Manual Reference Pages - bwa (1)

NAME

bwa - Burrows-Wheeler Alignment Tool

CONTENTS

SYNOPSIS

```
bwa index ref.fa

bwa mem ref.fa reads.fq > aln-se.sam

bwa mem ref.fa read1.fq read2.fq > aln-pe.sam

bwa aln ref.fa short_read.fq > aln_sa.sai

bwa samse ref.fa aln_sa.sai short_read.fq > aln-se.sam

bwa sampe ref.fa aln_sa1.sai aln_sa2.sai read1.fq read2.fq > aln-pe.sam

bwa bwasw ref.fa long_read.fq > aln.sam
```

DESCRIPTION

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 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 all the algorithms, BWA first needs to construct the FM-index for the reference genome (the **index** command). Alignment algorithms are invoked with different sub-commands: **aln**/**samse**/**sampe** for BWA-backtrack, **bwasw** for BWA-SW and **mem** for the BWA-MEM algorithm.

COMMANDS AND OPTIONS

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

Index database sequences in the FASTA format.

OPTIONS:

-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.

mem **bwa mem** [**-aCHMpP**] [**-t** *nThreads*] [**-k** *minSeedLen*] [**-w** *bandWidth*] [**-d** *zDropoff*] [**-r** *seedSplitRatio*] [**-c** *maxOcc*] [**-A** *matchScore*] [**-B** *mmPenalty*] [**-O** *gapOpenPen*] [**-E** *gapExtPen*] [**-L** *clipPen*] [**-U** *unpairPen*] [**-R** *RGline*] [**-v** *verboseLevel*] *db.prefix reads.fq* [*mates.fq*]

Align 70bp-1Mbp query sequences with the BWA-MEM algorithm. Briefly, the algorithm works by seeding alignments with maximal exact matches (MEMs) and then extending seeds with the affine-gap Smith-Waterman algorithm (SW).

If *mates.fq* file is absent and option **-p** is not set, this command regards input reads are single-end. If *mates.fq* is present, this command assumes the *i*-th read in *reads.fq* and the *i*-th read in *mates.fq* constitute a read pair. If **-p** is used, the command assumes the 2*i*-th and the (2*i*+1)-th read in *reads.fq* constitute a read pair (such input file is said to be interleaved). In this case, *mates.fq* is ignored. In the paired-end mode, the **mem** command will infer the read orientation and the insert size distribution from a batch of reads.

The BWA-MEM algorithm performs local alignment. It may produce multiple primary alignments for different part of a query sequence. This is a crucial feature for long sequences. However, some tools such as Picard's markDuplicates does not work with split alignments. One may consider to use option **-M** to flag shorter split hits as secondary.

OPTIONS:

-t *INT* Number of threads [1]

-k *INT* Minimum seed length. Matches shorter than *INT* will be missed. The alignment speed is usually insensitive to this value unless it significantly deviates 20. [19]

-w *INT* Band width. Essentially, gaps longer than *INT* will not be found. Note that the maximum gap length is also affected by the scoring matrix and the hit length, not solely determined by this option. [100]

-d *INT* Off-diagonal X-dropoff (Z-dropoff). Stop extension when the difference between the best and the current extension score is above $|i-j|*A+INT$, where i and j are the current positions of the query and reference, respectively, and A is the matching score. Z-dropoff is similar to BLAST's X-dropoff except that it doesn't penalize gaps in one of the sequences in the alignment. Z-dropoff not only avoids unnecessary extension, but also reduces poor alignments inside a long good alignment. [100]

-r *FLOAT* Trigger re-seeding for a MEM longer than *minSeedLen***FLOAT*. This is a key heuristic parameter for tuning the performance. Larger value yields fewer seeds, which leads to faster alignment speed but lower accuracy. [1.5]

-c *INT* Discard a MEM if it has more than *INT* occurence in the genome. This is an insensitive parameter. [10000]

-P In the paired-end mode, perform SW to rescue missing hits only but do not try to find hits that fit a proper pair.

