TMAP: the torrent mapping program [1]

Nils Homer

Contents

Preface

This document is meant to serve as a guide for the practical use of TMAP. It includes explanations of all command-line options for each command and binary in TMAP to give an idea of basic usage. Input and output file formats are also detailed. The default options of the various programs are designed intelligently to adapt to the various program uses, but in some cases the options may need to be customized.

This document does not try to explain the underlying algorithms or data-structures used in TMAP. Without proper understanding of the underlying algorithms, it is difficult to use this very flexible program knowledgeably to obtain your desired results. Please see chapter 3 for common questions and their answers.

If you have anything that you would be useful to add to this guide, feel free to relay the addition to the TMAP developers. This includes but is not limited to bugs, typos, and explanations. Please see the http://ioncommunity.iontorrent.comIon Community for more details.

Enjoy!

Chapter 1

Basic Usage

1.1 Program Organization

TMAP consists of a set of utilities, combined into one command line program called `tmap`. Each utility is specified by a unique name or command. Specifying TMAP without a command will give a list of commands, while specifying TMAP with a command but no options will give a list of options for that command.

There are two steps in mapping with TMAP. The first step is to build an index of the reference genome onto which we map. This index needs only to be built once for each genome, performing much of the mapping work upfront. The second step is to map the reads to the reference genome using this index.

TMAP is implemented as a command-line program. It accepts many command-line options to customize and tune the mapping algorithm. The key commands are organized into one binary program called `tmap`. To access each command, we use `tmap <name>`, where `<name>` is the name of the command we wish to execute.

1.2 Officially Supported Versions

The latest TMAP is unsupported. To use a supported version, please see the TMAP version associated with a Torrent Suite release in Table 1.1

1.3 Common Mapping Options

Some common options exist across some or all of the mapping commands (`map1`, `map2`, `map3`, `map4`, `mapvsw` and `mapall`). These options will be discussed here to avoid dupli-

Torrent Suite 2.2	http://github.com/iontorrent/TMAP/tarball/tmap.0.3.7
Torrent Suite 2.0.1	http://github.com/iontorrent/TMAP/tarball/tmap.0.2.3
Torrent Suite 2.0	http://github.com/iontorrent/TMAP/tarball/tmap.0.2.3
Torrent Suite 1.5.1	http://github.com/iontorrent/TMAP/tarball/tmap.0.1.3
Torrent Suite 1.5	http://github.com/iontorrent/TMAP/tarball/tmap.0.1.3
Torrent Suite 1.4.1	http://github.com/iontorrent/TMAP/tarball/tmap.0.0.28
Torrent Suite 1.4	http://github.com/iontorrent/TMAP/tarball/tmap.0.0.25
Torrent Suite 1.3	http://github.com/iontorrent/TMAP/tarball/tmap.0.0.19
Torrent Suite 1.2	http://github.com/iontorrent/TMAP/tarball/tmap.0.0.9
Latest TMAP	http://github.com/iontorrent/TMAP/tags

Table 1.1: TMAP versions associated with Torrent Suite versions

cation.

The TMAP mapping commands can accept their input file from the standard input stream. The output of these commands also write to the standard output stream. This facilitates the use of these TMAP commands in a pipe-and-filter model.

1.3.1 Global Options

`-f,--fn-fasta FILE`

Specifies the file name of the reference genome in FASTA format. The maximum number of bases in the reference genome can be 4294967295, due to the use of 32-bit values to represent each position in the reference.

`-r,--fn-reads FILE`

Specifies the file name of the reads to mapped. The input reads can be in FASTA, FASTQ, or SFF format. If an SFF is input, only reads that have a matching key sequence will be mapped.

`-i,--reads-format STRING`

Specifies the file format of the reads file. Without this option, the format is auto-recognized by the file extension. The valid inputs are `fa` or `fasta` for FASTA, `fq` or `fastq` for FASTQ, or `sff` for SFF. If compiled with SAMTools support, the option also supports `sam` for SAM, or `bam` for BAM.

`-s,--fn-sam FILE`

Specifies the file name of the output file SAM format. If this is not specified, the output will be written to stdout.

`-A,--score-match INT`

Specifies the match score. This number must always be positive.

`-M,--pen-mismatch INT`

Specifies the mismatch penalty. This number must always be positive.

`-O,--pen-gap-open INT`

Specifies the gap open penalty. This number must always be positive.
See section 3.2 for more information how insertions and deletions are scored.

`-E,--pen-gap-extension INT`

Specifies the gap extension penalty. This number must always be positive. See section 3.2 for more information how insertions and deletions are scored.

`-G,--pen-gap-long INT`

Specifies the long gap penalty. This number must always be positive. See section 3.2 for more information how insertions and deletions are scored.

