segemehl

Download

Download latest segemehl version (0.3.4)

Changes (in version 0.3)

Important changes

- now supports extended cigar strings (using = and X for matches and mismatches)
- collinear split alignments now reported in one SAM record (using N’s in the cigar)
- unaligned reads are reported in the sam file
- BAM output supported
- automatic generation of bed files for splice junctions
- summary of bisulfite converted C's with haarz.x
- summary of splice junctions with haarz.x
- supports read groups
- faster

Version history

Release	Timestamp	Changes
0.3.4	**20181225-204200**	output fastq ID in SAM instead of whole name (for old behaviour use flag "-f")
0.3.3	**20181225-182132**	Bugfix: flag error in empty alignments
0.3.2	**20181111-171849**	Bugfix: output of empty alignments in bam mode
0.3.1	**20181005-161011**	Output of haarz.x adjusted
0.3.0	**20181004-163244**	Initial release of 0.3

Go here for old versions

Introduction

segemehl is a software to map short sequencer reads to reference genomes. segemehl implements a matching strategy based on enhanced suffix arrays (ESA). segemehl accepts fasta and fastq queries (gzip’ed and bgzip'ed). In addition to the alignment of reads from standard DNA- and RNA-seq protocols, it also allows the mapping of bisulfite converted reads (Lister and Cokus) and implements a split read mapping strategy. The output of segemehl is a SAM or BAM formatted alignment file. In the case of split-read mapping, additional BED files are written to the disc. These BED files may be summarized with the postprocessing tool haarz. In the case of the alignment of bisulfite converted reads, raw methylation rates may also be called with haarz.

In brief, for each suffix of a read, segemehl aims to find the best-scoring seed. Seeds might contain insertions, deletions, and mismatches (differences). The number of differences allowed within a single seed is user-controlled [parameter

-D, -- differences] and is crucial for the runtime of the program. Subsequently, seeds that undercut the user-defined E-value [parameter -E, --evalue] are passed on to an exact semi-global alignment procedure. Finally, reads with a minimum accuracy of [-A, --accuracy] percent are reported to the user. Detailed explanations are given below.

Quick start

Dependencies

Segemehl now supports reading and writing bam files. Therefore, it is necessary to have `htslib` installed. In order to complile, segemehl will need the c header files of `htslib`. On some distributions it is therefore necessary to also install the `htslib` developer package (e.g. hts-devel).

Installation

Extract the archive and hit

```
$ make all
```

to built `segemehl.x` and `haarz.x`.

Installation

Please note that segemehl makes uses of pkg-config to locate the `htslib`. In some cases you might be required to set the pkg-config configuration path manually.

```
   export PKG_CONFIG_PATH=<path-to-pkg-config-configuration-
files>
```

Generation of indices

First, it is necessary to build the index structures (i.e. the ESA) for the reference sequences (i.e. chromosomes, genomes) you want to match against. Please make sure that the reference sequences are provided in fasta format. To build the index structure, e.g., for some fasta file chr1.fa, you call segemehl with option `-x`. To build the index for a whole genome stored in a single fasta, e.g. the human genome version hg38, type

```
$ ./segemehl.x -x hg38.idx -d hg38.fa
```

or alternatively for a single chromosome

```
$ ./segemehl.x -x chr1.idx -d chr1.fa
```

If you want to build indices for multiple fasta files, say chr1.fa chr2.fa chr3.fa, you simply type

```
$ ./segemehl.x -x chr1_2_3.idx -d chr1.fa chr2.fa chr3.fa
```

Please note that building and storing of index structures needs memory. The index for the human chromosome 1, e.g., takes a about 4 GB of disk space. Building the index needs approximately 6GB of RAM. Please make sure that your machine has at least 8 GB of RAM available. To build the index of a whole mammalian genome such as mouse or human we recommend a machine with * more than 64GB * of space.

Mapping of reads

To align your reads (provided in a fasta or fastq file) just type

```
$ ./segemehl.x -i chr1.idx -d chr1.fa -q myreads.fa > mymap.sam
```

All results will be written to the file mymap.sam. If the index was build from mutliple fasta files, please provide the fasta files in the same order used when building the index. Otherwise, you will recieve an error message (md5 keys do not match).

