Manual Reference Pages - bwa (1)

NAME

bwa - Burrows-Wheeler Alignment Tool

CONTENTS

SYNOPSIS

```
bwa index -a bwtsw database.fasta

bwa aln database.fasta short_read.fastq > aln_sa.sai

bwa samse database.fasta aln_sa.sai short_read.fastq > aln.sam

bwa sampe database.fasta aln_sa1.sai aln_sa2.sai read1.fq read2.fq > aln.sam

bwa bwasw database.fasta long_read.fastq > aln.sam
```

DESCRIPTION

BWA is a fast light-weighted tool that aligns relatively short sequences (queries) to a sequence database (targe), such as the human reference genome. It implements two different algorithms, both based on Burrows-Wheeler Transform (BWT). The first algorithm is designed for short queries up to ~200bp with low error rate (<3%). It does gapped global alignment w.r.t. queries, supports paired-end reads, and is one of the fastest short read alignment algorithms to date while also visiting suboptimal hits. The second algorithm, BWA-SW, is designed for long reads with more errors. It performs heuristic Smith-Waterman-like alignment to find high-scoring local hits (and thus chimera). On low-error short queries, BWA-SW is slower and less accurate than the first algorithm, but on long queries, it is better.

For both algorithms, the database file in the FASTA format must be first indexed with the ‘**index**’ command, which typically takes a few hours. The first algorithm is implemented via the ‘**aln**’ command, which finds the suffix array (SA) coordinates of good hits of each individual read, and the ‘**samse/sampe**’ command, which converts SA coordinates to chromosomal coordinate and pairs reads (for ‘sampe’). The second algorithm is invoked by the ‘**bwasw**’ command. It works for single-end reads only.

COMMANDS AND OPTIONS

index `bwa index [-p prefix] [-a algoType] [-c] <in.db.fasta>`

Index database sequences in the FASTA format.

OPTIONS:

-c Build color-space index. The input fast should be in nucleotide space.

-p *STR* Prefix of the output database [same as db filename]

-a *STR* Algorithm for constructing BWT index. Available options are:

- **is** IS linear-time algorithm for constructing suffix array. It requires 5.37N memory where N is the size of the database. IS is moderately fast, but does not work with database larger than 2GB. IS is the default algorithm due to its simplicity. The current codes for IS algorithm are reimplemented by Yuta Mori.
- **bwtsw** Algorithm implemented in BWT-SW. This method works with the whole human genome, but it does not work with database smaller than 10MB and it is usually slower than IS.

aln `bwa aln [-n maxDiff] [-o maxGapO] [-e maxGapE] [-d nDelTail] [-i nIndelEnd] [-k maxSeedDiff] [-l seedLen] [-t nThrds] [-cRN] [-M misMsc] [-O gapOsc] [-E gapEsc] [-q trimQual] <in.db.fasta> <in.query.fq> > <out.sai>`

Find the SA coordinates of the input reads. Maximum *maxSeedDiff* differences are allowed in the first *seedLen* subsequence and maximum *maxDiff* differences are allowed in the whole sequence.

OPTIONS:

-n *NUM* Maximum edit distance if the value is INT, or the fraction of missing alignments given 2% uniform base error rate if FLOAT. In the latter case, the maximum edit distance is automatically chosen for different read lengths. [0.04]

-o *INT* Maximum number of gap opens [1]

-e *INT* Maximum number of gap extensions, -1 for k-difference mode (disallowing long gaps) [-1]

-d *INT* Disallow a long deletion within INT bp towards the 3’-end [16]

-i *INT* Disallow an indel within INT bp towards the ends [5]

-l *INT* Take the first INT subsequence as seed. If INT is larger than the query sequence, seeding will be disabled. For long reads, this option is typically ranged from 25 to 35 for ‘-k 2’. [inf]

-k *INT* Maximum edit distance in the seed [2]

-t *INT* Number of threads (multi-threading mode) [1]

-M *INT* Mismatch penalty. BWA will not search for suboptimal hits with a score lower than (bestScore-misMsc). [3]

-O *INT* Gap open penalty [11]

-E *INT* Gap extension penalty [4]

-R *INT* Proceed with suboptimal alignments if there are no more than INT equally best hits. This option only affects paired-end mapping. Increasing this threshold helps to improve the pairing accuracy at the cost of speed, especially for short reads (~32bp).

-c Reverse query but not complement it, which is required for alignment in the color space.

-N Disable iterative search. All hits with no more than *maxDiff* differences will be found. This mode is much slower than the default.