`-K,--gap-long-length INT`

Specifies the number of extra bases to add when searching for long indels. If this number is negative, long indels are not considered. See section 3.2 for more information how insertions and deletions are scored.

`-w,--band,width INT`

Specifies the band width for local alignment. This number must always be positive.

`-g,--softclip-type INT`

Specifies that the type of soft-clipping to perform.

0. soft-clip both the left and right portions of the read.
1. soft-clip only the left portion of the read.
2. soft-clip only the right portion of the read.
3. do not soft-clip any portion of the read

See section 3.5 to learn how bases are soft clipped.

`-W,--duplicate-window INT`

Specifies to remove duplicate mappings that occur within this bp window.

`-B,--max-seed-band INT`

Specifies the window in bases in which to group seeds prior to Smith Waterman.

`-U,--unroll-banding`

Specifies to unroll the grouped seeds from banding if multiple alignments are found. With this option, if multiple best scoring alignments are found within a group of seeds (see `-B`), the seeds are ungrouped and Smith Waterman is performed on each seed.

`-T,--score-thres INT`

Specifies the number of multiples of the match score (`-A`) for the minimum scoring threshold.

`-q,--reads-queue-size INT`

Specifies the number of reads to cache or load into memory at one time.

`-n,--num-threads INT`

Specifies the number of threads to run. The default value will be auto-detected on the system.

`-a,--aln-output-mode INT`

Specifies the output filter for the mappings.

0. returns the mapping with the best score only if all other mappings had worse score, otherwise the read is unmapped.
1. returns the mapping with the best score. If more than one mapping has this score, a random mapping with this score is returned.
2. returns all the mappings with the best score.
3. returns all the mappings, regardless of score.

Reads that have no mapping are returned as unmapped reads.

`-R,--sam-read-group STRING`

Specifies the RG (read group) line to use in the SAM file (with tab separators). The "PG", "FO", and "KS" tags should not be specified in this string; they will be added by tmap. Alternatively, multiple `-R` options can also be used to populate the RG line. For example, `-R CN:SEQUENCING_CENTER -R PG:TMAP -R PL:IONTORRENT` will populate the `CN`, `PG`, and `PL` tags in the RG record.

`-D,--bidirectional`

Specifies the input reads are to be annotated as bidirectional using the XB SAM record optional tag.

`-I,--use-seq-equal`

Specifies to use the '=' symbol in the SEQ field.

`-u,--rand-read-name`

Specifies to randomize based on the read name.

`-C,--ignore-rg-from-sam`

Specifies to do not use the RG header and RG record tags in the SAM file (if the input file is a SAM file, and SAM input files are enabled).

`-j,--input-bz2, -z,--input-gz`

Specifies that the input is bzip2 (`-j`) or gzip (`-z`) compressed. This is auto-recognized if the input file name has the extension `.bz2` for bzip2 and `.gz` for gzip.

`-o,--output-type`

Specifies the output type:

0. SAM.
1. BAM (compressed). This is equivalent of using `-o 0` and piping the output into `samtools view -Sb -`.
2. BAM (uncompressed). This is equivalent of using `-o 0` and piping the output into `samtools view -Su -`.

`-k,--shared-memory-key INT`

Specifies the shared memory key if the reference index has been loaded into shared memory.

`-H,--vsw-type INT`

Specifies the vectorized Smith Waterman algorithm to use. There are many existing implementations of Smith Waterman, which is a crucial step in TMAP used to determine the quality of each candidate alignment. TMAP includes a number of implementations, including those obtained from a Top Coder contest. These implementations vary on speed and correctness, and so your mileage may vary. Please proceed with caution. The algorithms included are:

1. lh3/ksw.c/nh13
2. simple VSW
3. SHRiMP2 VSW [not working]
4. Psyho (Top Coder #1)
5. ACRush (Top Coder #2)
6. folsena (Top Coder #3)
7. logicmachine (Top Coder #4)

8. venco (Top Coder #5) [not working]
9. Bladze (Top Coder #6)
10. ngthuydiem (Top Coder #7) [Farrar cut-and-paste]

NB: currently only #1, #4, and #6 have been tested.

`-v,--verbose`

Specifies to print verbose progress messages, otherwise progress messages will be surpressed.

`-h,--help`

Specifies to print a help message, listing all available options.

`--min-seq-length INT`

Specifies the minimum sequence length to consider (inclusive). This is applied to each algorithm independently. Therefore, when using `mapall`, we could specify a unique sequence length range (using `---min-seq-length` and `---max-seq-length`) each algorithm.

`--max-seq-length INT`

Specifies the maximum sequence length to consider (inclusive). This is applied to each algorithm independently. Therefore, when using `mapall`, we could specify a unique sequence length range (using `---min-seq-length` and `---max-seq-length`) each algorithm.

