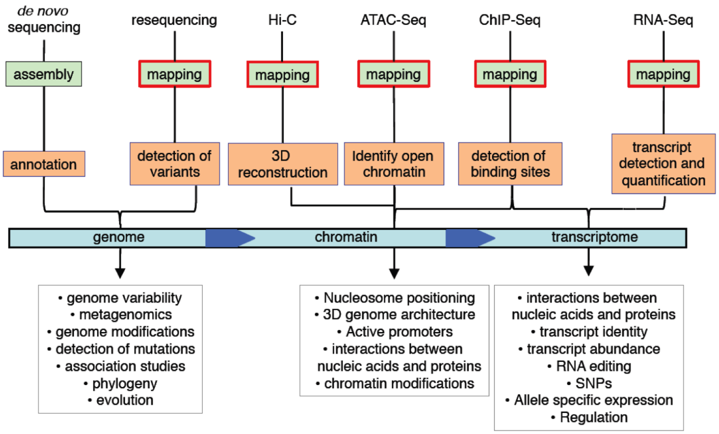
Genome-scale technologies 2 / Algorithmic and statistical aspects of DNA sequencing

Sequencing read mapping

Ewa Szczurek
szczurek@mimuw.edu.pl

Instytut Informatyki
Uniwersytet Warszawski

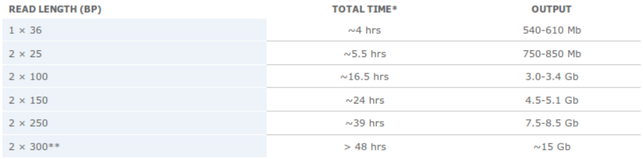
Read mapping



de novo sequencing
resequencing
Hi-C
ATAC-Seq
ChIP-Seq
RNA-Seq
assembly
mapping
mapping
mapping
mapping
mapping
annotation
detection of variants
3D reconstruction
Identify open chromatin
detection of binding sites
transcript detection and quantification
genome
chromatin
transcriptome
• genome variability • metagenomics • genome modifications • detection of mutations • association studies • phylogeny • evolution
• Nucleosome positioning • 3D genome architecture • Active promoters • interactions between nucleic acids and proteins • chromatin modifications
• interactions between nucleic acids and proteins • transcript identity • transcript abundance • RNA editing • SNPs • Allele specific expression • Regulation

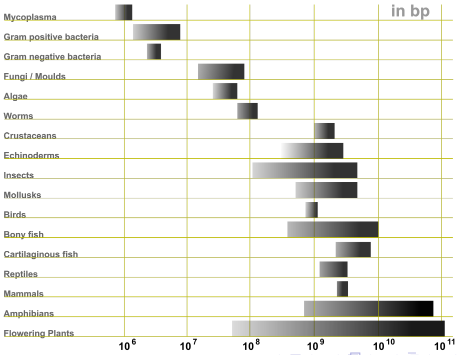
Read mapping data

- sequencing data



READ LENGTH (BP)	TOTAL TIME*	OUTPUT
1 × 36	~4 hrs	540-610 Mb
2 × 25	~5.5 hrs	750-850 Mb
2 × 100	~16.5 hrs	3.0-3.4 Gb
2 × 150	~24 hrs	4.5-5.1 Gb
2 × 250	~39 hrs	7.5-8.5 Gb
2 × 300**	> 48 hrs	~15 Gb

- reference genome – long sequence over alphabet $\{A, C, G, T\}$



The FastQ format

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
```

Consecutive lines:

- Identifyier
- Sequence
- Identifyier
- Quality scores

Quality report

Phred score

- let P be the base-calling error probability,
- $Q = -10\log_{10}(P) \Rightarrow P = 10^{-Q/10}$.

For example

- $Q = 10 \Rightarrow P = 0.1$
- $Q = 20 \Rightarrow P = 0.01$
- $Q = 30 \Rightarrow P = 0.001$
- $Q = 40 \Rightarrow P = 0.0001$

Read mapping

Problem

For each read find a corresponding genome fragment.

Read mapping

Problem

For each read find a corresponding genome fragment.