-q *INT* Parameter for read trimming. BWA trims a read down to argmax_x{\sum_{i=x+1}^l(INT-q_i)} if q_l<INT where l is the original read length. [0]

-I The input is in the Illumina 1.3+ read format (quality equals ASCII-64).

-B *INT* Length of barcode starting from the 5'-end. When *INT* is positive, the barcode of each read will be trimmed before mapping and will be written at the **BC** SAM tag. For paired-end reads, the barcode from both ends are concatenated. [0]

-b Specify the input read sequence file is the BAM format. For paired-end data, two ends in a pair must be grouped together and options **-1** or **-2** are usually applied to specify which end should be mapped. Typical command lines for mapping pair-end data in the BAM format are:

```
bwa aln ref.fa -b1 reads.bam > 1.sai
bwa aln ref.fa -b2 reads.bam > 2.sai
bwa sampe ref.fa 1.sai 2.sai reads.bam reads.bam > aln.sam
```

-0 When **-b** is specified, only use single-end reads in mapping.

-1 When **-b** is specified, only use the first read in a read pair in mapping (skip single-end reads and the second reads).

-2 When **-b** is specified, only use the second read in a read pair in mapping.

samse bwa samse [-n maxOcc] <in.db.fasta> <in.sai> <in.fq> > <out.sam>

Generate alignments in the SAM format given single-end reads. Repetitive hits will be randomly chosen.

OPTIONS:

-n *INT* Maximum number of alignments to output in the XA tag for reads paired properly. If a read has more than INT hits, the XA tag will not be written. [3]

-r *STR* Specify the read group in a format like '@RG\tID:foo\tSM:bar'. [null]

sampe bwa sampe [-a maxInsSize] [-o maxOcc] [-n maxHitPaired] [-N maxHitDis] [-P] <in.db.fasta> <in1.sai> <in2.sai> <in1.fq> <in2.fq> > <out.sam>

Generate alignments in the SAM format given paired-end reads. Repetitive read pairs will be placed randomly.

OPTIONS:

-a *INT* Maximum insert size for a read pair to be considered being mapped properly. Since 0.4.5, this option is only used when there are not enough good alignment to infer the distribution of insert sizes. [500]

-o *INT* Maximum occurrences of a read for pairing. A read with more occurrneces will be treated as a single-end read. Reducing this parameter helps faster pairing. [100000]

-P Load the entire FM-index into memory to reduce disk operations (base-space reads only). With this option, at least 1.25N bytes of memory are required, where N is the length of the genome.

-n *INT* Maximum number of alignments to output in the XA tag for reads paired properly. If a read has more than INT hits, the XA tag will not be written. [3]

-N *INT* Maximum number of alignments to output in the XA tag for disconcordant read pairs (excluding singletons). If a read has more than INT hits, the XA tag will not be written. [10]

-r *STR* Specify the read group in a format like '@RG\tID:foo\tSM:bar'. [null]

bwasw bwa bwasw [-a matchScore] [-b mmPen] [-q gapOpenPen] [-r gapExtPen] [-t nThreads] [-w bandWidth] [-T thres] [-s hspIntv] [-z zBest] [-N nHspRev] [-c thresCoef] <in.db.fasta> <in.fq>

Align query sequences in the <in.fq> file.

OPTIONS:

-a *INT*	Score of a match [1]
-b *INT*	Mismatch penalty [3]
-q *INT*	Gap open penalty [5]
-r *INT*	Gap extension penalty. The penalty for a contiguous gap of size k is q+k*r. [2]
-t *INT*	Number of threads in the multi-threading mode [1]
-w *INT*	Band width in the banded alignment [33]
-T *INT*	Minimum score threshold divided by a [37]
-c *FLOAT*	Coefficient for threshold adjustment according to query length. Given an l-long query, the threshold for a hit to be retained is a*max{T,c*log(l)}. [5.5]
-z *INT*	Z-best heuristics. Higher -z increases accuracy at the cost of speed. [1]
-s *INT*	Maximum SA interval size for initiating a seed. Higher -s increases accuracy at the cost of speed. [3]
-N *INT*	Minimum number of seeds supporting the resultant alignment to skip reverse alignment. [5]

SAM ALIGNMENT FORMAT

The output of the **'aln'** command is binary and designed for BWA use only. BWA outputs the final alignment in the SAM (Sequence Alignment/Map) format. Each line consists of:

Col	Field	Description
1	QNAME	Query (pair) NAME
2	FLAG	bitwise FLAG
3	RNAME	Reference sequence NAME
4	POS	1-based leftmost POSition/coordinate of clipped sequence
5	MAPQ	MAPping Quality (Phred-scaled)
6	CIAGR	extended CIGAR string
7	MRNM	Mate Reference sequence NaMe ('=' if same as RNAME)
8	MPOS	1-based Mate POSistion
9	ISIZE	Inferred insert SIZE
10	SEQ	query SEQuence on the same strand as the reference
11	QUAL	query QUALity (ASCII-33 gives the Phred base quality)
12	OPT	variable OPTional fields in the format TAG:VTYPE:VALUE

Each bit in the FLAG field is defined as:

Chr	Flag	Description
p	0x0001	the read is paired in sequencing

P	0x0002	the read is mapped in a proper pair
u	0x0004	the query sequence itself is unmapped
U	0x0008	the mate is unmapped
r	0x0010	strand of the query (1 for reverse)
R	0x0020	strand of the mate
1	0x0040	the read is the first read in a pair
2	0x0080	the read is the second read in a pair
s	0x0100	the alignment is not primary
f	0x0200	QC failure
d	0x0400	optical or PCR duplicate

The Please check <http://samtools.sourceforge.net> for the format specification and the tools for post-processing the alignment.

BWA generates the following optional fields. Tags starting with ‘X’ are specific to BWA.

Tag	Meaning
NM	Edit distance
MD	Mismatching positions/bases
AS	Alignment score
BC	Barcode sequence
X0	Number of best hits
X1	Number of suboptimal hits found by BWA
XN	Number of ambiguous bases in the referenece
XM	Number of mismatches in the alignment
XO	Number of gap opens
XG	Number of gap extentions
XT	Type: Unique/Repeat/N/Mate-sw
XA	Alternative hits; format: (chr,pos,CIGAR,NM;)*
XS	Suboptimal alignment score
XF	Support from forward/reverse alignment
XE	Number of supporting seeds

Note that XO and XG are generated by BWT search while the CIGAR string by Smith-Waterman alignment. These two tags may be inconsistent with the CIGAR string. This is not a bug.

NOTES ON SHORT-READ ALIGNMENT

Alignment Accuracy

When seeding is disabled, BWA guarantees to find an alignment containing maximum *maxDiff* differences including *maxGapO* gap opens which do not occur within *nIndelEnd* bp towards either end of the query. Longer gaps may be found if *maxGapE* is positive, but it is not guaranteed to find all hits. When seeding is enabled, BWA further requires that the first *seedLen* subsequence contains no more than *maxSeedDiff* differences.

When gapped alignment is disabled, BWA is expected to generate the same alignment as Eland, the

Illumina alignment program. However, as BWA change 'N' in the database sequence to random nucleotides, hits to these random sequences will also be counted. As a consequence, BWA may mark a unique hit as a repeat, if the random sequences happen to be identical to the sequences which should be unqiue in the database. This random behaviour will be avoided in future releases.

By default, if the best hit is no so repetitive (controlled by -R), BWA also finds all hits contains one more mismatch; otherwise, BWA finds all equally best hits only. Base quality is NOT considered in evaluating hits. In paired-end alignment, BWA pairs all hits it found. It further performs Smith-Waterman alignment for unmapped reads with mates mapped to rescue mapped mates, and for high-quality anomalous pairs to fix potential alignment errors.

Estimating Insert Size Distribution

BWA estimates the insert size distribution per 256*1024 read pairs. It first collects pairs of reads with both ends mapped with a single-end quality 20 or higher and then calculates median (Q2), lower and higher quartile (Q1 and Q3). It estimates the mean and the variance of the insert size distribution from pairs whose insert sizes are within interval [Q1-2(Q3-Q1), Q3+2(Q3-Q1)]. The maximum distance x for a pair considered to be properly paired (SAM flag 0x2) is calculated by solving equation Phi((x-mu)/sigma)=x/L*p0, where mu is the mean, sigma is the standard error of the insert size distribution, L is the length of the genome, p0 is prior of anomalous pair and Phi() is the standard cumulative distribution function. For mapping Illumina short-insert reads to the human genome, x is about 6-7 sigma away from the mean. Quartiles, mean, variance and x will be printed to the standard error output.

Memory Requirement

With bwtsw algorithm, 2.5GB memory is required for indexing the complete human genome sequences. For short reads, the '**aln**' command uses ~2.3GB memory and the '**sampe**' command uses ~3.5GB.