1.3.2 Flowspace Options

These options control how TMAP uses and outputs information about the flow order from the sequencing machine. Flow space re-alignment will be performed when the flow order is specified (`-F,--flow-order`). See section 3.6 for a description of flow space re-alignment.

`-X,--pen-flow-error INT`

Specifies the flow score penalty. This number must always be positive.

`-y,--softclip-key`

Specifies to soft clip only the last base of the key sequence.

`-Y,--sam-flowspace-tags`

Specifies that flowspace specific tags should be added to the output SAM file when available. See section 2.1 for more information.

`-N,--ignore-flowgram`

Specifies to not use the flowgram, otherwise uses the flowgram for flowspace re-alignment when available.

`-G,--remove-sff-clipping`

Specifies to not remove bases and qualities based on the adapter and quality clipping fields found in the an input SFF file.

`-F,--final-flowspace`

Specifies to produce the final alignment in flow space.

1.3.3 Pairing Options

These options control how TMAP uses and outputs information about the paired end or mate pair reads. Pairing will be performed when exactly two input read files are given using the `-r` option twice. The insert size distribution (mean and standard deviation) are computed from the first chunk of reads (`-q`). Currently, the strandedness (`-S`) and positioning (`-P`) must be given. Read rescue significantly improves specificity at the cost of slower run times.

`-Q,--pairing`

Specifies the pairing orientation. This will override `-S` and `-P`.

0. no pairing is to be performed
1. 1 - mate pairs (`-S 0 -P 1`)
2. 2 - paired end (`-S 1 -P 0`)

`-S,--strandedness`

Specifies the strand orientation between the pairs.

0. the reads should be mapped to the same strand
1. the reads should be mapped to opposite strands

`-P,--positioning`

Specifies that position orientation between the pairs.

0. the first read of the pair (A) is before the second read of the pair (B).
1. the second read of the pair (B) is before the first read of the pair (A).

`-b,--ins-size-mean FLOAT`

Specifies the expected mean insert size.

`-c,--ins-size-std FLOAT`

Specifies the expected insert size standard deviations.

`-d,--ins-size-std-max-num FLOAT`

Specifies the maximum number of standard deviations for a pair to be proper. See section 2.1 for more informaton about proper pairs.

`-p,--ins-size-outlier-bound FLOAT`

Specifies the insert size 25/75 quartile outlier bound.

`-t,--ins-size-min-mapq INT`

Specifies the minimum mapping quality to consider for computing the insert size.

`-L,--read-rescue`

Specifies to perform read rescuing during pairing.

`-l,--read-rescue-std-num FLOAT`

Specifies the number of insert size standard deviations to search around the inferred rescue position.

`-m,--read-rescue-mapq-thr INT`

Specifies the mapping quality threshold for read rescue.

1.4 tmap index

The **index** command creates a compact version of the reference genome and associated index. The index is stored as a compressed suffix array using the FM-index and BWT transform (Ferragina and Manzini (2000); Burrows and Wheeler (1994)). A hash into this index accelerates this lookups of DNA sequences in this index. In fact, a second additional index of the reference genome is created that indexes the reverse (but not complimented) reference genome. This second index further speeds up the search time.

See section 1.3 for common options that are in use in this command.

1.4.1 Usage

`-o INT`

Specifies the occurence interval size o, storing only every oth occurrence interval. This be a power of two greater than or equal to 32.

`-w INT`

Specifies the k-mer size (the number of bases) to hash. The size of the hash give the k-mer size k in bytes is:

$$= 2\sum_{n=1}^{k} 4^n = 2\left(\frac{1-4^{k+1}}{1-4} - 1\right)$$

using the Taylor series.

`-i INT`

Specifies the suffix array interval size i, storing only every ith suffix array interval. This must be one, or a powerof two.

`-a STRING`

Specifies the BWT construction algorithm (`bwtsw` or `is`). The `bwtsw` algorithm is for genomes larger than or equal to 10Mb, and the `is` algorithm is for genomes smaller than 10Mb. This will be auto-recognized during index creation if this option is omitted.

`-H`

Specifies to not validate the BWT hash.

`--version`

Specifies to print the index format that will be created byt TMAP and exit. Format strings are of the form `tmap-f<n>`, where `<n>` will only increase as new formats are created. Format strings will be the same for all compatible indices.

1.4.2 IUPAC Ambiguity Codes

Please see section 3.1 for more details.

1.5 tmap map1

The **map1** is a command to quickly map short sequences to a reference genome by intelligently enumerating errors. This algorithm is not well suited for longer reads (< 150bp) and based off of the BWA short-read algorithm (Li and Durbin (2009)).

See section 1.3 for common options that are in use in this command.

1.5.1 Usage

`--seed-length INT`

Specifies the primary seed (k-mer) length; reads must be of at least this length.

`--seed-max-diff INT`

Specifies the maximum number of edits allowed in the primary seed. This includes both mismatches and each base in an insertion or deletion.