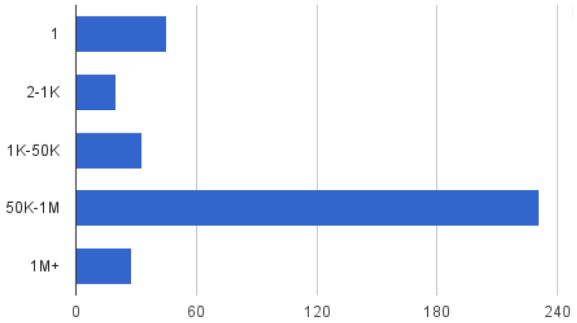
Kind of approximate string matching problem

- models
  - Hamming distance: (the minimum number of substitutions required to transform one string into the other)
  - edit distance (the minimum number of operations required to transform one string into the other: insertion, deletion, substitution of one character)
  - alignment score with gap penalty
- match should be unique (or have assigned quality)
- fixed length of patterns (25-250bp)
- huge amount of data
- repetitions and unknown fragments in a reference text

Technical issues with indexing

Ambiguous bases

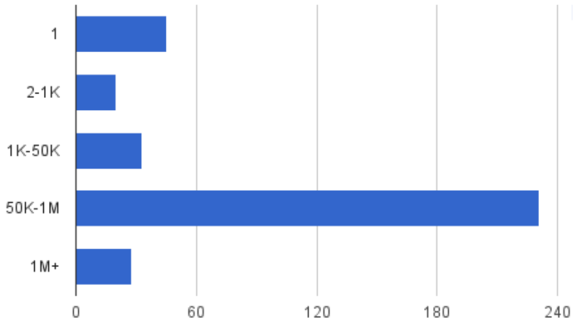
7% of human genome sequence is unknown



Technical issues with indexing

Ambiguous bases

7% of human genome sequence is unknown



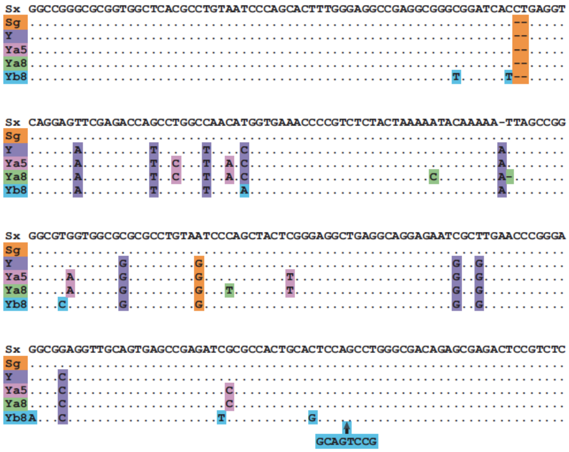
Possible approaches

- skip long ambiguous fragments
- replace short ambiguous fragments with random nucleotides

Technical issues with indexing

Repetitive elements

Alu repetitive element has $\sim$ 1 million occurrences in human genome ($\sim$ 10%)



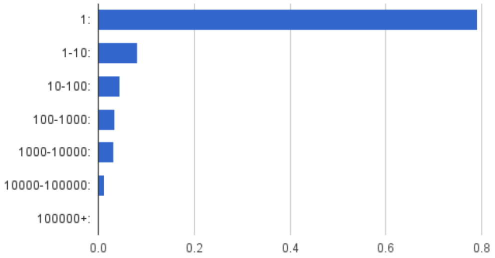
Sx GGCCGGGCGCGGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGGCCGAGGCGGGCGGATCACCTGAGGT
Sx CAGGAGTTCGAGACCAGCCTGGCCAACATGGTGAAACCCCGTCTCTACTAAAAATACAAAAA-TTAGCCGG
Sx GGCGTGGTGGCGCGCGCCTGTAATCCCAGCTACTCGGGAGGCTGAGGCAGGAGAATCGCTTGAACCCGGGA
Sx GGCGGAGGTTGCAGTGAGCCGAGATCGCGCCACTGCACTCCAGCCTGGGCGACAGAGCGAGACTCCGTCTC
Sg
Y
Ya5
Ya8
Yb8
GCAGTCCG

Alignment of Alu-subfamily consensus sequences.

