We are here

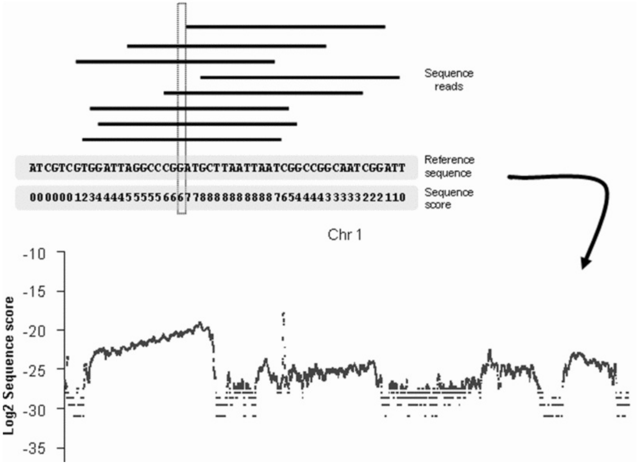
Fast mapping methods

Overview

- Short introduction to modern mapping methods (first DNA mapping)
- NGS alignment format: SAM/BAM

Introduction: A common task

After sequencing reads need to be mapped to reference genome (if present)



(figure from Biochem. Soc. Trans. (2008) 36, 1091-1096)

Introduction

These tasks do not sound very difficult:

Problem 1 *Given a long text t and a short query q. Is there an occurrence of q in t?*
Problem 2 *Given a long text t and many short queries $q_1, \ldots, q_k$. For each query sequence q_i, find all its occurrences in t.*

Introduction

Example: The text t is a genomic sequence and the queries are short reads from NGS sequencing.

- Text $t =$
  tttttttttttttttgagacggagtctcgctctgtcgcccaggctggagtgcagt ggcgggatctcggctcactgcaagctccgcctcccgggttcacgcca ttctcctgcctcagcctcccaagtagctgggactacaggcgcccgcca ctacgcccggctaattttttgtatttttagtagagacggggtttcaccgtttta gccgggatggtctcgatctcctgacctcgtgatccgcccgcctcggcct cccaaagtgctgggattacaggcgt
- Query $q =$ tta
- Find (all) occurrences of query in text

Introduction

- Result:
  tttttttttttttttgagacggagtctcgctctgtcgcccaggctggagtgcagt ggcgggatctcggctcactgcaagctccgcctcccgggttcacgcca ttctcctgcctcagcctcccaagtagctgggactacaggcgcccgcca ctacgcccggctaattttttgtatttttagtagagacggggtttcaccgtttta gccgggatggtctcgatctcctgacctcgtgatccgcccgcctcggcct cccaaagtgctgggattacaggcgt

Mapping of NGS data

Assume for the species of interest, a reference genome exists.

To make sense of the reads in an NGS data set, their positions within the reference sequence must be determined.

This process is known as aligning or mapping the read to the reference

To map reads to sequences of origin is the first and (maybe) most crucial step of NGS analysis.

Mapping of NGS data

The process of determining the genomic positions of the reads actually leads to the reconstruction of the genome by using the sequence of a closely related one.

Therefore this process is also referred to as *resequencing*.

The other, non-comparative approach to reconstruct the genome from reads is assembly, also known as the de novo reconstruction of the genome.

Mapping of NGS data

Problems that can occur:

- SNPs
- error rate of sequencing process
- RNA editing
- single base insertion / deletions
- splicing and fusion

Reads can either be mapped

- uniquely with 0-2 (?) mismatches (and/or insertions / deletions), or
- non-uniquely with 0-2 mismatches, or
- not at all.

Mapping of NGS data: Challenges

In general mapping of reads is the most commonly used bioinformatics task: local pairwise alignment

The problem and first challenge: if the reference genome is very large, and if we have millions of reads, how quickly can we align the reads to the genome?

Problem: algorithms such as Smith-Waterman local alignment or even BLAST are too computationally demanding and too memory demanding.

The second challenge: if a read comes from a repetitive element in the reference, the method must decide which copy of the repeat the read belongs to.

Overview of mapping algorithms

Two mapping versions:

1. without large gaps (for most WGS data)
2. in case of RNA-seq when the data is from a eukaryotic organism, large gaps must be allowed, in order to span introns (so-called spliced-read mappers).

Overview of mapping algorithms

All current mapping methods construct an auxiliary data structure, the index, of either the reads or the reference to speed up their mapping algorithms.

Overview of mapping algorithms

All current mapping methods construct an auxiliary data structure, the index, of either the reads or the reference to speed up their mapping algorithms.