Input

Alignment of single-end data

Reads can be supplied in fasta or fastq format. Reference genomes should be supplied in fasta format. For reference genomes with multiple chromosomes, it is recommended to use multiple fasta files, i.e., one file that contains all chromosomes in fasta format. In this way, the order of chromosomes - important for the matching with the suffix array - is ensured. The reference or database sequence(s) are provided using the `[-d, --database <file>]` option. Reads or queries are provided using the `[-q, --query <file>]` option (also see Quick Start).

Paired-end and mate pair sequencing

For paired-end or mate pair mapping, two fastq or fasta files are required. The first mate is provided with the `[-q,--query <file1>]` parameter, the second mate pair/paired end is provided with `[-p, --mate <file2>]` . The parameter `[-I, --maxinsertsize <n>]` merely influences the internal algorithmics of the read mapper. While it has an important influence on the mapping time of paired-end or mate pair reads, it should not have an impact on the output. It is especially important to notice that `[-I, --maxinsertsize <n>]` is not a filter and read pairs with a bigger insert size will not be purged.

Generation of database indices

The generation of a new index for the database `[-d, --database <file>]` can be triggered via the options `[-x, --generate <file>]` and `[-y, --generate2`

`<file>]` . The latter option is only used for the generation of indicies for the alignment for bisulfite converted DNA (see below).

Gzip’ed files

You may also supply gzip’ed fastq and fasta files. Also bgzf files are supported.

Read groups

All aligned reads will be assigned the read group "A1". Alternatively, a string with a user-defined read group ID may be provided via `[-g, --readgroupid]` . The read group ID can also be provided in a file containing a SAM format @RG header.

Threads

Parallel threads `[-t, --threads <n>]` will make matching much faster on machines with multiple cores. Use them! To start segemehl with, e.g., with 8 parallel threads, type -t 8.

Summary

Short	Long	Remarks
-d	--database [<file>]	list of filename(s) of fasta database sequence(s)
-q	--query	filename of query sequences
-p	--mate	filename of paired end sequences
-i	--index	filename of database index
-j	--index2	filename of second database index
-x	--generate	generate db index and store to disk
-y	--generate2	generate second db index and store to disk
-G	--readgroupfile	filename to read @RG header (default:none)
-g	--readgroupid	read group id (default:none)
-t	--threads	start threads (default:1)
-f	--fullname	write full fastq/a name to SAM

Alignment options modes

Controlling sensitivity, specificity and accuracy

In principle, segemehl knows five different alignment thresholds controlling sensitivity, specificity, accuracy, and multiple matches. The parameters `[-M, --maxocc]` , `[-E, --evalue]` , and `[-D, --differences]` control the seed matching of segemehl. Especially the parameter `-D` has a big influence on the runtime. Note that for longer reads (>=100bp) `-D 0` should be sufficient, since segemehl will be able to find enough seeds to recover the optimal alignment in most cases. `[-M, --maxocc <n>]` controls the number of multiple matches. Each seed that passes the score based E-value `[-E, --evalue <float>]` criteria will be aligned using a semi-global alignment algorithm. Read alignments with an accuracy equal to or better than `[-A, --accuracy <float>]` will be reported. The hit strategy `[-H, --hitstrategy <n>]` may be changed form best-only (default: -H 1) to all (-H 0). The table below explains the options and parameters in greater detail.

Additional alginment modes

For the alignment of bisulfite converted DNA sequences using the option `[-F, --bisulfite <n>]` and the split read alignment mode `[-S, --split [basename]]` please also see the sections below.

Summary

Short	Long	Remarks
-M	--maxocc <n>	This option controls the number of multiple hits. Specifically, in the seed alignment step, all seeds with more than -M occurences will be discarded.
-E	--evalue <float>	segemehl only considers seeds with a maximum score-based E-value. This E-value needs to be adjusted when exceptionally short reads or their fragments (10-20) shall be aligned. Increasing the E-value also leads to a higher sensitivity of the read mapping.
-D	--differences <n>	Seeds in segemehl are local alignments with mismatches, insertions, and deletions. To increase the sensitivity, this parameter may be set to 2. For lower sensitivity set -D 0. Note that setting -D to higher values has significant impact on the runtime Please note that for mapping long reads (>100bp) -D 0 should be sufficient in most cases.
-A	--accuracy <float>	This option controls the minimum alignment accuracy (in percent). All reads with a best alignment below this threshold will be discarded.
-H	--hitstrategy <n>	This option controls the minimum alignment accuracy (in percent). All reads with a best alignment below this threshold will be discarded. By default, only the best-scoring alignments will be shown for each read. To show all alignments that pass the -A criterion, use -H 1.
-S	--splits [basename]	triggers the split read alignment mode for reads that failed the attempt of collinear alignment. The algorithm will also consider circular alignments and alignments in trans. Optionally, a basename for the split files can be given. If it is not given, the basename of the first query file will be used.
-F	--bisulfite	alignment of bisulfite converted DNA sequences generated with the methylC-seq (Lister et al.) (=1) or bs-seq (Cokus et al.) protocol (=2) (default:0)