Technical issues with indexing

low complexity regions

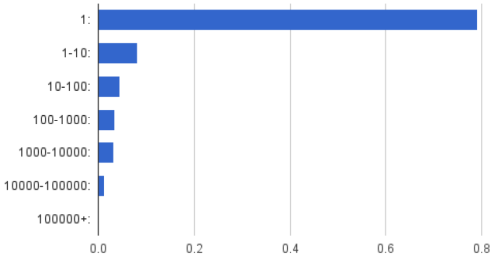
number of occurences of 20bp-long substrings in human genome



Technical issues with indexing

low complexity regions

number of occurences of 20bp-long substrings in human genome



$> 3Mbp$ ($\sim 0.1\%$ of human genome) consists of long sequences of the form

- aaaa. . . (tttt. . .)
- caca. . . (gtgt. . .)

Technical issues with indexing

repetitions

Most popular substrings of length 20 in human genome GRCh37

Substring	Occurrences
TTTTTTTTTTTTTTTTTTTT	451296
AAAAAAAAAAAAAAAAAAAA	447468
GTGTGTGTGTGTGTGTGTGT	246066
ACACACACACACACACACAC	243148
TGTGTGTGTGTGTGTGTGTG	241608
CACACACACACACACACACA	238826
CTCCCAAAGTGCTGGGATTA	170026
TAATCCCAGCACTTTGGGAG	169758
CCTCCCAAAGTGCTGGGATT	166855
AATCCCAGCACTTTGGGAGG	166726

Technical issues with indexing

repetitions

Most popular substrings of length 20 in human genome GRCh37

Substring	Occurrences
TTTTTTTTTTTTTTTTTTTT	451296
AAAAAAAAAAAAAAAAAAAA	447468
GTGTGTGTGTGTGTGTGTGT	246066
ACACACACACACACACACAC	243148
TGTGTGTGTGTGTGTGTGTG	241608
CACACACACACACACACACA	238826
CTCCCAAAGTGCTGGGATTA	170026
TAATCCCAGCACTTTGGGAG	169758
CCTCCCAAAGTGCTGGGATT	166855
AATCCCAGCACTTTGGGAGG	166726

Possible approach

- Mask and skip repetitive fragments

Mapping tools

BFAST, Bowtie, BWA, ELAND, Exonerate, GenomeMapper, GMAP, gnumap, MAQ, MOSAIK, MrFAST, MUMmer, Novocraft, PASS, RMAP, SeqMap, SHRiMP, Slider, SOAP, SSAHA, SOCS, SWIFT, SXOligoSearch, Vmatch, Zoom . . .

Mapping tools

BFAST, Bowtie, BWA, ELAND, Exonerate, GenomeMapper, GMAP, gnumap, MAQ, MOSAIK, MrFAST, MUMmer, Novocraft, PASS, RMAP, SeqMap, SHRiMP, Slider, SOAP, SSAHA, SOCS, SWIFT, SXOligoSearch, Vmatch, Zoom . . .

General schema

1. Build an index of one dataset (genome or reads) allowing effective substring searching.
2. Process the other dataset against the index to find potential mappings.
3. Verify potential mappings.

Main differences between mapping tools

Index type

- hash table
  - q-gram index (all q-mers in a dataset)
  - q-sample index (selected q-mers in a dataset)
- suffix index

Main differences between mapping tools

Index type

- hash table
  - q-gram index (all q-mers in a dataset)
  - q-sample index (selected q-mers in a dataset)
- suffix index

Indexed dataset

- reference genome
- sequenced reads

Main differences between mapping tools

Index type

- hash table
  - q-gram index (all q-mers in a dataset)
  - q-sample index (selected q-mers in a dataset)
- suffix index

Indexed dataset

- reference genome
- sequenced reads

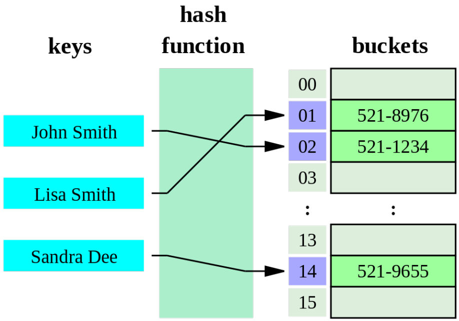
Filtering strategy

Hash tables

- Data structure mapping keys to values.
