SevenBridges genomics

Applied Bioinformatics

Short Read Alignment
October 2015, Belgrade

Goran Rakočević, PhD
goran.rakocevic@sbgenomics.com

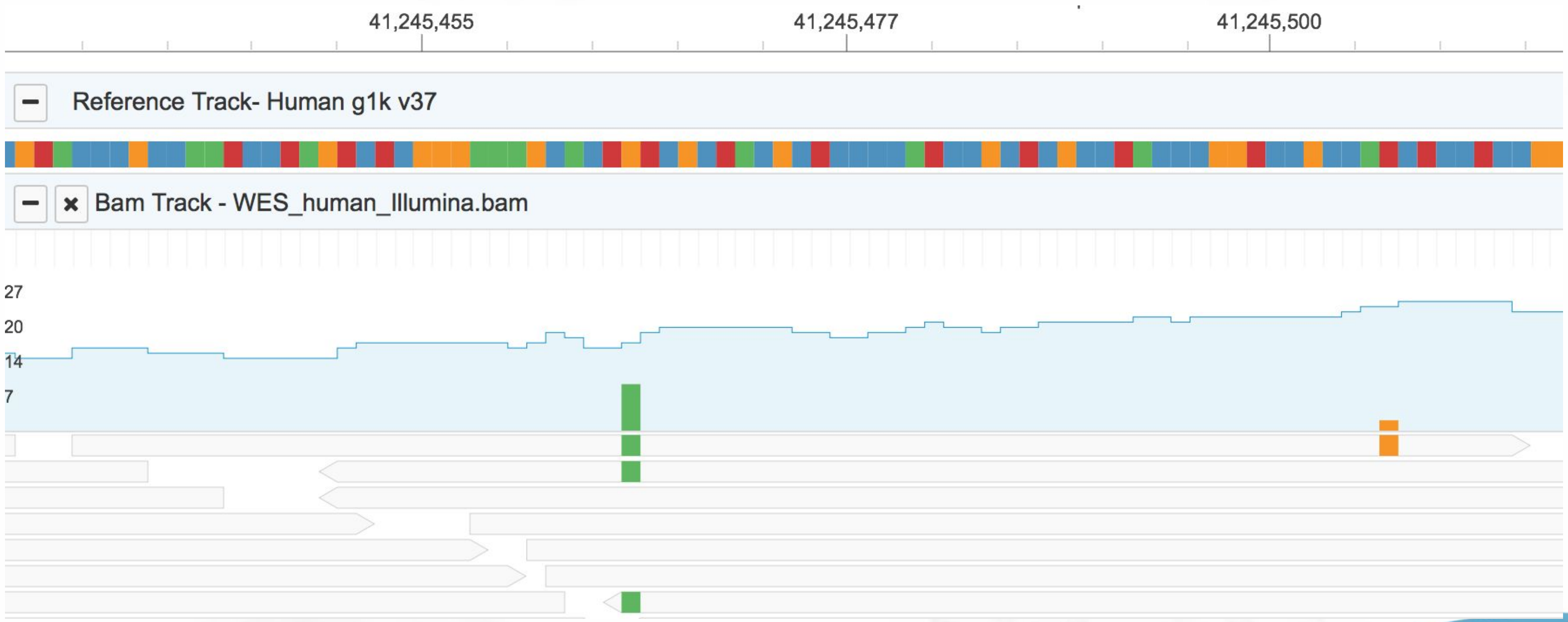
Today’s agenda

1. Overview of alignment of short reads
    a. Two-step aligner algorithms and examples
2. Using the BWA-MEM tool
3. SAM and BAM file formats
4. Genome browsers (IGV)
5. Using PySam to explore BAM files

DNA Sequencing - Reminder

- We got a FASTQ file with the “reads” - little pieces of the genome

Aligning reads to the reference


41,245,455
41,245,477
41,245,500
Reference Track- Human g1k v37
Bam Track - WES_human_Illumina.bam
27
20
14
7

Penalty (score) -based alignment

- Some of the reads will match perfectly to the reference
- Many reads will not:
  - Genomic variations (‘mutations’, SNP, Indel, etc.)
  - Sequencing errors
- Aligners commonly take a score-based approach:
  - Calculate a score, related to the distance between the read and the local reference sequence
  - Place the read so that the score in maximised
- Many algorithms exist, there is a speed/precision tradeoff

Two-step aligners

- Finding precise alignments can be expensive (for a human genome ~1 billion reads)
- Most modern aligners take a two-step approach (also called “seed and extend”)
- First find “coarse” alignments or seeds
- Than do fine grain alignments in the vicinity of the seeds
- Chose the best scoring fine grain alignment