Output

The default output of segemehl is the SAM format. It is written to `stdout`. Alternatively, the output can be redirected to a file via `[-o, --output <file>]`. Since version 0.3 segemehl also supports the output of unsorted BAM files. For split read and bisulfite alignments please also see the sections below.

BAM format

The output of the BAM format instead of SAM can be turned on via the option `[-b, --bamabafixoida]`. Please note, that this might take a bit longer.

Extended vs. simple cigar string

Since version 0.3 segemehl uses the extended cigar string, i.e. a cigar string distinguishes between matches ('=') and mismatches ('X'). In the simple version of the SAM cigar string, both edit operations are denoted by 'M'. To ensure compatibility with downstream programs, the the use of the extended cigar may be turned off with `[-e, --briefcigar]`. Alternatively, a multi edit operation string stored as a custom key/value pair XE:Z: may be attached to each SAM alignment using the flag `[-V, --MEOP]`.

Unmapped reads

Note, that segemehl now reports unmapped reads within the output SAM. You may use the option `[-u, --unmatched <filename>]` to dump unmatched fasta or fastq reads to a user defined file.

Progress bar

The progress bar which is displayed on the `stderr` device can be shown to monitor the alignment run on the shell. By default, the progress bar is turned off to save IO and CPU time.

Binning and Sorting

The options supporting binning and sorting are *deprecated*. To ensure backward compatibility, however, it is still possible to use a bining option `[-B, --filebins <string>]` where is a user supplied base name. segemehl will automatically generate a number of temporary file bins and then merge them to one sam file for each chromosome. If combined with the option `[-O, --order]` the output will be sorted. This combination may facilitate the sorting problem of very large output files. The `[-O, --order]` option can be used without binning. This, however, may take significantly longer.

Summary

Short	Long	Remarks
-b	--bamabafixoida	output a BAM file instead of a SAM file (takes longer)
-s	--showprogress	show a progress bar
-o	--outputfile <file>	a filename to write the SAM or BAM to (default: stdout)
-u	--nomatchfilename <file>	filename for unmatched reads (default:none)
-e	--briefcigar	brief cigar string (M vs X and =)
-s	--progressbar	show a progress bar
-V	--MEOP	output MEOP field for easier variance calling in SAM (XE:Z:)
-B	--filebins	file bins with basename for easier data handling (default:none)

Split read alignments

segemehl supports (multiple) split read mapping. To activate the split read mapping, it is only required to give the option `[-S, --splits]` . The split read alignment will be triggered in all cases a read can not be aligned because of insufficient ac- curacy `[-A, --accuracy <n>]` . Please note that the alignment accuracy has a great influence on which reads are actually subjected to the split read alignment. The higher the accuracy parameter is set, the more reads will be probed for a split read alignment. The split read strategy employed by segemehl has two steps. In the first steps, all seeds that pass the `[-E, --evalue <float>]` and `[-M, --maxocc]` criteria will be chained using a greedy chaining approach. This chain of seeds guides a modified local transition alignment. Each split of this alignment needs to have a minimum score `[-U, --minfragscore <n>]` and a minimum length `[-Z, --minfraglen]` . A split read alignment is accepted if the local alignment covers at least `[-W, --minsplicecover <n>]` percent of the original read. Split reads are reported in the SAM format. In contrast to the current SAM format, segemehl reports the split reads using custom SAM tags. Long reads more than 1 split. Therefore it is more convinient to not only store the next but also the predecessing fragment. Below, the custom SAM tags are explained in detail.

Additional output files for the split read mode

In case segemehl was run in the split read mode `[-S, --split]` additional output files holding information on the predicted splits are generated. Note, that in some cases and depending on the `--accuracy` thresholds also shorter insertions and deletions are reported here.