- A hash function simply converts a string ("key") to an integer ("value")
- The integer is then used as an index in an array, for fast look up.
- Space: $O(n)$, Operations: average $O(1)$, worst case $O(n)$.

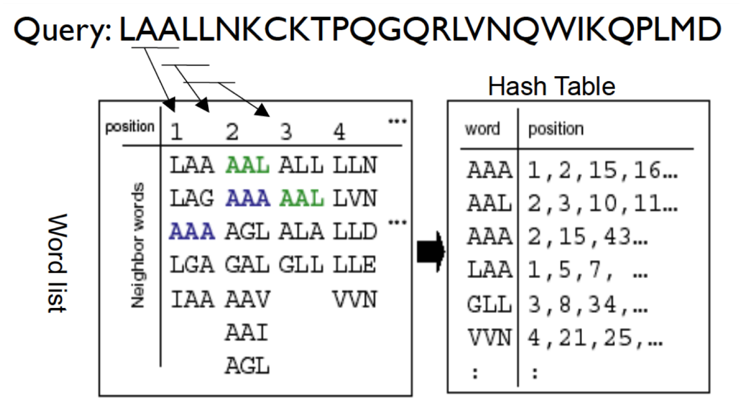
Hash tables

- Data structure mapping keys to values.
- A hash function simply converts a string ("key") to an integer ("value")
- The integer is then used as an index in an array, for fast look up.
- Space: $O(n)$, Operations: average $O(1)$, worst case $O(n)$.



A small phone book as a hash table

Example: 3-gram hash table



position	1	2	3	4	...
	LAA	AAL	ALL	LLN	
	LAG	AAA	AAL	LVN	
	AAA	AGL	ALA	LLD	...
	LGA	GAL	GLL	LLE	
	IAA	AAV		VVN	
		AAI			
		AGL			

word	position
AAA	1,2,15,16...
AAL	2,3,10,11...
AAA	2,15,43...
LAA	1,5,7, ...
GLL	3,8,34,...
VVN	4,21,25,...
:	:

Query word hash table constructed by BLAST.

Mapping with Q-gram index

Mosaik, BFAST, PASS use BLAST-like technique

- build a q-gram index of a genome
- find seeds with index
- extend (sequences of) seeds to full alignment
- spaced seeds are often used

Mapping with Q-gram index

Mosaik, BFAST, PASS use BLAST-like technique

- build a q-gram index of a genome
- find seeds with index
- extend (sequences of) seeds to full alignment
- spaced seeds are often used

Performance

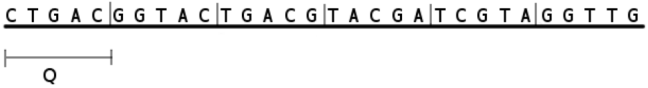
– huge memory required

– relatively slow for low error rates

+ easy to handle higher error rates

Q-sample index

…is a hash table with positions of all non-overlapping text substrings of length q.



Partitioning into exact search

Consider distance $k = 2$, and divide the searched string of length L into 3 equal parts.

pgflastimage

- Three "worst case"placements of mutations
- In each case there is a perfect match with substring of length $L/3$

Partitioning into exact search

Let $A = A_1 \ldots A_{k+s}$, where each A_i is a substring of length $|A|/(k+s)$ and B be two strings such that $d(A,B) \leq k$, where d is Hamming/edit distance. Then at least s substrings $A_{i_1} \ldots A_{i_s}$ appear in B without errors. Moreover:

- when d is Hamming distance, their positions in B are the same as in A,
- when d is edit distance, their relative distances in B cannot differ from those in A by more than k.