Coarse alignment step (seeding)

- Find a set of possible coordinates
- Many false-positives, but very is usually fast
- Several common approaches:
  - K-mer hashing-based approaches
  - Radix-tree based approaches
  - FM index-based approaches (Burrows-Wheeler)
- Most require an index-building step
- A common tradeoff is speed vs RAM footprint

A simple coarse aligner

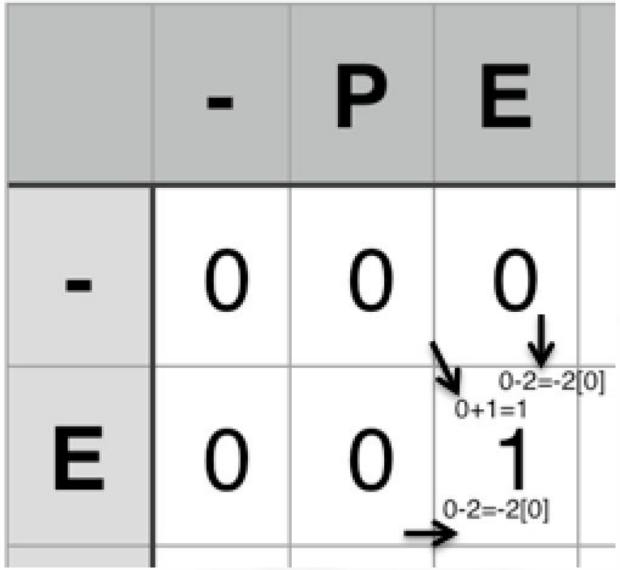
- Example: a simple hash-based scheme
- Create a hash-table which holds the positions, where each kmer in the genome occurs
- For each kmer in the read find the list of locations
- Regions with hits from many different kmers are hits
- Some kmers will have no hits, some many
- Indices quickly grow too large for whole genomes

Fine-grain alignment step (extending)

- Calculates a precise sequences match score
- Algorithms slower than core grain aligners
- Often use a lot of RAM (related to sequence size)
- The extend step also needs to produce
  the information on *how* the sequences match
- Mostly based on dynamic programming

Smith-Waterman aligner

- Dynamic programming local alignment algorithm



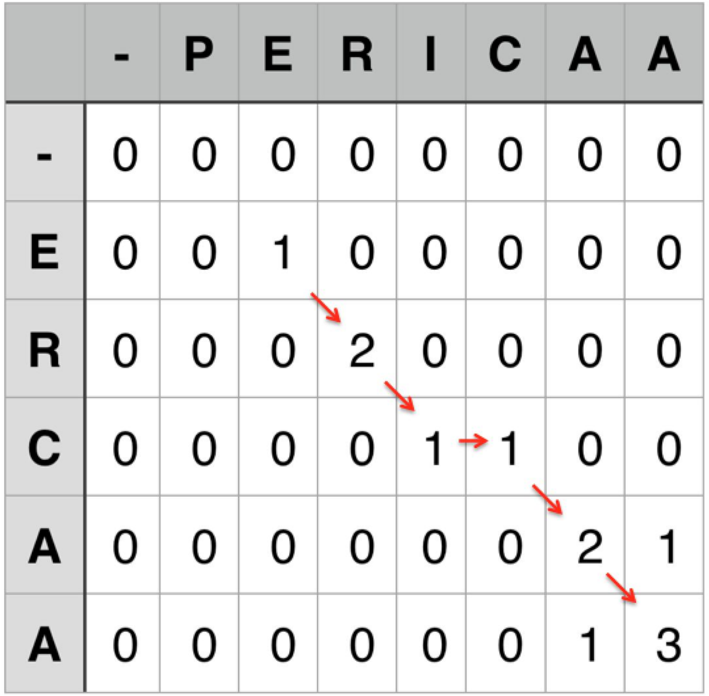
	-	P	E
-	0	0	0
E	0	0	1

0-2=-2[0]
0+1=1
0-2=-2[0]

$$\text{Value}_{i,j} = \begin{cases} 0 \\ \text{Value}_{i-1,\,j-1} + M \\ \text{Value}_{i-1,\,j} + G \\ \text{Value}_{i,\,j-1} + G \end{cases}$$

M = *Match score* if letters match, otherwise *mismatch penalty* (+/- 1 in this example)
G = *Gap penalty* (-2 in this example)