`--seed2-length INT`

Specifies the secondary seed (k-mer) length; this number of bases will be examimed before extending all mappings.

`--max-diff NUM`

Specified the maximum number of edits or false-negative probability assuming the maximum error rate (`--max-error-rate`). This former includes both mismatches and each base in an insertion or deletion. The latter false-negative probability is the probability of not consideing the correct mapping if it exists.

`--max-error-rate NUM`

Specified the assumed maximum per-base error rate. This classifies both mismatches and each base in an insertion or deletion as errors

`--max-mismatches NUM`

Specifies the maximum number of mismatches allowed, or the fraction of mismatches with respect to the read length. When the secondary seed is enabled, this parameter is relative to the secondary seed length, not the read length;

`--max-gap-opens NUM`

Specifies the maximum number of indels allowed in the mapping, or the fraction of indels with respect to the read length. The number of indels is equal to the number of gap opens. When the secondary seed is enabled, this parameter is relative to the secondary seed length, not the read length;

`--max-gap-extensions NUM`

Specifies the maximum number of indel extensions allowed in the mapping, or the fraction of indel extensionswith respect to the read length. The number of indel extensions is equal to the number of gap extensions. When the secondary seed is enabled, this parameter is relative to the secondary seed length, not the read length;

`--max-cals-deletion INT`

Specifies the maximum number of candidate alignment locations (CALs) to allow a deletion to be extended.

`--indel-ends-bound INT`

Specifies to disallow indels within this number of bases from the ends of the read.

`--max-best-cals INT`

Specifies to stop searching for mappings when this many best scoring mappings have been found.

`--max-nodes INT`

Specifies the maximum number of alignment nodes in memory in the implicite prefix trie traversal before the search is stopped.

1.6 tmap map2

The **map2** is a command to quickly map long sequences to a reference genome. This algorithm is well suited for longer reads ($\geq$ 150bp) and based off of the BWA long-read algorithm (Li and Durbin (2010)).

See section 1.3 for common options that are in use in this command.

1.6.1 Usage

`--max-seed-hits INT`

Specifies the maximum number of CALs allowed to be returned by a seed before it is ignored.

`--length-coef FLOAT`

Specifies the coefficient to adjust the mapping score threshold based on the read length. Given a l-long read, the threshold for a mapping to be retained is $a * max\{T, c * log(l)\}$, where a is the match score (`-A`) and T is the minimum score threshold (`-T`).

`--max-seed-intv INT`

Specifies the maximum seed interval size (mappings) to retain at during extension of the seed.

`--z-best INT`

Specifies the maximum number of top-scoring nodes to keep on each iteration.

`--seeds-rev INT`

Specifies the maximum seeding interval size for extending a mapping.

`--narrow-rmdup`

Specifies to remove duplicate seeds for narrow suffix array.

1.7 tmap map3

The **map3** is a command to map sequences to a reference genome. This algorithm is well suited for longer reads ($\geq$ 150bp) and a simplification of the SSAHA long-read algorithm (Ning *et al.* (2001)).

See section 1.3 for common options that are in use in this command.

1.7.1 Usage

`--hit-frac FLOAT`

Specifies the fraction of seed positions that are under the maximum (`--max-seed-hits`).

`--seed-step INT`

Specifies the number of bases to increase the seed while repetitive (-1 to disable).

`--seed-length INT`

Specifies the k-mer length to seed candidate alignment locations (CALs). With a value of -1 will set the seed length to $(ceiling(log_4(R)) + 2)$ where R is the reference genome length.

`--max-seed-hits INT`

Specifies the maximum number of CALs allowed to be returned by a seed before it is ignored.

`--hp-diff INT`

Specifies the number of bases to enumerate for a single homopolymer within the seed.

`--fwd-search`

Specifies to perform a forward search instead of a reverse search.

`--skip-seed-frac FLOAT`

Specifies the fraction of a seed to skip when a lookup succeeds.

1.8 tmap map4

The **map4** is a command to map sequences to a reference genome. This algorithm is well suited for longer reads ($\geq$ 100bp) and is a variation of the super-maximal exact matching (SMEM) algorithm proposed by Heng Li (Li (2012)). SMEMs are equivalent to returning all longest common substrings between the read and reference.

See section 1.3 for common options that are in use in this command.

1.8.1 Usage

`--hit-frac FLOAT`

Specifies the fraction of seed positions that are under the maximum (`--max-seed-hits`).

`--seed-step INT`

Specifies the number of bases to increase the seed for each seed increase iteration (-1 to disable). The algorithm tries to find SMEMs starting at specific offsets within the read. Beginning at the start of the read, this option controls how many bases to skip to obtain the next offset

`--min-seed-length INT`

Specifies the minimum number of bases to use to seed (smallest SMEM). With a value of -1 will set the seed length to $(ceiling(log_4(R)) + 2)$ where R is the reference genome length.