Read mapping with q-sample indexes

Eland, MAQ, SeqMap, RMAP:

- Each read is split into substrings
- Use hashing to index read substrings, then scan with reference sequence
- Hits are potential mappings with up to ≤ 2 errors.
- Very fast, but at the cost of accuracy (ungapped)

Read mapping with q-sample indexes

Eland, MAQ, SeqMap, RMAP:

- Each read is split into substrings
- Use hashing to index read substrings, then scan with reference sequence
- Hits are potential mappings with up to ≤ 2 errors.
- Very fast, but at the cost of accuracy (ungapped)

ZOOM uses spaced q-samples.

Read mapping with q-sample indexes

Eland, MAQ, SeqMap, RMAP:

- Each read is split into substrings
- Use hashing to index read substrings, then scan with reference sequence
- Hits are potential mappings with up to ≤ 2 errors.
- Very fast, but at the cost of accuracy (ungapped)

ZOOM uses spaced q-samples.

Performance

+ Read dataset may be split into subsets to fit into available memory.
– Aligning with gaps decreases efficiency.

Suffix indexes

Suffix tree suffixes = paths from root to leaves

- index size: $\geq 10 \cdot |genome|$ bytes
- exact mapping time: $\mathcal{O}(|read| + |occurences|)$

Suffix indexes

Suffix tree	suffixes = paths from root to leaves
	▶ index size: $\geq 10 \cdot \lvert genome \rvert$ bytes
	▶ exact mapping time: $\mathcal{O}(\lvert read \rvert + \lvert occurences \rvert)$
Suffix array	lexicographic order on suffixes
	▶ index size: $\geq 4 \cdot \lvert genome \rvert$ bytes
	▶ exact mapping time: $\mathcal{O}(\lvert read \rvert \cdot \log \lvert genome \rvert + \lvert occurences \rvert)$

Suffix indexes

Suffix tree suffixes = paths from root to leaves

- index size: $\geq 10 \cdot |genome|$ bytes
- exact mapping time: $\mathcal{O}(|read| + |occurences|)$

Suffix array lexicographic order on suffixes

- index size: $\geq 4 \cdot |genome|$ bytes
- exact mapping time: $\mathcal{O}(|read| \cdot \log|genome| + |occurences|)$

FM index self-index based on Burrows-Wheeler transform

- index size: $< 1 \cdot |genome|$ bytes
- exact mapping time: 2-1000$\times$ slower than suffix arrays

Read mapping with FM-index

Bowtie (does not support gapped alignment), BWA, Bowtie 2 (support gaps)

Neighbourhood generation

All words matching a pattern with $0, 1, 2, \ldots$ errors are generated and searched in the index.

Read mapping with FM-index

Bowtie (does not support gapped alignment), BWA, Bowtie 2 (support gaps)

Neighbourhood generation

All words matching a pattern with $0, 1, 2, \ldots$ errors are generated and searched in the index.

Backtracking

Partial results of backward searching are recycled in searching words sharing suffixes.

Read mapping with FM-index

Bowtie (does not support gapped alignment), BWA, Bowtie 2 (support gaps)

Neighbourhood generation

All words matching a pattern with $0, 1, 2, \ldots$ errors are generated and searched in the index.

Backtracking

Partial results of backward searching are recycled in searching words sharing suffixes.

In practice, only close neighbourhood is considered…

…the complexity of neighbourhood generation is exponential with respect to the number of allowed errors, even with backtracking.

Read mapping with FM-index

Bowtie (does not support gapped alignment), BWA, Bowtie 2 (support gaps)

Neighbourhood generation

All words matching a pattern with $0, 1, 2, \ldots$ errors are generated and searched in the index.

Backtracking

Partial results of backward searching are recycled in searching words sharing suffixes.

In practice, only close neighbourhood is considered...

...the complexity of neighbourhood generation is exponential with respect to the number of allowed errors, even with backtracking.

Break for a cartoon BWT search.