Index of reference $\rightarrow$ more memory needed, faster mapping

Index of reads $\rightarrow$ memory needed depends on size of data set, slower mapping

Overview of mapping algorithms

All current mapping methods construct an auxiliary data structure, the index, of either the reads or the reference to speed up their mapping algorithms.

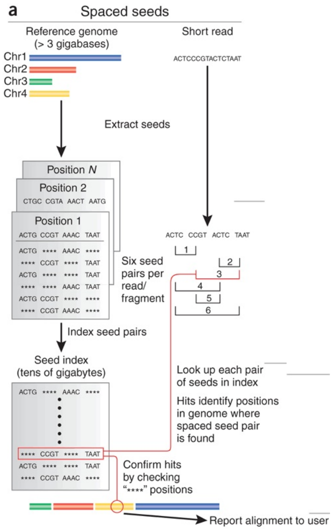
Index of reference → more memory needed, faster mapping

Index of reads → memory needed depends on size of data set, slower mapping

Depending on type of index, the algorithms can be categorized as either being based on

- Seeds and Hash tables or on
- Suffix trees / arrays and the Burrows-Wheeler transformation

Overview of seed-based approach



Source: C Trapnell and S Salzberg, Nature Biotechnology 2009

- A read is divided into four segments of equal length, called the seeds
- If a read maps perfectly to reference, then also all seeds
- If a read has 1 mismatch (due to a SNP for example), then 3 seeds will match perfectly
- If a read has 2 mismatches, then at least 2 seeds will match perfectly
- Look up pairs of seeds that match perfectly

Overview of seed- and hash table-based algorithms

- Seed-and-extend, long seed matches: BLAST
- Seed-and-extend, multiple short seed matches: SSAHA, BLAT
- Using spaced seeds: $\rightarrow$ improved sensitivity
  - Index of reads, allows for mismatches: ELAND (Commercial!), one of the fastest
  - Ma et al.
- Using q-grams (to allow for gaps in seeds): RazerS

Burrows-Wheeler transformation

Given: text T plus \$ (lexicographically smaller than all chars of T)

Build: matrix of T comprised of all cyclic rotations of T, and then all rows sorted lexicographically.

Burrows-Wheeler transformation (BWT) of T is the rightmost column of that matrix.

Burrows-Wheeler transformation

Transformations				
Input	All Rotations	Sorting all Rows	Taking last column	Output
T= acaacg$				

Burrows-Wheeler transformation

Transformations				
Input	All Rotations	Sorting all Rows	Taking last column	Output
	acaacg\$			
	caacg\$a			
	aacg\$ac			
T= acaacg\$	acg\$aca			
	cg\$acaa			
	g\$acaac			
	\$acaacg			

Burrows-Wheeler transformation

Transformations				
Input	All Rotations	Sorting all Rows	Taking last column	Output
	acaacg$	**$**acaacg		
	caacg$a	**a**acg$ac		
	aacg$ac	**a**caacg$		
T= acaacg$	acg$aca	**a**cg$aca		
	cg$acaa	**c**aacg$a		
	g$acaac	**c**g$acaa		
	$acaacg	**g**$acaac		

Burrows-Wheeler transformation

Transformations				
Input	All Rotations	Sorting all Rows	Taking last column	Output
	acaacg$	**$**acaacg	$acaac**g**	
	caacg$a	**a**acg$ac	aacg$a**c**	
	aacg$ac	**a**caacg$	acaacg**$**	
T= acaacg$	acg$aca	**a**cg$aca	acg$ac**a**	
	cg$acaa	**c**aacg$a	caacg$**a**	
	g$acaac	**c**g$acaa	cg$aca**a**	
	$acaacg	**g**$acaac	g$acaa**c**	

Burrows-Wheeler transformation

Transformations				
Input	All Rotations	Sorting all Rows	Taking last column	Output
	acaacg$	**$**acaacg	$acaac**g**	
	caacg$a	**a**acg$ac	aacg$a**c**	
	aacg$ac	**a**caacg$	acaacg**$**	
T= acaacg$	acg$aca	**a**cg$aca	acg$ac**a**	gc$aaac
	cg$acaa	**c**aacg$a	caacg$**a**	
	g$acaac	**c**g$acaa	cg$aca**a**	
	$acaacg	**g**$acaac	g$acaa**c**	

Characteristics of BWT

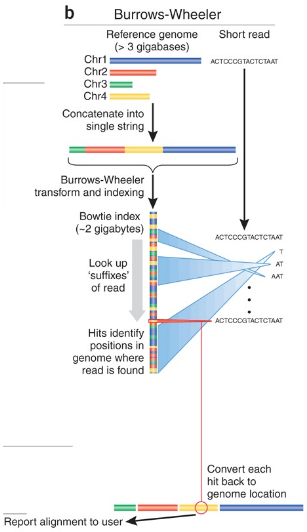
If the original string had several substrings that occurred often, then the transformed string will have several places where a single character is repeated multiple times in a row. This is useful for compression.