File	Type	Contents
*.sngl.bed	bed	The bed file contains all single splice events predicted in the split read alignments. Start and end position indicate the genomic range of the predicted intron. The name has the format (read-group;type;read-name;mate-status), the bed score is the alignment score of the respective alignment. The type is either 'R' (in case of a regular, collinear split), 'C' (circular split) or 'B' (backsplice).
*.mult.bed	bed	Segemehl is able to align the reads with multiple splits. Thus, this bed file contains all splice events of a read. The start and end positions indicate the nucleotide after the first split (i.e. the beginning of the first intron) and the nucleotide before the last split (i.e. the end of the last intron), respectively. The name and score are equivalent to the one in the *.sngl file described above. The following fields 7 & 8 (thickStart and thickEnd) should be the identical to fields 2 & 3. Field 9 holds the color information for the item in RGB encoding (itemRGB). Field 10 (blockCount) indicates the number of splits represented by the BED item. Field 11 is a comma separated list of the intron sizes (blockSizes). Field 12 is the comma separated list of intron starts (blockStarts).
*trns.txt	custom	The custom text file contains all single split alignments predicted to be in trans, i.e. split alignments that are located on different chromosomes and/or different strands. For each pair of split alignments the first two columns in the file have the format (ref-chr,ref-pos,ref-strand,start-in-read,align-length,algin-edist,score).

Summarizing single splits

The program `haarz.x split` can be used to summarize single splits reported by segemehl. The call

```
$ ./haarz.x split -f my.sngl.bed > my.sum.bed
```

In the output file, the first six columns of each line have the format (chromosome;intron-start;intron-end;number-of-supporting-splits;median-map-quality;strand). The field number-of-supporting-splits holds the number of reads that were found to support this split junction.

Depending on the input, further columns are attached to each line of the summary file. It is possible to give multiple sngl.bed files to `haarz.x` :

```
$ ./haarz.x split -f my.first.sngl.bed my.second.sngl.bed >
my.sum.bed
```

Here, the output lines have the following format:

```
sgr19	18602674		18605680		5	27.000000	.	4		1
```

The last two integers are the number of split reads supporting this junction in each of the individual files, i.e. 4 split reads in my.first.sngl.bed and 1 split read in my.second.sngl.bed.

Annotation

`haarz.x` supports a simple annotation of split reads. The call

```
$  ./haarz.x split -f my.first.sngl.bed my.second.sngl -a my.gff3	
> my.sum.bed
```

will have the output format:

```
sgr19	18602674		18605680		5	27.000000	.	4		1	ID=Nfu_g_1_016245	
annotation=cytoplasmic linker associated protein
2																								biotype=protein_coding
db_xref=ENSDARG00000020345			gene=CLASP2	geneid=Nfu_g_1_016245	
source=EVM_PASA ID=Nfu_t_1_038940																		
Parent=Nfu_g_1_016245	biotype=protein_coding	geneid=Nfu_g_1_016245	
proteinid=Nfu_p_1_038940		transcriptid=Nfu_t_1_038940															
ID=Nfu_t_1_038940.e3	Parent=Nfu_t_1_038940	ID=Nfu_t_1_038940.c3		
Parent=Nfu_t_1_038940
```

The annotation is attached to the columns with the split read counts. In this case, two overlaps have been found. Via the options `-M LARGEST` or `-M SMALLEST` it is possible to reduce the number of annotation items to the largest overlapping or the smallest overlapping interval, respectively. The option `[-A, --attributes]` allows the user to select specific attributes in the GFF file to be used for the annotation. The call

```
$ ./haarz.x  split -f my.first.sngl.bed my.second.sngl -a my.gff3
-M LARGEST -attributes gene source > my.sum.bed
```

yields the output

```
sgr19	18602674		18605680		5	27.000000	. 4	1	gene=CLASP2	
geneid=Nfu_g_1_016245	source=EVM_PASA
```

Note, that haarz has performed a prefix match with the pattern 'gene' and thus selected the attributes 'gene' and 'geneid' for output.

Minimum number of splits

Using the parameter `[-m, --minsplit <n>]` , the user can control the total minimum number of supporting splits, required for the output in the summarized split file.