-A *INT* Matching score. [1]

-B *INT* Mismatch penalty. The sequence error rate is approximately: {.75 * exp[-log(4) * B/A]}. [4]

-O *INT* Gap open penalty. [6]

-E *INT* Gap extension penalty. A gap of length k costs O + k*E (i.e. **-O** is for opening a zero-length gap). [1]

-L *INT* Clipping penalty. When performing SW extension, BWA-MEM keeps track of the best score reaching the end of query. If this score is larger than the best SW score minus the clipping penalty, clipping will not be applied. Note that in this case, the SAM AS tag reports the best SW score; clipping penalty is not deducted. [5]

-U *INT* Penalty for an unpaired read pair. BWA-MEM scores an unpaired read pair as scoreRead1+scoreRead2-*INT* and scores a paired as scoreRead1+scoreRead2-insertPenalty. It compares these two scores to determine whether we should force pairing. [9]

-p Assume the first input query file is interleaved paired-end FASTA/Q. See the command description for details.

-R *STR* Complete read group header line. '\t' can be used in *STR* and will be converted to a TAB in the output SAM. The read group ID will be attached to every read in the output. An example is '@RG\tID:foo\tSM:bar'. [null]

-T *INT* Don't output alignment with score lower than *INT*. This option only

-T *INT* Don't output alignment with score lower than *INT*. This option only affects output. [30]

-a Output all found alignments for single-end or unpaired paired-end reads. These alignments will be flagged as secondary alignments.

-C Append append FASTA/Q comment to SAM output. This option can be used to transfer read meta information (e.g. barcode) to the SAM output. Note that the FASTA/Q comment (the string after a space in the header line) must conform the SAM spec (e.g. BC:Z:CGTAC). Malformated comments lead to incorrect SAM output.

-H Use hard clipping 'H' in the SAM output. This option may dramatically reduce the redundancy of output when mapping long contig or BAC sequences.

-M Mark shorter split hits as secondary (for Picard compatibility).

-v *INT* Control the verbose level of the output. This option has not been fully supported throughout BWA. Ideally, a value 0 for disabling all the output to stderr; 1 for outputting errors only; 2 for warnings and errors; 3 for all normal messages; 4 or higher for debugging. When this option takes value 4, the output is not SAM. [3]

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. (Disabled since 0.6.x)

-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> [mate.fq]

Align query sequences in the *in.fq* file. When *mate.fq* is present, perform paired-end alignment. The paired-end mode only works for reads Illumina short-insert libraries. In the paired-end mode, BWA-SW may still output split alignments but they are all marked as not properly paired; the mate positions will not be written if the mate has multiple local hits.

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 version 1, 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.

By default, if the best hit is not highly 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 the paired-end mode, BWA pairs all hits it found. It further performs Smith-Waterman alignment for unmapped reads to rescue reads with a high erro rate, 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, 5GB memory is required for indexing the complete human genome sequences. For short reads, the **aln** command uses ~3.2GB memory and the **sampe** command uses ~5.4GB.

Speed

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

The 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.

CHANGES IN BWA-0.6

Since version 0.6, BWA has been able to work with a reference genome longer than 4GB. This feature makes it possible to integrate the forward and reverse complemented genome in one FM-index, which speeds up both BWA-short and BWA-SW. As a tradeoff, BWA uses more memory because it has to keep all positions and ranks in 64-bit integers, twice larger than 32-bit integers used in the previous versions.

The latest BWA-SW also works for paired-end reads longer than 100bp. In comparison to BWA-short, BWA-SW tends to be more accurate for highly unique reads and more robust to relative long INDELs and structural variants. Nonetheless, BWA-short usually has higher power to distinguish the optimal hit from many suboptimal hits. The choice of the mapping algorithm may depend on the application.

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 BWA-backtrack algorithm, please cite the following paper:

Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

If you use the BWA-SW algorithm, please cite:

Li H. and Durbin R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics, 26, 589-595. [PMID: 20080505]

If you use the fastmap component of BWA, please cite:

Li H. (2012) Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. Bioinformatics, 28, 1838-1844. [PMID: 22569178]

The BWA-MEM algorithm has not been published yet.

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.

The BWA-MEM algorithm is based on an algorithm finding super-maximal exact matches (SMEMs), which was first published with the fermi assembler paper in 2012. I first implemented the basic SMEM algorithm in the **fastmap** command for an experiment and then extended the basic algorithm and added the extension part in Feburary 2013 to make BWA-MEM a fully featured mapper.