Example:

'tomorrow and tomorrow and tomorrow' $\rightarrow$

'wwwdd nnoooaatttmmmrrrrrrooo $ooo'

Mapping using BWT


b
Burrows-Wheeler
Reference genome
(> 3 gigabases)
Short read
Chr1
Chr2
Chr3
Chr4
ACTCCCGTACTCTAAT
Concatenate into single string
Burrows-Wheeler transform and indexing
Bowtie index (~2 gigabytes)
ACTCCCGTACTCTAAT
T
AT
AAT
Look up 'suffixes' of read
ACTCCCGTACTCTAAT
Hits identify positions in genome where read is found
Convert each hit back to genome location
Report alignment to user

- The genome is transformed into a compressed BWT: size of human genome for example is less than 3GB
- BWT-based methods such as Bowtie maps bases of reads one after the other (starting at the 3' end of each read) by first trying to find exact matches
- BWT-based approaches are much more complicates than seed-based ones, but much faster, too.

Source: C Trapnell and S Salzberg, Nature Biotechnology 2009

Overview of suffix tree/array/BWT based algorithms

Two step approach: first identify exact matches via suffix tree-based structure, then build inexact alignment

- Suffix trees: MUMmer, OASIS
- Suffix array: Vmatch, Segemehl
- FM-index and BWT: **Bowtie/2, BWA**, SOAP2, BWT-SW, BWA-SW

The SAM/BAM formats

More than 20 read mapping tools exist, some of them have their own output format.

The SAM/BAM format was created as a common file format for aligned sequences (Li, 2009).

SAM

= Sequence Alignment/Map Format
human readable, simple to parse

BAM

= Binary Alignment/Map Format
compressed, efficient access

The latest version is ec1fec2 (3 March 2015).

SAM format: Header

SAM is a tab-delimited text format.
A SAM file contains a header (optional) and an alignment section of many read records.

Header

Header lines start with @ and contain a header type, and key:value pairs.

The key/tag is a two-letter string.
The standard defines 5 header types with a total of 26 keys.

Example:

`@HD`	`VN:1.5`	SAM file format version
`@SQ`	`SN:chr20 LN:62435964`	SN = ref. sequence, LN = ref. length

SAM format: Read Records

Also alignment entries in tab-delimited text format, each line has 11 mandatory fields.

Alignment of read records in SAM format

```
QNAME FLAG RNAME POS MAPQ CIGAR MRNM MPOS ISIZE SEQ QUAL
```

QNAME	read identifier (string)
FLAG	bit-wise flags describing the read (int)
RNAME	target name (header SQ field, string)
POS	mapping start position (int)
MAPQ	mapping quality (int)
CIGAR	extended CIGAR alignment string (string)
MRNM, MPOS, ISIZE	Mate-pair mapping information
SEQ	read sequence (string)
QUAL	read sequence quality values (string)

Further tags can be added at the end of each line. 32 additional tag fields are defined in the SAM standard.

The SAM format: Example

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCTAAGCTAA * SA:Z:ref,29,-,6H5M,17
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 2064 ref 29 17 6H5M * 0 0 TAGGC * SA:Z:ref,9,+,5S6M,30,1
```

SAM format: CIGAR strings

The central feature of a SAM line is the CIGAR string. CIGAR strings describe the alignment of a read with the target/reference using matches (M), insertions (I), deletions (D), clipping (S), skipping (N), and padding (P).

CIGAR example 1

```
  REF: CACGATCAGACCGATACGTCCGA
READ1:   CGATCAGACCGAGA           14M
READ2:     ATCA--CCGATAC          4M2D7M
READ3:    GATCAGGGCCG             6M1I4M
READ4:    GACCA---------GTCCG     5M9N5M
```

SAM format: CIGAR strings

De novo assembly can for example result in reads that have insertions wrt reference, but how these inserted sequences are aligned against each other, is not clear. A so-called padded alignment fully resolves this.

CIGAR example 2