Summary

Short	Long	Remarks
-f	--files [<file>]	list of path/filename(s) of bed files with split info (s)
-m	--minsplit <n>	minimum total split number (all samples) of junction (default:5)
-q	--minqual <f>	minimum median quality of junction (default: 25.000000)
-a	--annotationfiles <file> [<file>]	list of path/filename(s) of GFF file (s))
-M	--ovlmode <string>	annotation mode LARGEST
-A	--attributes <string>	attributes that shall be selected for overlap annotation

Bisulfite converted DNA mapping

segemehl supports mapping of bisulfite-treated sequencing data where DNA fragments are treated with sodium bisulfite which results in the conversion of unmethylated cytosines into uracils while methylated cytosines remain unchanged. With bisulfite sequencing, it is possible to capture DNA methylation genomewide in an unbiased fashion with single-base resolution and is hence considered 'gold standard'. We support both currently used protocols for the construction of the bisulfite-treated libraries, namely methylC-seq (Lister et al., 2009) and BS-seq (Cokus et al., 2008) with the option [-F 1, --bisulfite 1] and [-F 2, --bisulfite 2], respectively. The sequencing reads exhibit asymmetric bisulfite-related mismatches and suffer from an effective reduction of the alphabet size in unmethylated regions. segemehl uses a hybrid approach with a seed-based search on a reduced alphabet in conjunction with a bisulfite-sensitive semi-global alignment to provide highly sensitive mappings. Note that both of the steps consider bisulfite-related mismatches, i.e., a thymine in the read aligned to a genomic cytosine, as matches which will appear as such in the alignment output as well.

Generation of bisulfite indices

To enable the seed search on a reduced alphabet, it is necessary to generate two bisulfite indices, one (C-to-T) and one (G-to-A) converted ESA by using

```
$ ./segemehl.x -x mygenome.ctidx -y mygenome.gaidx -d mygenome.fa
-F [1 or 2]
```

Note that the bisulfite option can be set to either 1 or 2 since segemehl uses the same bisulfite indices for both library preparation protocols.

Mapping

To map the reads against the indicies you run.

```
$ ./segemehl.x -i mygenome.ctidx -j mygenome.gaidx -d mygenome.fa
-q myreads.fa -t 20
              -o mymap.sam -F [1 or 2]
```

Warning

The merging step is IO intensive and may consume a signficant amout of time.

In the SAM- formatted output, the custom tag XB:Z indicates the genomic strand from which the read was derived according to the mapping, i.e., plus and minus strand in case of XB:Z:CT and XB:Z:GA, respectively.

Counting of methylated and un-methylated cytosines

The tool `haarz.x methylcall` may be used to count the number of methylated and un-methylated cytosines from the alignments obtained with segemehl's bisulfite mode. The tool also calculates a simple methylation rate, i.e., C/(C+T), as confidence measure of each methylation call. The program generates a file in the vcf format.

Data preparation

The program expects a sorted bam file as an input. In case you have chosen to write the segemehl alignments to a sam file, you can run

```
$ samtools view -b mymap.sam > mymap.bam
```

to convert the sam file into a bam. Subsequently, the new bam file needs to be sorted and indexed.

```
$ samtools sort mymap.bam --threads 20 -T /my/temp/dir -o mymap.tmp
$ mv mymap.tmp mymap.bam
$ samtools index mymap.bam;
```

Please note, that `samtools sort` uses temporary files. The option `-T /my/temp/dir` tells samtools where to store these files, i.e. a location with sufficient disk space. By default, the index will be stored in the same directory as the `mymap.bam` file. This is where `haarz.x` is expecting it.

Program call

You may run

```
$ ./haarz.x callmethyl -t 10  -d mygenome.fa -b mymap.bam >
mycalls.vcf
```

to count methylated and un-methylated cytosines in the sorted and indexed `mymap.bam` with 10 threads. Please note, that haarz also requires the reference genome `mygenome.fa` which was used to align the reads with segemehl. The output is written to `mycalls.vcf`. The counting of methylated and un-methylated cytosines may be restricted to uniquely mapped reads only with the `[-u, --uniqueonly]` option.

Summary

Short	Long	Remarks
-d	--database <file> [<file>]	list of path/filename(s) of fasta database sequence(s) (default:none)
-b	--bam <file>	path/filename of sorted and indexed (!) bamfile (default:none)
-t	--threads <n>	start threads (default:10)
-o	--output <file>	path/filename of output file (will be sorted) (default:none)
-u	--uniqueonly	only use uniquely mapped reads

SAM format output

As mentioned above segemehl uses the extended cigar string format.