Smith-Waterman aligner



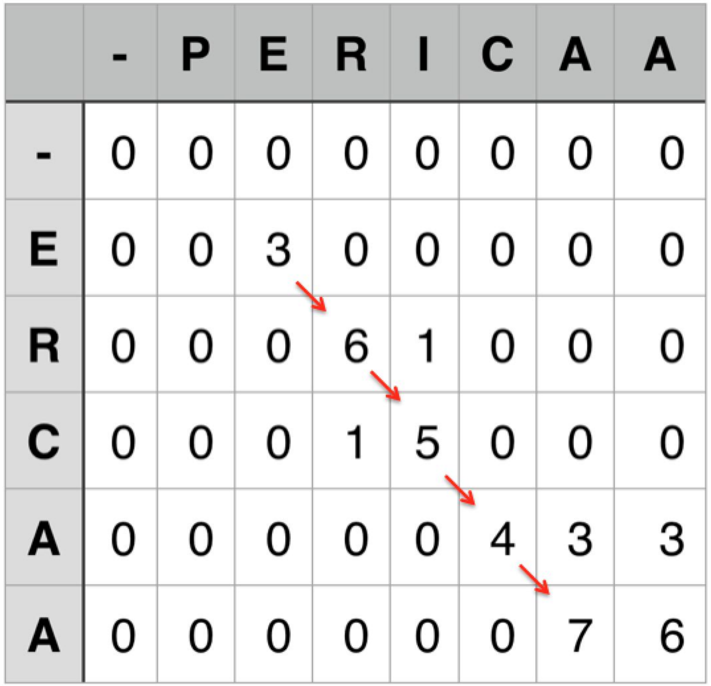
	-	P	E	R	I	C	A	A
-	0	0	0	0	0	0	0	0
E	0	0	1	0	0	0	0	0
R	0	0	0	2	0	0	0	0
C	0	0	0	0	1	1	0	0
A	0	0	0	0	0	0	2	1
A	0	0	0	0	0	0	1	3

- Calculate scores for each field
- Remember the path to each field
- Backtrack from maximum to zero
- Diagonal step is match/mismatch
- Horizontal step is a deletion
- Vertical step in an insertion

Match=1, Mismatch=-1, Gap =-2

P	E	R	I	C	A	A
-	E	R	-	C	A	A

Smith-Waterman aligner



	-	P	E	R	I	C	A	A
-	0	0	0	0	0	0	0	0
E	0	0	3	0	0	0	0	0
R	0	0	0	6	1	0	0	0
C	0	0	0	1	5	0	0	0
A	0	0	0	0	0	4	3	3
A	0	0	0	0	0	0	7	6

- The best path and the scores is defined by choice of M and G!

Match=3, Mismatch=-1, Gap =-5

P	E	R	I	C	A	A
-	E	R	C	A	A	-

SevenBridges genomics

CIGAR Strings

- Run length encoding:

  AAAABBAAAD = 4A2B3A1D

- CIGAR Codes:
  - **M** - alignment match (Match or Mismatch)
  - **I** - insertion, **D** - Deletion, **S** - soft clip
  - H, X, =, P, N - rare in DNA Seq

```
P|E|R|I|C|A|A|
-|E|R|C|A|A|-|  => 1S5M1S
```

```
|P|E|R|I|C|A|A|
|-|E|R|-|C|A|A|  => 1S2M1D2M
```

Exercise 1 - a simple aligner (30 min)

- Implement a simple hash-based aligner in Python
  - A dict can be used to create the index
- To map a read, find locations for each kmer in the read
  - Find the region with most kmers mapping to it
- A fast Smith-Waterman implementation is installed
  - Use this tool to produce fine-grain alignments
- Try modifying the to see the effect
  - Add insertions, deletions, and mismatches

BWA Aligner (BWA-MEM command)

- A widely used aligner for DNA Sequencing
- http://bio-bwa.sourceforge.net for more info
- BWA requires an index to be built:
  - `bwa index ref.fa`
  - Fast for small FASTA files, 2h for whole human genome
  - Produces a set of files in the same folder as the FASTA
- MEM command used for aligning
  - `bwa mem ref.fa read1.fq read2.fq > aln-pe.sam`

Exercise 2 - BWA-MEM (15 min)

- Align the example FASTQ files to the example FASTA
  - All files are in the *data* folder
- First, create the BWA index for the example FASTA file
  - Be sure to make your own copy of the FASTA!
- Align the two example paired end files using the BWA-MEM
  - Pipe the tool output to a file!
- Inspect the produced file
  - You can use less in the Jupyter notebook