Bowtie (Langmead et al. '09)

- Extends basic FM-index search to handle mismatches
- Will find an exact match if it exists.
- Two innovations:
  - quality-aware backtracking
  - double indexing

Bowtie (Langmead et al. '09)

Seed – high-quality part of the read (default: first 28bp)

Bowtie (Langmead et al. '09)

Seed – high-quality part of the read (default: first 28bp)

Search for read occurrences in the genome with

- limited number of errors in the seed,
- limited sum of quality values of mismatched positions in the whole read.

Bowtie (Langmead et al. '09)

Seed – high-quality part of the read (default: first 28bp)

Search for read occurrences in the genome with

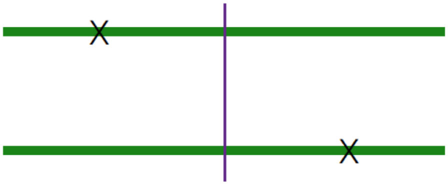
- limited number of errors in the seed,
- limited sum of quality values of mismatched positions in the whole read.

Algorithm

- Genome index is searched with *k*-neighbourhood of the seed of a read.
- Located occurrences are extended to whole read mappings and the quality criterion is checked.

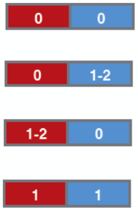
Bowtie – avoiding excessive backtracking

- Using forward (genome sequence) and mirror (reverse index) index
- If mutation in first half of read, then the forward index can walk at least |read| / 2 before backtracking.
- If mutation in second half of read, then the second index can walk at least |read| / 2 before backtracking.
- Try both the forward and reverse indexes; will avoid a lot of backtracking because you will have narrowed the BWT range a lot by the time you start backtracking

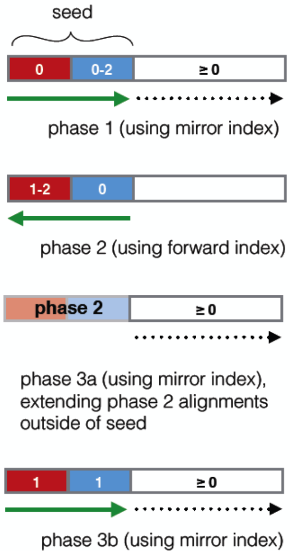


Bowtie – phases

4 possible cases for distribution of 0-2 mutations within the seed:



Same backtracking scheme used to handle mismatches


seed
0
0-2
≥ 0
phase 1 (using mirror index)
1-2
0
phase 2 (using forward index)
phase 2
≥ 0
phase 3a (using mirror index), extending phase 2 alignments outside of seed
1
1
≥ 0
phase 3b (using mirror index)

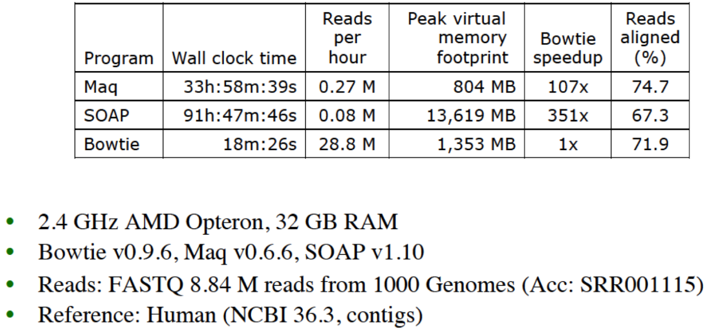
Bowtie 2

- Supports gapped alignment
- Uses bi-directional BWT instead of two separate BWTs
- Supports paired-end alignment
  - Align one end as normal
  - Find the window where the other end could go
  - Do dynamic programming alignment step to align the other end of the pair within this window

BWA (Li and Durbin '09)

- FM-index of a genome is searched with k-neighbourhood of a read.
