Alignment & BAM files – Answers Part 2

4. Difference between bowtie2 and bwa?

Burrows-Wheeler Transform (BWT) is an efficient data indexing technique that maintains a relatively small memory footprint when searching through a given data block. BWT was extended by Ferragina and Manzini to a newer data structure, named FM-index, to support exact matching. By transforming the genome into an FM-index, the lookup performance of the algorithm improves for the cases where a single read matches multiple locations in the genome. However, the improved performance comes with a significantly large index build up time compared to hash tables.
Both bowtie2 and bwa are based on BWT.
Bowtie starts by building an FM-index for the reference genome and then uses the modified Ferragina and Manzini matching algorithm to find the mapping location. There are two main versions of Bowtie namely Bowtie and Bowtie 2. Bowtie 2 is mainly designed to handle reads longer than 50 bps. Additionally, Bowtie 2 supports features not handled by Bowtie.
The BWA tool uses the Ferragina and Manzini matching algorithm to find exact matches, similar to Bowtie. To find inexact matches, the authors provided a new backtracking algorithm that searches for matches between substring of the reference genome and the query within a certain defined distance.
BWA accept or reject an alignment based on counting the number of mismatches between the read and the corresponding genomic position. On the other hand, Bowtie use a quality threshold to perform the same function. The quality threshold is different from the mapping quality. The former is the probability of the occurrence of the read sequence given an alignment location while the latter is the Bayesian posterior probability for the correctness of the alignment location calculated from all of the alignments found for the read. Both Bowtie2 and BWA determine the maximum number of mismatches in the read based on the read length. Gapped alignment is enabled for Bowtie2 and BWA. Minimum and maximum insert sizes for paired-end mapping are 0 and 500 for BWA and Bowtie2.

5. BWA
5.1. What is it?
5.2. Usage?

BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two are for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate. BWA-MEM also has better performance than BWA-backtrack for 70-100bp Illumina reads.
For short sequences, BWA-backtrack may be better. BWA-SW may have better sensitivity when alignment gaps are frequent.
BWA-backtrack is mainly designed for sequencing error rates below 2%. Although users can ask it to tolerate more errors by tuning command-line options, its performance is quickly degraded. BWA-SW and BWA-MEM both tolerate more errors given longer alignment. Simulation suggests that they may work well given 2% error for an 100bp alignment, 3% error for a 200bp, 5% for 500bp and 10% for 1000bp or longer alignment.
BWA only does alignment and it outputs alignments in the SAM format.

5.3. Create indexes

bwa index inputRef.fasta

5.4. Align fastq pairs

bwa aln inputRef.fasta input_1.fastq > output_1.sai
bwa aln inputRef.fasta input_2.fastq > output_2.sai
bwa sampe inputRef.fasta output_1.sai otput_2.sai input_1.fastq input_2.fastq > output_bwa.sam

6. Bowtie2
6.1. Create Dockerfile

Uploaded ti gitHub file DockerfileBowtie2

6.2. Usage?

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s of characters to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index (based on the Burrows-Wheeler Transform or BWT) to keep its memory footprint small. Bowtie 2 outputs alignments in SAM format.
Bowtie 2 is geared toward aligning relatively short sequencing reads to long genomes. That said, it handles arbitrarily small reference sequences (e.g. amplicons) and very long reads (i.e. upwards of 10s or 100s of kilobases), though it is slower in those settings. It is optimized for the read lengths and error modes yielded by typical Illumina sequencers. Bowtie 2 does not support alignment of colorspace reads. (Bowtie 1 does.)

6.3. Create indexes

bowtie2-build input.fasta output_index

6.4. Align fastq pairs

bowtie2 -x output_index -1 input_1.fastq -2 input_2.fastq -S output_bowtie.sam

7. What is fasta indexing?

Using an fai index file in conjunction with a FASTA file containing reference sequences enables efficient access to arbitrary regions within those reference sequences. The index file typically has the same filename as the corresponding FASTA file, with .fai appended.
An fai index file is a text file consisting of lines each with five TAB-delimited columns: NAME - Name of this reference sequence; LENGTH - Total length of this reference sequence, in bases; OFFSET - Offset within the FASTA file of this sequence's first base; LINEBASES - The number of bases on each line; LINEWIDTH - The number of bytes in each line, including the newline.
The NAME and LENGTH columns contain the same data as would appear in the SN and LN fields of a SAM @SQ header for the same reference sequence. The OFFSET column contains the offset within the FASTA file, in bytes starting from zero, of the first base of this reference sequence, i.e., of the character following the newline at the end of the ">" header line. Typically the lines of a fai index file appear in the order in which the reference sequences appear in the FASTA file, so .fai files are typically sorted according to this column. The LINEBASES column contains the number of bases in each of the sequence lines that form the body of this reference sequence, apart from the final line which may be shorter. The LINEWIDTH column contains the number of bytes in each of the sequence lines (except perhaps the final line), thus differing from LINEBASES in that it also counts the bytes forming the line terminator.

8. STAR
8.1. What is it?

STAR (Spliced Transcripts Alignment to a Reference) is an aligner designed to specifically address many of the challenges of RNA-seq data mapping using a strategy to account for spliced alignments. STAR is shown to have high accuracy and outperforms other aligners by more than a factor of 50 in mapping speed, but it is memory intensive. The algorithm achieves this highly efficient mapping by performing a two-step process: Seed searching and Clustering, stitching, and scoring.
Seed searching: For every read that STAR aligns, STAR will search for the longest sequence that exactly matches one or more locations on the reference genome. These longest matching sequences are called the Maximal Mappable Prefixes (MMPs). The different parts of the read that are mapped separately are called 'seeds'. So the first MMP that is mapped to the genome is called seed1. STAR will then search again for only the unmapped portion of the read to find the next longest sequence that exactly matches the reference genome, or the next MMP, which will be seed2. This sequential searching of only the unmapped portions of reads underlies the efficiency of the STAR algorithm. STAR uses an uncompressed suffix array (SA) to efficiently search for the MMPs, this allows for quick searching against even the largest reference genomes. Other slower aligners use algorithms that often search for the entire read sequence before splitting reads and performing iterative rounds of mapping. If STAR does not find an exact matching sequence for each part of the read due to mismatches or indels, the previous MMPs will be extended. If extension does not give a good alignment, then the poor quality or adapter sequence (or other contaminating sequence) will be soft clipped.
Clustering, stitching and scoring: The separate seeds are stitched together to create a complete read by first clustering the seeds together based on proximity to a set of 'anchor' seeds, or seeds that are not multi-mapping. Then the seeds are stitched together based on the best alignment for the read (scoring based on mismatches, indels, gaps, etc.).

8.2. Create Dockerfile
8.3. Remove answer to Drive/Git

Uploaded to gitHub file DockerfileSTAR.

8.4. Create indexes

STAR --runMode genomeGenerate --genomeFastaFiles inputRef.fasta

8.5. Aligned fastq pairs

STAR --readFilesIn input_1.fastq input_2.fastq --outFileNamePrefix output_star

8.6. Output unmapped reads within the main SAM file

To save also unmapped reads:
STAR --readFilesIn input_1.fastq einput_2.fastq --outSAMunmapped Within --outFileNamePrefix output_star2

samtools view -f 4 output_starAligned.sam