`--max-seed-length INT`

Specifies the maximum number of bases to use to seed (longest SMEM).

`--max-seed-length-adj-coef FLOAT`

Specifies the maximum seed length adjustment coefficient. Suppose the maximum seed length is L, the query length is N, and the co-efficient is Z. If $N < L * Z$ then the maximum seed length will be adjusted to $(N + 1)/Z$.

`--max-iwidth INT`

Specifies the maximum interval size to accept a hit.

`--max-repr INT`

Specified the maximum representitive hits for repetitive hits. This is useful in repetitive genomes, where the returned SMEM masks the correct seed, which is contained in smaller SMEM. This will keep a number of repetitive SMEM hits.

`--rand-repr`

Specifies to choose the representitive hits randomly, otherwise uniformly. See `--max-repr` for details on repetitive htis.

`--use-min`

Specifies when seed stepping, try seeding when at least the minimum seed length is present, otherwise maximum.

1.9 tmap mapvsw

The **mapvsw** is a command to map sequences to a reference genome. This algorithm performs the full Smith Waterman algorithm alignment of the read to the reference genome, using SSE2 (vectorized) programming instructions. This algorithm should not be used for large number of reads or large genomes, and instead should be used for debugging and investigating small numbers of reads..

See section 1.3 for common options that are in use in this command.

1.9.1 Usage

There are no algorithm specific options.

1.10 tmap mapall

The **mapall** is a command to quickly map short sequences to a reference genome. This command combines available mapping algorithms for fast and sensitive alignment. The algorithms follows a multi-stage approach, with a set of algorithms and associated settings for each stage. If there are no mappings for a read by applying the algorithms in the ith stage that pass filters, then the algorithms in the $i+1$th stage are applied. For example, a set of algorithms to quickly align near-perfect reads may be used in the first stage, while a set of sensitive algorithms may be used to map difficult reads in the second stage

It is recommended that `mapvsw` is not used for any large scale mapping project. Please see section 1.9 for more details.

See section 1.3 for common options that are in use in this command.

1.10.1 Overview

First, a set of global options is specified that will be used for the algorithms to be applied at all stages. Global options should not be given in the options for a specific stage or algorithm. Next, a stage will be specified and its associate options that will be applied across algorithms in that stage. Stage-specific options should not be given in the options for any algorithm in that stage. The stage name should be specified by "stage%d", where "%d" is the stage number (one-based). Finally, the algorithms and their associated options will be specified for the given stage.

An example would be:

```
tmap mapall -f ref.fasta -r reads.fastq -g 1 -M 3 stage1
--stage-keep-all map1 --seed-length 12 --seed-max-diff 4
stage2 map2 --z-best 5 map3 --max-seed-hits 10
```

In this case, the global options `-f ref.fasta -r reads.fastq -g 1 -M 3` will be applied to all stages and algorithms. Therefore, `map1` algorithm will be applied with the options `-g 1 -M 3 --seed-length 12 --seed-max-diff 4` in the first stage. If no mapping is found for a read, the `map2` algorithm with the options `-g 1 -M 3 --z-best 5` and the `map3` algorithm with the options `-g 1 -M 3 --max-seed-hits 10` will be applied in the second stage. Notice how the global options `-g 1 -M 3` are applied to all the algorithms where applicable. It is possible to have the same algorithm run in two different stages.

The recommended values are pre-set, and the following command line is recommended:

```
tmap mapall -f <in.fasta> -r <in.fastq> -n <num.threads> -v
map1 map2 map3 > <out.sam>
```

1.10.2 Usage

The following options apply to a single stage, and control how mappings are handled between stages.

`--stage-score-thres INT`

Specifies the number of multiples of the match score (`-A`) for the minimum scoring threshold for the current stage. An alignment is filtered in the given stage if it has an alignment score less than this threshold, and there are more stages to process.

`--stage-mapq-thres INT`

Specifies the mapping quality threshold for the current stage. An alignment is filtered in the current stage if it has a mapping quality less than this threshold, and there are more stages to process.

`--stage-keep-all`

Specifies to keep mappings from the current stage for the next stage. If this option is given, the mappings from the previous stage are added to any mappings from the subsequent stage as candidates.

`--stage-seed-freq-cutoff FLOAT`

Specifies the minimum frequency a seed must occur in order to be considered for mapping.

`--stage-seed-freq-cutoff-group-frac FLOAT`

Specifies if more than this fraction of groups were filtered, keep representative hits.

`--stage-seed-freq-cutoff-rand-repr INT`

Specifies the number of representative hits to keep. In addition, the group with most seeds will always be kept if this option has a greater than zero value.