- Supports gapped alignment (can find small indels)
- Avoids excessive backtracking

Comparison



Program	Wall clock time	Reads per hour	Peak virtual memory footprint	Bowtie speedup	Reads aligned (%)
Maq	33h:58m:39s	0.27 M	804 MB	107x	74.7
SOAP	91h:47m:46s	0.08 M	13,619 MB	351x	67.3
Bowtie	18m:26s	28.8 M	1,353 MB	1x	71.9

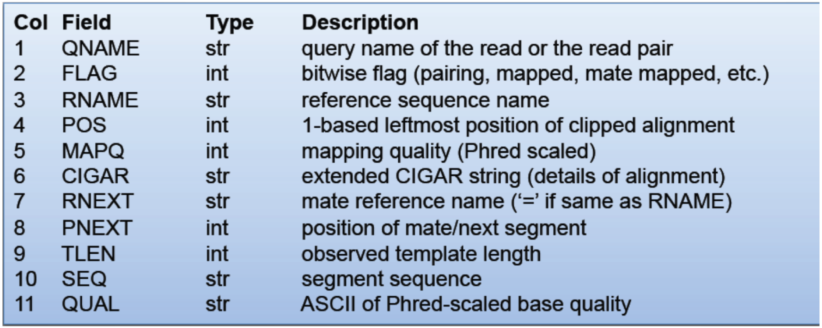
- 2.4 GHz AMD Opteron, 32 GB RAM
- Bowtie v0.9.6, Maq v0.6.6, SOAP v1.10
- Reads: FASTQ 8.84 M reads from 1000 Genomes (Acc: SRR001115)
- Reference: Human (NCBI 36.3, contigs)

SAM files

- Sequence Alignment/Map format
- is a concise file format that contains information about how sequence reads maps to a reference genome
- Can be further compressed in BAM format, which is a binary format of SAM.
- Is produced by bowtie, bwa

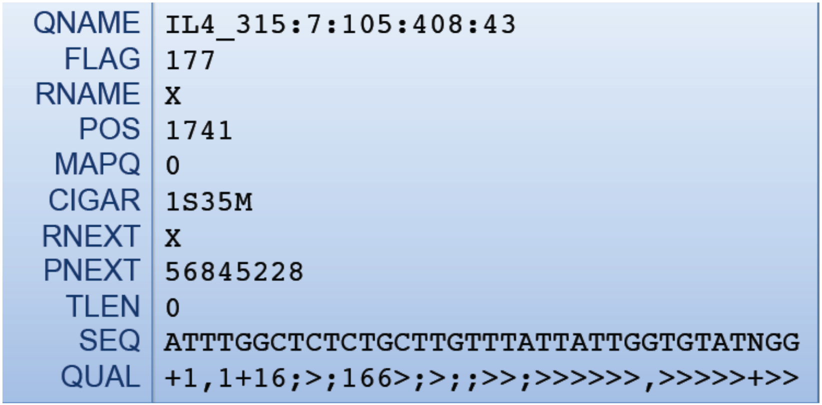
SAM files

- Header
- Alignment lines (one per read)
  - 11 mandatory fields
  - several optional fields (format TAG:TYPE:VALUE)



Col	Field	Type	Description
1	QNAME	str	query name of the read or the read pair
2	FLAG	int	bitwise flag (pairing, mapped, mate mapped, etc.)
3	RNAME	str	reference sequence name
4	POS	int	1-based leftmost position of clipped alignment
5	MAPQ	int	mapping quality (Phred scaled)
6	CIGAR	str	extended CIGAR string (details of alignment)
7	RNEXT	str	mate reference name ('=' if same as RNAME)
8	PNEXT	int	position of mate/next segment
9	TLEN	int	observed template length
10	SEQ	str	segment sequence
11	QUAL	str	ASCII of Phred-scaled base quality

SAM files example



QNAME	IL4_315:7:105:408:43
FLAG	177
RNAME	X
POS	1741
MAPQ	0
CIGAR	1S35M
RNEXT	X
PNEXT	56845228
TLEN	0
SEQ	ATTTGGCTCTCTGCTTGTTTATTATTGGTGTATNGG
QUAL	+1,1+16;>;166>;>;;>>;>>>>>>,>>>>>+>>

Acknowledgements

For input to these slides of this course thanks to

- Norbert Dojer
- Bartosz Wilczyński
- Ben Langmead
- Michael Schatz