A segemehl manual (version 0.1.7; rev 1)

S. Hoffmann, C. Otto, C. Sharma,
J. Hackermueller, S. Kurtz & P.F. Stadler

Email steve@bioinf.uni-leipzig.de
Address Transcriptome Bioinformatics Group and Interdisciplinary Centre for Bioinformatics, Leipzig, Germany

1 Introduction

segemehl is a software to map short sequencer reads to reference genomes. Unlike other methods, segemehl is able to detect not only mismatches but also insertions and deletions. Furthermore, segemehl is not limited to a specific read length and is able to map primer- or polyadenylation contaminated reads correctly. segemehl implements a matching strategy based on enhanced suffix arrays (ESA). segemehl now supports the SAM format, reads gzip'ed queries to save both disk and memory space and allows bisulfite sequencing mapping and split read mapping.

2 The matching model

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.

3 Quick start

To build segemehl from source, you simply download the archive and extract it with

```
> tar -xvzf segemehl_0_*.tar.gz
```

subsequently go to the new directory and type

```
> make
```

First, you have to build the index structures for the reference sequences (i.e. chromosomes, genomes) you want to match against. Please make sure that the reference are provided in fasta format. To build the index structure, e.g., for some fasta file chr1.fa, you call segemehl with option -x

```
> ./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 6 GB of RAM. Please make sure that your machine has at least 8 GB of RAM available. Machines with 32 GB of RAM, e.g., allow building of a single index for half of the human chromosomes at once.

Matching is just as easy. To match your reads (provided in a single multiple fasta 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). If you are sure you have supplied a correct pair of index and reference sequence you can continue and the md5 field will be automatically updated.

3.1 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[1], 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.

4 Input

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 chromsomes 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).

4.1 Paired-end and mate pair sequencing

For paired-end or mate pair mapping, two fastq or fasta files are required. The first read file <file1> is provided with the [-q,--query <file1>] parameter, the second mate pair/paired end file <file2> is provided with the [-p, --matefile <file2>].

[1] http://picard.sourceforge.net/

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.

4.2 Gzip'ed files

The current version of segemehl understands gzip'ed files. Hence, you may supply gzip'ed fastq and fasta files, too, to reduce your space requirements.

5 Output

The standard output is the SAM format. If an output file <outfile> is not specified via the [-o,--output <outfile>] option the results will be dumped to the stdout.

5.1 SAM format, Binning and Sorting

To faciliate post-processing of the data, it is possible to use a bining option [-B, --filebins <string>] where <string> 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.

5.2 Unmatched reads

Note that segemehl does not follow the SAM format in terms of unmatched reads. Instead, the option [-u, --unmatched <filename>] will dump all unmatched fasta or fastq reads to the file <filename>.

6 Alignment parameters

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 long reads (i.e. 454 reads) -D 0 is absolutely 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 <float> [-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.

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 is totally 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>	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.

7 Threads

Parallel threads [-t, --threads <n>] will make matching much faster on machines with <n> multiple cores. Use them! To start segemehl with, e.g., with 8 parallel threads, type -t 8. Using 8 threads, ten million reads with 35 bp can be matched in 30 minutes.

8 Adapter and polyA clipping

Segemehl has a build-in function for adapter clipping and polyA clipping. Adapters may be clipped at the 5′- and 3′-ends of the reads with the options [-P, --prime5 <string>] and [-Q, --prime3 <string>] respectively, where <string> is the adapter sequence. Additionally, for cDNA-reads it is possible to automatically clip polyA-tails [-T, --polyA]. All clipping options may be combined. In brief, for some adapter of length l the clipping is done using a local alignment with the $1.2 * l$ last or first nucleotides of the read. The adapter is clipped if a minimum alignment accuracy is achieved. In the case of polyA-clipping, the best local alignment needs to be in close vicinity to the 3′-end of the read. Again, the polyA signal is clipped if a minimum accuracy is achieved. The user can control the clipping accuracy using the parameter [-R, --clipacc <n>], where <n> is an integer value between 0 and 100. The default is set to 70%. In case the adapter or polyA-tail could not be clipped, segemehl will attempt to map the read without the clipping.

All adapters and polyA-Signals will be soft-clipped, i.e., potential polyA- or adapter sequences will simply be excluded from the alignment steps but not physically removed. Soft-clipped parts of the queries are flagged with an ′S′ in the cigar string of the SAM output. If you want hard-clipping, i.e., the removal of the adapter sequences, the parameter [-C,--hardclip] can be used. Split reads are an exception! Like all the other reads also split reads are clipped prior to any alignment with the reference. However, since the split read method uses a local alignment method (see below), soft-clipped parts of the sequence are not shown in the SAM output.

8.1 Automatic 3′-adapter detection

Segemehl offers automatic 3′-adapter detection. This is achieved using an 3′-kmer assembly approach. In brief, overrepresented k-mers occuring at the ends of the reads are puzzled together using the overlap information. This option is meant to help in cases where the adapter information can not be retrieved. After the k-mer assembly, segemehl will ask the user if the clipping should be performed with the identified adapter. The automatic primer detection can be turned on with [-Y, --autoclip].

9 Split reads 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 accuracy [-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 fragment (a.k.a. split or segment) 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 454 cDNA reads may have 5 or more splits. 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.

Please note that the aligner reports the positions of 5′- and 3′-ends of the splits. The fields XS:i and XT:i inform on the strandiness of the previous and next splits. Specifically, the XV:i field will always yield the same value as the POS field (see SAM format specification) of the respective next fragment (5′-end). In contrast, XU:i holds the 3′-end of the previous split. Hence, this is the POS field *plus* the alignment length of the previous split.

custom tag	**explanation**
XX:i	start of current split in the query
XY:i	end of the current split in the query
XQ:i	number of the current split
	fields for the previous split
XP:Z	reference sequence name to which the previous split maps
XU:i	position of the 3′-end of the prevous split on XP:Z:
XS:i	strandiness of the previous split (0 = minus, 64 = plus)
	fields for the next split
XC:Z	reference sequence name to which the next split maps
XV:i	position of 5′-end of the next split on XC:Z
XT:i	strandiness of the next split (0 = minus, 32 = plus)

10 Bisulfite 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 genome-wide 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. 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 chr1.ctidx -y chr1.gaidx -d chr1.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.

Accordingly, the mapping can be done in similar manner but uses two consecutive mapping runs, each against one of the bisulfite indices.