`--stage-seed-freq-cutoff-min-groups INT`

Specifies the minimum of groups required after the filter has been applied, otherwise iteratively reduce the filter.

`--stage-seed-max-length INT`

Specifies the length of the prefix of the read to consider during seeding.

1.11 tmap server

The **server** command loads the reference genome index and data into shared memory. This lets other mapping instances avoid having to load the index upon each execution. This program will try to fail gracefully, detaching shared memory upon exiting.

See section 1.3 for common options that are in use in this command.

1.11.1 Usage

`-c STRING`

Specifies the command for the server (`start`, `stop`, `kill`). The `start` command loads data into shared memory, and waits for a `ctrl-c` or `SIGINT` signal. The `stop` command will signal a currently running server to stop and detach shared memory. The `kill` command will forcible detach the shared memory segment and signal a currently running server to exit. The latter command is especially useful for killing zombied processes and detaching lost shared memory segments.

`-k INT`

Specifies the shared memory key for this server. The shared memory segment will be identified by this key.

`-a`

Specifies to load all reference genome data structures into memory.

`-r`

Specifies to load the forward packed reference sequence.

`-R`

Specifies to load the reverse packed reference sequence.

`-b`

Specifies to load the forward BWT sequence.

`-B`

Specifies to load the reverse BWT sequence.

`-s`

Specifies to load the forward suffix array.

`-S`

Specifies to load the reverse suffix array.

Chapter 2

File Formats

2.1 SAM Alignment Format

TMAP is produces mappings in the SAM format (Li *et al.* (2009)). Optional tags are used to store information about mappings useful for downstream processing.

2.1.1 SAM Header Fields

The HD, RG, SQ, and PG SAM header fields will be outputted. Specific details can be found below.

RG

The RG field can be populated using the `-R` option. By default the `ID`, `SM`, and `PG` tags are included. When specifying an RG line, do not include the PG tag as it will be populated by tmap: If the input is a file with flowspace information (SFF or SAM/BAM), then the read group will include the FO and KS tags. The FO and KS tags indicate the flow order and the key sequence.

PG

The PG field will include the ID, VN, and CL tags. The ID tag should always be `tmap`.

2.1.2 SAM Record Optional Tags

AS

This tag stores the alignment score. All IUPAC code positions are treated as mismatches. If the alignment contains an N in the reference but an A in the read, this score will be incorrect (overestimated) as the N was converted to an A in the reference. Please see section 3.1 for more details.

FZ

This tag stores the flow signals when an SFF is inputted and the `-Y` option is used. The flowgram values are stored as a string of hexidecimal values, with each group of four hexidecimal values corresponding to one flow signal.

MD

This tag stores the MD array. The goal of this array is to follow the conventions in SAMTools (Li *et al.* (2009)). Any IUPAC code originally in the reference will be present in this field. Please see section 3.1 for more details.

NM

This tag stores the edit distance from the reference sequence. Matches to any non-DNA IUPAC ambiguity code (B, D, H, K, M, N, R, S, V, W, Y) will be counted as a mismatch. Please see section 3.1 for more details.

PG

This tag stores the program group identifier corresponding to the PG SAM header field.

RG

This tag stores the read group identifier corresponding to the RG SAM header field.

NH

This tag stores the number of hits found during alignment. The number of alignments reported in the output is determined by `-a` and therefore fewer alignments may be present in the output than specified by the `NH` tag.

XA

This tag stores the algorithm that produced this mapping and from what stage. The format is the algorithm name, and then the zero-based stage, separated by a dash.

XB

This tag indicates that the given read originates from a bidirectional read.

XE

This tag stores the number of seeds supporting this mapping specifically for `map2`.

XF

This tag stores from where the mappings originated: one indicates from the forward search, two indicates from the reverse search, and three indicates from both the forward and reverse search.

XG

This tag stores the number of indel extensions (gap extensions) in the mapping, or in the secondary seed if used.

XI

This tag stores the size of the suffix interval for this mapping.

XM

This tag stores the number of mismatches in the mapping, or in the secondary seed if used.

XO

This tag stores the number of indels (gap opens) in the mapping, or in the secondary seed if used.

XS

This tag stores the alignment score of next-best sub-optimal mapping.

XT

This tag stores the number of seeds supporting this mapping.

XZ

This tag stores the original alignment score, if an SFF was inputted.

ZF

This tag stores the zero-based index of the first flow signal corresponding to the template, inferred from the clipping values in the SFF. This this is outputted only when an SFF is inputted and the `-Y` option is used.

Chapter 3

Commonly Asked Questions

Contents

3.1 How are IUPAC ambiguity codes handled in the reference/target FASTA?

Ambiguous IUPAC codes in the reference/target FASTA will be converted to the lexicographically smallest DNA base that is not compatible to the IUPAC code to ensure minimum reference bias. For example, an IUPAC base `R`, which represents an `A` or a `G`, will be converted to a C. All Ns in the reference will be converted to As. Furthermore, any non-IUPAC character will be treated as an N. The ambiguity codes will only be re-considered when calculating the NM and MD SAM record optional tags.