```
  REF: CACGATCA**GACCGATACGTCCGA
READ1:   CGATCAGAGACCGAGA         6M2I8M
READ2:     ATCA*AGACCGATAC        4M1P1I9M
READ3:    GATCA**GACCG            5M2P5M
```

BAM: A binary version of SAM

- The BAM format stores SAM records in binary form.
- Records are compressed using the BGZF format. BGZF is a block compression method on top of gzip.

- The BGZF format can be uncompressed using `gunzip`.
- Even in the compressed form, applications can directly access any record in the file, such as FastQC or other tools from the samtools package for example.

BAM files are often used together with .bai files that is an index to a sorted BAM file for a very fast lookup of records.

For more details, see the SAM/BAM specification document at `http://samtools.sourceforge.net/SAM1.pdf`.

Viewing alignments in Genome browsers

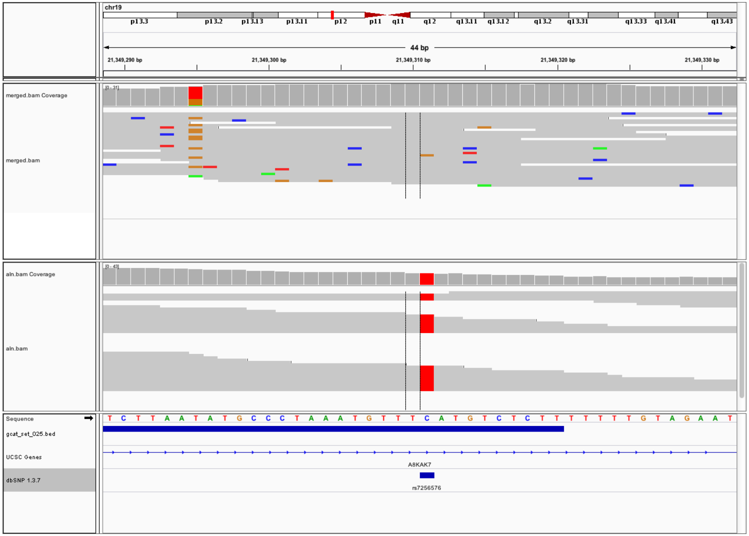
- display genomic data in a ‘position-centric’ view
- genome serves as reference for positions
- usually track-based
- varying levels of interactivity
- browsing vs exploration
- web-browser-based or desktop applications

Viewing alignments in Genome browsers

Integrative Genome Viewer:

- visualization tool for interactive exploration of large, integrated datasets.
- supports a wide variety of data types including sequence alignments, expression data, copy number variation, RNA-seq, annotations

Viewing alignments in Genome browsers



Further common file formats

- BED files are used to represent, has 3 mandatory entries: chrom, chromStart (0-based), chromEnd
  Example: exon positions in the human genomes

```
chr1 14642 14882
chr1 14943 15063
chr1 15751 15990
chr1 16599 16719
chr1 16834 17074
chr1 17211 17331
chr1 30275 30431
chr1 69069 70029
```

  Large bed files can be transformed into binary format: bigBed

Further common file formats

- GTF / GFF files are important for providing feature annotations, e.g., of exons organization into transcripts and genes.
  Example:

```
##gff-version 3
# Generated on Tue Nov 27 19:25:49 2012
# UCSC table file ./ucsc_tables/hg19/ensGene.txt
chr1 ensGene gene 11869 14412 . + . Name=ENSG00000223972;ID=ENSG00000223972;Al
chr1 ensGene ncRNA 11869 14409 . + . Name=ENST00000456328;Parent=ENSG000002239
chr1 ensGene exon 11869 12227 . + . Name=ENST00000456328.exon0;Parent=ENST0000
chr1 ensGene ncRNA 11872 14412 . + . Name=ENST00000515242;Parent=ENSG000002239
chr1 ensGene exon 11872 12227 . + . Name=ENST00000515242.exon0;Parent=ENST0000
chr1 ensGene ncRNA 11874 14409 . + . Name=ENST00000518655;Parent=ENSG000002239
chr1 ensGene exon 11874 12227 . + . Name=ENST00000518655.exon0;Parent=ENST0000
chr1 ensGene gene 14363 29806 . - . Name=ENSG00000227232;ID=ENSG00000227232;Al
```

Practical Session

Questions?

Practical Session

Questions?

Now off to the second practical session

Learning Objectives of Practical Session

- Run BWA and Bowtie2 with parameters suitable for DNA-seq data
- Use samtools to demonstrate the features of the SAM/BAM format and basic manipulation of these alignment files (view, sort, index, filter)
- Use samtools flagstat, samstat, to assess quality of alignments (e.g., how many reads have been mapped (uniquely)?)
- Use samtools rmdup to remove PCR duplicates
- Use Qualimap to assess essential statistics after mapping, such as coverage etc.