```
> ./segemehl.x -i chr1.ctidx -j chr1.gaidx -d chr1.fa -q myreads.fa
             -o mymap.sam -F [1 or 2]
```

Due to the default hit strategy (best-only), it is necessary to merge both mapping runs afterwards and hence an output file must be provided. 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.

We provide a simple methylation caller which is based on majority voting and uses the output generated by the mpileup tool from the samtools package. In addition to the majority vote, the tool calculates the methylation rate, i.e., C/(C+T), as confidence measure of each methylation call. The methylation rate may also be directly used as a measure. To get further information on the required input and output as well as on the majority voting approach, just type

```
> perl callMajorMethyl.pl
```

Note that the methylation caller is currently limitted to bisulfite sequencing data generated by the methylC-seq protocol. The script is also available on the segemehl website.

11 Extracting Splice Junctions in SAM files

To extract splice site pairs, i.e., pairs of donors and acceptors you may want to use *testrealign*. **Note that testrealign will be soon reintegrated to** *haarz* The output of *testrealign* also informs on potential trans-splicing events, including strand trans splicing or inter-chromosomal trans splicing. If applied to DNA, this information may indicate structural genomic aberrations.

11.1 Quick start

Currently *testrealign* accepts the segemehl SAM and gzip'ed SAM files. BAM files are not yet supported. With *testrealign.x* you dont need to create an index anymore. **However, it is crucial that you sort the SAM file!**. You can use unix sort (first key: chromosome-field (any order), second key: position (ascending order); watch the header!) or samtools sort for sorting the alignments.

```
> ./testrealign.x -d chr.fa -q mysam.sam -n
```

The tool will generate two files with the extension 'splice.bed' and 'trans.bed'. These two files are in the standard bed format and can readily be uploaded to the UCSC genome browser or other browsers. The file ending with 'splice.bed' holds the regular splice events, the file ending with 'trans.bed' holds all potential trans splice events including all splice events that span more than 200kb or strand trans splicing. The 'splice.bed' file holds entries with the following format:

```
CHR     POS1    POS2    INFO   SCORE    STRAND
```

An example is:

```
chr10   67330517    67331183    splits:365:380:370:N:P  0   +
```

The first field holds the chromsome CHR. The positions of the two splice sites, `POS1` and `POS2`, are shown in field 2 and 3. The information `INFO` in field 4 shows three numbers. The first number indicates the number of reads that support the reported splice site. The second and the third number inform on the total number of splits that can be observed at `POS1` and `POS2`, respectively. The letter 'N' stands for 'normal', indicating that this is a regular splice junction. The letter 'C' stands for 'circular', indicating that this is a potential backsplice junction. The last letter is a quality flag 'P' stands for 'passed' and indicates that there was only one corresponding acceptor-donor pair. The letter 'M' indicates that their are 'multiple' acceptor-donor pairs. The letter 'F' is a warning. It indicates that the algorithm identified sequence similarities between different acceptor or donor sites, i.e. the junction could be wrong. The strand information is only meaningful if a strand specific sequencing protocol was used. The '<basename>.trans.bed' holds distant splice events and strand trans splice events. Three different types of splice events are reported:

tag	explanation
`distsplice`	a distant splice event (>200kB)
`strandsplice`	a local strand trans splice event
`diststrandsplice`	a distant strand trans splice event

Note that for all distant splice events, i.e., `distsplice` and `disttrandsplice`, two BED entries are written to the file. For example:

```
chr9     93673819    93673819    distsplice:chr10:67331183:90:80:80:L:P  0   -
chr10    67331183    67331183    distsplice:chr9:93673819:90:80:80:R:P   0   -
```

The corresponding acceptor or donor site (chromosome and position) is given in the INFO field. The last three numbers again indicate the split reads as described above. This format was chosen to simplify visualization in a genome browser. The letters 'L' and 'R' indicate whether the position is the right or left site of the junction.

12 Complaint department

`steve at bioinf dot uni-leipzig dot de`

13 Under development

The options of haarz not documented here are under heavy development and thus unstable. You may use and evaluate them, but please be advised that in several scenarios you might get a segfault or some other error.