3.2 How are insertions and deletions scored?

Given gap open and gap extension penalties O and E, a contiguous indel of length L will have a score of $O + (E * L)$.

3.3 How are insertions and deletions justified?

TMAP will attempt to justify insertions and deletions (indels) to the 5' (left-most) end of the genomic (reference) forward strand. This allows the indel to be consistently placed with respect the reference, regardless of the strand during sequencing. Neither the query sequence nor the reference sequence are changed, only the indel position in the alignment.

For example, if we have the alignment (query on top, target on the bottom):

```
CGAACATTTTTCGTGGA
||||||||||-||||||
CGAACATTTT-CGTGGA
```

the insertion will be moved to its left-most equivalent position:

```
CGAACATTTTTCGTGGA
||||||-||||||||||
CGAACA-TTTTCGTGGA
```

Another example would be in a non-homopolymer deletion:

```
CGAACAT----CGTGGA
|||||||----||||||
CGAACATACATCGTGGA
```

Justifying this deletion to the left would give:

```
CGA----CAATCGTGGA
|||----||||||||||
CGAACATACATCGTGGA
```

Justification will not occur if it disconnects a contiguous indel, causing a potential difference in the alignment score. The following alignment:

```
AGGTGC--TTTACAAGC
||||||--|||||||||
AGGTGCCTTTTACAAGC
```

could be changed to:

```
AGGTG-C-TTTACAAGC
|||||-|-|||||||||
AGGTGCCTTTTACAAGC
```

This would change the alignment and decrease) the resulting score.

3.4 Will the output of TMAP be the same if run twice?

TMAP will always produce the same output if run twice, even when using multiple threads (`-n`). This is enabled by using a complementary-multiply-with-carry random number generator, one per thread. Of course, if different options are used, then the results will most likely vary, even if a different number of threads are used.

3.5 How does TMAP soft clip bases in the read?

TMAP has the ability to soft clip bases during alignment, that is to not include in the alignment bases that are present in the read. This may be useful to only map high quality portions of the read, for example if the 3' end of the read has many low quality bases called but their inclusion in the alignment results in many mismatches or insertions/deletions (indels).

The user may specify the desired type of soft-clipping using the `-g` option, with any combination of 5' and 3' soft-clipping (or neither) allowed. The clipped bases are determined from local alignment (glocal or semi-global alignment). For example, if 3' end soft-clipping is enabled, then a prefix of the read is aligned to a sub-sequence of the reference. Where

the read prefix ends in the alignment all comes from the local alignment, which is heavily influenced by the scoring parameters.

For example, consider the alignment below, which aligns the full read (GATTACA) to (a subsequence of) the reference (AGATTAAC). Suppose we have scoring parameters (1,-3,-7,-2) for the match, mismatch, indel start, and indel extend operations. The alignment score would be 1+1+1+1+1-7+1 = -1.

```
READ: GATTACA
      |||||+|
REF:  GATTA-A
```

If instead we allow soft-clipping at the end of the read, we need only align a prefix of the read to (a subsequence of) the reference. The optimal alignment score would be 5, with the alignment shown below.

```
READ: GATTACA
      |||||SS
REF:  GATTA
```

The details of the local alignment algorithms and their variations are well characterized in the Literature or in an Introduction to Bioinformatics text book (Jones and Pevzner (2004)). Using the local alignment and the scoring parameters to determine the soft-clipping is a powerful and flexible solution to trade off reduced alignment length for higher quality alignments via soft-clipping.

For how soft-clipping is represented in SAM files, see the SAM specification for more details.

3.6 What is flow space re-alignment

Re-alignment in flow space is possible with TMAP, and is turned on only when the flow order option is set (`-F`). With normal Smith Waterman, errors in homopolymers are treated similarly to insertions and deletions (indels) that are from biological origins. Re-alignment in flow space allows TMAP to consider indels and errors in homopolymers differently than other insdels, even though the resultant base space alignment may be similar.