SevenBridges genomics

BAM File format

- Standardized format for holding aligned reads
  - The read sequence and qualities
  - Position (chromosome and the first matching base)
  - CIGAR string
  - Flag (various info, like *read has a pair*, *read is aligned*, etc.)
  - Read pair position
  - Other optional tags
- Bgzip compression (roughly ⅓ of raw text)
- Besides BAM, SAM (plain text) also exists

BAM File format (2)

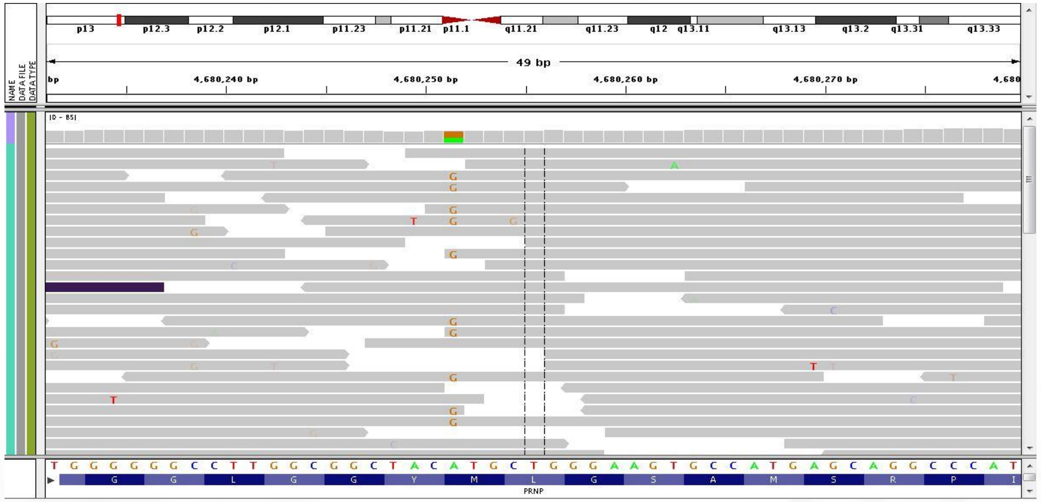
- BAM is 0-based, SAM is 1-based
- BAM files can be sorted:
  - Coordinate sort, by read position
  - Query name (read name), read pairs are placed together
- Coordinate sorted BAM files can be *indexed*
- By convention, BAM index files:
  - Have .bai appended to the name
  - Are placed in the same folder as the BAM
  - Usually not explicitly passed to tools

Exercise 3 - SamtoBam and Sort (10 min)

- Convert the SAM file you created by BWA-MEM into BAM
  - You can use Samtools to do this
- Coordinate sort this BAM file
  - You can use Samtools or Picard to do this
- Use Samtools to view the contents of this BAM file
  - Pipe the output into less for easier viewing
- Create the bai index for this file
  - You can use Samtools or Picard to do this

The Integrative Genomics Viewer (IGV)

- IGV - A visualization tool for genomic data
- Can be used to visualize BAM, BED, and many other file types



NAME
DATA FILE
DATA TYPE
p13 p12.3 p12.2 p12.1 p11.23 p11.21 p11.1 q11.21 q11.23 q12 q13.11 q13.13 q13.2 q13.31 q13.33
49 bp
4,680,240 bp
4,680,250 bp
4,680,260 bp
4,680,270 bp
PRNP

IGV - Exercise 4 (15 min)

- Use IGV to inspect a BAM file:
  - WES_human_Illumina.bam from the *data* folder
- A BAM files needs to be sorted and indexed!
- Go to BRCA1 region and zoom to see reads
- Try different viewing options:
  - View as pairs
  - Collapsed view
  - Color by strand/insert size
- Guess where there are mutations, and what are errors!

Pysam - Python interface for BAMs

- Pysam can be used to process BAM files
- pysam.AlignmentFile
  - AlignmentFile(path_to_file)
  - for read in AlignmentFile(path_to_file):
- Reads are wrapped in AlignedSegment objects
  - AlignedSegment provides access to read fields and helpers
- pysam.AlignmentFile supports fetching regions
  - The BAM file needs to be sorted and indexed!

SevenBridges genomics

Pysam - Exercise 5 (30 min)

- Create an AlignmentFile object
  - Use WES_human_Illumina.bam from the *data* folder
- Take the first read from the AlignmentFile
  - Inspect the fields in the AlignedSegment
- How many unmapped reads are there in the file?
- Create a BedTool with the exome.bed from the ../data folder
- Get the regions from the BedTool that cover BRCA1 gene
- Fetch all of the reads from the BAM file mapped to BRCA1

Questions?