Speed

Indexing the human genome sequences takes 3 hours with bwtsw algorithm. Indexing smaller genomes with IS or divsufsort algorithms is several times faster, but requires more memory.

Speed of alignment is largely determined by the error rate of the query sequences (r). Firstly, BWA runs much faster for near perfect hits than for hits with many differences, and it stops searching for a hit with l+2 differences if a l-difference hit is found. This means BWA will be very slow if r is high because in this case BWA has to visit hits with many differences and looking for these hits is expensive. Secondly, the alignment algorithm behind makes the speed sensitive to [k log(N)/m], where k is the maximum allowed differences, N the size of database and m the length of a query. In practice, we choose k w.r.t. r and therefore r is the leading factor. I would not recommend to use BWA on data with r>0.02.

Pairing is slower for shorter reads. This is mainly because shorter reads have more spurious hits and converting SA coordinates to chromosomal coordinates are very costly.

In a practical experiment, BWA is able to map 2 million 32bp reads to a bacterial genome in several minutes, map the same amount of reads to human X chromosome in 8-15 minutes and to the human genome in 15-25 minutes. This result implies that the speed of BWA is insensitive to the size of database and therefore BWA is more efficient when the database is sufficiently large. On smaller genomes, hash based algorithms are usually much faster.

NOTES ON LONG-READ ALIGNMENT

Command '**bwasw**' is designed for long-read alignment. The algorithm behind, BWA-SW, is similar to BWT-SW, but does not guarantee to find all local hits due to the heuristic acceleration. It tends to be faster and more accurate if the resultant alignment is supported by more seeds, and therefore BWA-SW usually performs better on long queries than on short ones.

On 350-1000bp reads, BWA-SW is several to tens of times faster than the existing programs. Its accuracy is comparable to SSAHA2, more accurate than BLAT. Like BLAT, BWA-SW also finds chimera which may pose a challenge to SSAHA2. On 10-100kbp queries where chimera detection is important, BWA-SW is over 10X faster than BLAT while being more sensitive.

BWA-SW can also be used to align ~100bp reads, but it is slower than the short-read algorithm. Its sensitivity and accuracy is lower than SSAHA2 especially when the sequencing error rate is above 2%. This is the trade-off of the 30X speed up in comparison to SSAHA2's -454 mode.

SEE ALSO

BWA website <http://bio-bwa.sourceforge.net>, Samtools website <http://samtools.sourceforge.net>

AUTHOR

Heng Li at the Sanger Institute wrote the key source codes and integrated the following codes for BWT construction: bwtsw <http://i.cs.hku.hk/~ckwong3/bwtsw/>, implemented by Chi-Kwong Wong at the University of Hong Kong and IS <http://yuta.256.googlepages.com/sais> originally proposed by Nong Ge <http://www.cs.sysu.edu.cn/nong/> at the Sun Yat-Sen University and implemented by Yuta Mori.

LICENSE AND CITATION

The full BWA package is distributed under GPLv3 as it uses source codes from BWT-SW which is covered by GPL. Sorting, hash table, BWT and IS libraries are distributed under the MIT license.

If you use the short-read alignment component, please cite the following paper:

Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-60. [PMID: 19451168]

If you use the long-read component (BWA-SW), please cite:

Li H. and Durbin R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics. [PMID: 20080505]

HISTORY

BWA is largely influenced by BWT-SW. It uses source codes from BWT-SW and mimics its binary file formats; BWA-SW resembles BWT-SW in several ways. The initial idea about BWT-based alignment also came from the group who developed BWT-SW. At the same time, BWA is different enough from BWT-SW. The short-read alignment algorithm bears no similarity to Smith-Waterman algorithm any more. While BWA-SW learns from BWT-SW, it introduces heuristics that can hardly be applied to the original algorithm. In all, BWA does not guarantee to find all local hits as what BWT-SW is designed to do, but it is much faster than BWT-SW on both short and long query sequences.

I started to write the first piece of codes on 24 May 2008 and got the initial stable version on 02 June 2008. During this period, I was acquainted that Professor Tak-Wah Lam, the first author of BWT-SW paper, was collaborating with Beijing Genomics Institute on SOAP2, the successor to SOAP (Short Oligonucleotide Analysis Package). SOAP2 has come out in November 2008. According to the SourceForge download page, the third BWT-based short read aligner, bowtie, was first released in August 2008. At the time of writing this manual, at least three more BWT-based short-read aligners are being implemented.

The BWA-SW algorithm is a new component of BWA. It was conceived in November 2008 and implemented ten months later.