Warning

The extended cigar string format is used to distinguish mismatches and matches and to indicate collinear split alignments. Circular split alignments and other non collinear alignments are stored in multiple alignment records

General tags

Tag	Remarks
HI:i:	Number of current hit (zero-based)
NH:i:	Number of total hits for this read
NM:i:	Edit distance of this alignment
MD:Z:	MD string
RG:Z:	read group
YZ:Z:	mapping type (0: normal, 1: circular, 2:backsplice, 4: strand-switch, 8: ref-switch)

Seed tags

Tag	Remarks
ZE:A:	E-value of the best seed alignment
ZI:A:	number of matches for the best seed alignment
ZM:A:	strandedness of the best seed alignment (0: fwd, 1:rc)
ZS:i:	start of seed sequence (in the read)
ZL:i:	length of seed
ZR:i:	position of best seed alignment on the reference
ZP:i:	reference sequence (chromosome) for best seed alignment

split read tags

Tag	Remarks
XX:i:	begin of aligned splits in the read
XY:i:	end of aligned splits in the read
XS:i:	predicted strand of transcript ('+':fwd, '-':rc) [1]
XM:B:	partial edit distances of all split alignments represented by this SAM record
XL:B:	partial lengths of edit distances of all split alignments represented by this SAM record
XJ:i:	total number of split alignments for the whole read
XH:i:	number of split alignments within this SAM record
XI:i:	consecutive number of the first split alignment within this SAM record
XP:Z:	location of the preceding split alignment for this read
XC:Z:	location of the succeding split alignment for this read

The flags XP:Z: and XC:Z: are encoded as character strings and the values are separated by the comma character (","). The format is

```
(rname ,pos ,strand , len , err, qual ;)+
```

where rname, pos and strand give the location of the previous/following split alignment in the reference. Furthemore, len denotes the length of the alignment on the reference. The fields err and qual inform about the number of errors and the quality of the previous/following split alignment.

Reducing memory requirements

To reduce the memory consumption, you can also run segemehl chromosome-wise as follows.

```
$ ./segemehl.x -i chr1.idx -d chr1.fa -q myreads.fa >
mymap_chr1.sam
$ ./segemehl.x -i chr2.idx -d chr2.fa -q myreads.fa >
mymap_chr2.sam
```

Afterwards, you can merge the sam files (e.g. using MergeSamFiles from Picard tools [2], update the SAM tags and enforce best-only (if necessary) using our tool processMerged.pl (available on segemehl website) by typing

```
java -jar MergeSamFiles.jar QUIET=true MSD=true SO=queryname
    O=/dev/stdout I=mymap_chr1.sam I=mymap_chr2.sam | samtools
view -h - |
    perl processMerged.pl -b > mymap.sam
```

By running segemehl on each human chromosome separately, its peak memory consumption is reduced to around 6 GB of RAM.

Adapter and poly-A clipping

Use of clipping options

The adapter clipping support has been discontinued. In case you still require this feature, please contact the authors.

Licence

segemehl version 0.3 is distributed under the GPLv3.

Complaint department

steve at bioinf dot uni-leipzig dot de

Citations

Please cite at least one of the following papers when you are using our software

- Hoffmann S, Otto C, Kurtz S, Sharma CM, Khaitovich P, Vogel J, Stadler PF, Hackermueller J: "Fast mapping of short sequences with mismatches, insertions and deletions using index structures", PLoS Comput Biol (2009) vol. 5(9) pp. e1000502
- Hoffmann S, Otto C, Doose G, Tanzer A, Langenberger D, Christ S, Kunz M, Holdt L, Teupser D, Hackermueller J, Stadler PF: "A multi-split mapping algorithm for circular RNA, splicing, trans-splicing, and fusion detection", Genome Biology (2014) 15:R34
- Otto C, Stadler PF, Hoffmann S: "Lacking alignments? The next-generation sequencing mapper segemehl revisited", Bioinformatics (2014) 30:1837-43.

For the Bisulfite option:

- Otto C, Stadler PF, Hoffmann S: "Fast and sensitive mapping of bisulfite-treated sequencing data", Bioinformatics (2012) 28:1698-1704

---

1. If strand is unknown, XS:i: is set to '+' and the additional flag 'YQ:A' is set.
2. http://picard.sourceforge.net/