100	100	100	100	100
Max	**253458**	252985	181975	182194	182217	141106

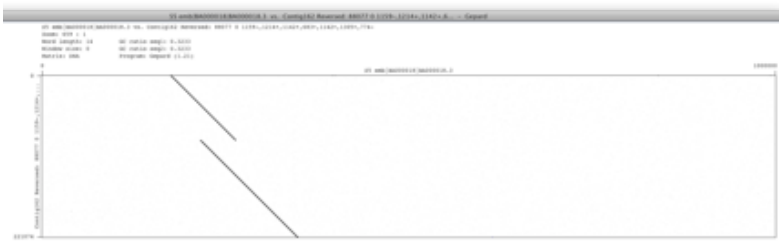
Best scores

K =	Velvet31	ABySS29	SOAPdenovo23
Nr contigs	252	1339	**213**
Consensus size bp	2936521	3078819	**2825424**
N50	**107440**	31596	99286
Min	61	29	**100**
Max	**369778**	132996	252985

However longest contig is not always the best…



- Velvet: 369’778bp



- ABySS: 88’077bp and 132’996bp

Beware of unwanted options…

- E.g., Velvet scaffolding (ON by default in the last version 0.7.55)…

```
>NODE_1931_length_5525_cov_11.776470
TTTTTAAGTGCATGTGTATAATTTTCTACTGGGATAGGATCTGATGTTGCTGAACCTTCA
AATATAGTTATTTCTGGCAATCTTTCCTCTGCATAGTTAAAAGCTTTATTTAAAATTTCA
TCTATGTCTACATATATTTTTGTATACAGTCTCTTACCTAATTGAGTATTTAAATAACAA
TATTCACACATCCCCATGCACCCACTAACTAAAGGTAACTGATAATGTGCGGATGGTTTN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
…
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNTATACATCTGTAGATATCCATTCTCTAGAAACTTCT
ACTCCATTTTCATATGTTATTTGGTAAGAACTTGCCTTATAGCCTGGCATACCATTAATT
TTTACTTTTTGCTCTCCTTCTTTTAAAGCTGGATTATCAATATATTCAACTTCTGGATTA
TATTTTTCGTGTACTTCATTTACTAGTTGATATGTCTTTCCACTTAAAGCCTCTTTATTT
CCGTAAATATTAAATCCTACAGTTCCTCCACCAATATATCCTTCTATATATACTGGAAAA
TCATATGAATTAACAAATTTATAATCTACTACTCCATAAGCTACTGTAGCATCTAAACCT
GGATCTGAATATGAAACTGTTAATGTGTGGTTAGCTCTTTCTGTTGATTTTAAATTAGCA
```


Summary

- Lessons from the *de novo* genome assembly
  - Contigs obtained must be verified
  - Repeats are a nightmare in any case
  - Paired-ends help!

© 2009 SIB LF June 4, 2010 Swiss Institute of Bioinformatics

Commands ABySS

```
# loop varying the kmer value
foreach k (64 59 55)
   abyss-pe k=$k n=5 name=abysau_$k in='../saureus_1.fq ../saureus_2.fq'
end

# ABYSS can run in multi-threaded mode or via MPI on a cluster

# warning abyss-pe is a wrapper around ABYSS executable

# by default it is limited to max kmer=64 (need to recompile)
```


Commands Velvet

```
# velvet runs in 2 steps
foreach k (63 59 55 51)
   velveth64 velsau_$k $k -fastq -shortPaired ../saureus_1.fq ../saureus_2.fq;
   velvetg64 velsau_$k -ins_length 600 -ins_length_sd 100 -min_contig_lgth 100
end

# by default it is limited to max kmer=31 (need to change parameter &
   recompile)
```

VelvetOptimiser.pl is an add-on script to optimise the paramaters for Velvet.

You must select the parameter to be optimised, but it is not faster than doing it yourself.

It has the ability to pre-calculate the memory required by Velvet.

Commands SOAPdenovo

```
# SOAPdenovo requires a config file
foreach k (31 29 27 25 23)
   SOAPdenovo all -s soap.config -K $k -d 2 -o soapsau_$k -p 2
End

#more soap.config
max_rd_len=76
[LIB]
avg_ins=600
reverse_seq=0
# for scaffolding
asm_flags=3
rank=1
#fastq file for read 1
q1=/home/saureus_1.fq
#fastq file for read 2 always follows fastq file for read 1
q2=/home/saureus_2.fq

# SOAPdenovo can run faster in multithreaded mode (-p)
# SOAPdenovo can combine several librarie sequentially
```


The practicals

- http://edu.isb-sib.ch

- select «workshops» on the left menu
- select «Helsinki NGS workshop» at the top
- Enrol yourself with the key «EMBRACE2010»

- Login to hippu server, then follow the instructions of the exercises

Thank You