For example, the following alignment (read on top) shows a one-base insertion of a G. This could be due to an error estimating the number of Gs from a specific flow, or a true biological difference/insertion.

```
TAGGACACCGGTCTGA
|||||||||-||||||
TAGGACACC-GTCTGA
```

In the following example, we observe a mismatch/SNP difference in base space:

```
TAGGACACCGGTCTGA
||||||||| ||||||
TAGGACACCCGTCTGA
```

This could be due to a biological difference/SNP, or due to an under-call of the C and an overcall of the adjacent G. In the latter case, an acceptable alignment may be:

```
TAGGACACC-GGTCTGA
|||||||||--||||||
TAGGACACCC-GTCTGA
```

Finally, more complicated scenarios can arise, where flow space re-alignment may yield a more faithful alignment. In the following example, a base space alignment may be:

```
TCAT--AAATTTT
|||  --|||||||
TCAATAAAATTTT
```

A suitable flow space re-alignment would be:

```
TCA-T-AAATTTT
|||-|-||||||
TCAATAAAATTTT
```

The first alignment may be preferred in base space due to the use of affine gap penalties (see section 3.2) and the other scoring parameters. If we penalize insertions or deletions due to homopolymer differences less than other insertions or deletions, the latter alignment may be more likely. Given Ion Torrent data, the latter may be more faithful to the underlying error mode.

Another feature is the `-G` and `-K` options. When indels are detected using this method, the score for the indel is given by `-G`. Since this is applied after the above alignment, typically only long gaps are given this score.

3.7 How are multiple equally scoring alignments in the same region chosen?

In some cases, there exist two alignments of a read in the same target region with equal score. Although these alignments have the same score, they may contain a different number of query bases. For example, the following two alignments can be produced when the end of the query is allowed to be clipped.

Example 1:

```
GGGCGGTAGCAGCCTGCCTGGGAATGGGCCACCCCGAATCCCGACCCGAAATAAATGCAC
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
GGGCGGTAGCAGCCTGCCTGGGAATGGGCCACCCCGAATCCCGACCCGAAATAAATGCAC

CAGCATGCTGACGCGGTAAAACAGGAGCCGCCGGCAATACGCCAGTTTCACCCTCATCCA
|||||||||||||||||||||||||||||||||||| --|||||||||||||||||
CAGCATGCTGACGCGGTAAAACAGGAGCCGCCGGCAATAG--CAGTTTCACCCTCATCCC

CCTGAGGTCCTATGTTATCACCTACTGCCATAGAG
 ||||||SSSSSSSSSSSSSSSSSSSSSSSSSS
ACTGAGG--------------------------
```

Example 2:

```
GGGCGGTAGCAGCCTGCCTGGGAATGGGCCACCCCGAATCCCGACCCGAAATAAATGCAC
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
GGGCGGTAGCAGCCTGCCTGGGAATGGGCCACCCCGAATCCCGACCCGAAATAAATGCAC

CAGCATGCTGACGCGGTAAAACAGGAGCCGCCGGCAATACGCCAGTTTCACCCTCATCCA
|||||||||||||||||||||||||||||||||||| --|||||||||||||||||S
CAGCATGCTGACGCGGTAAAACAGGAGCCGCCGGCAATAG--CAGTTTCACCCTCATCC-

CCTGAGGTCCTATGTTATCACCTACTGCCATAGAG
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
----------------------------------
```

If the penalty for having two mismatches is equal to the score of having six matches, then the two alignments will be equivalently scored.

TMAP will guarantee to find the alignment with the contains the most number of query bases (the first one above) unless the user specifies that soft-clipping is allowed on both ends of the read. In the latter case, no guarantee can be made.

3.8 How do I view the default parameters?

The default parameters are shown using the `-h` option. The displayed values are updated for any command line parameters that are given before the `-h` option.

To see the default values for the mapall command:

```
tmap mapall -h
```

To see the default values for the mapall command and stage-specific options:

```
tmap mapall stage1 -h
```

To see the default values for the mapall command, stage-specific options, and a subset of algorithms:

```
tmap mapall stage1 map1 map2 map3 -h
```

3.9 How does the TMAP algorithm work on a high level?

There are two steps in mapping with TMAP. The first step is to build an index of the reference genome onto which we map. This index needs only to be built once for each genome, performing much of the mapping work upfront. The second step is to map the reads to the reference genome using this index.

The second step is performed for each run or experiment and there are two sub-steps for each read. The first sub-step tries to quickly find a short/small list of candidate mapping locations, effectively reducing our search space from billions of bases to thousands of bases. This is called the seeding step. The next step examines these locations with a more sensitive algorithm, Smith Waterman, to prioritize these candidates.

3.10 Is mapping done in base space or flow space?

The mapping is done in base space. Due to the high quality of the input data, mapping considering the flow information yields little advantage. TMAP does however carry forward any flow information, outputting it to the SAM file. The flow information is critical to the downstream variant calling.

3.11 How many errors can a read have and still be mapped?

There is no hard limit to the number of errors a read can have for it to be mapped. The practical limit is bounded by a number of factors including but not limited to:

1. The length of the read (longer reads have a better chance of mapping).
2. The quality across the read (reads with the first 100 bp of high quality will more likely be mapped than those with a uniform error rate).
3. The sequence complexity of the read relative to the reference (repetitive elements confound mapping).
4. The reference genome size (the larger the reference, the more evidence is necessary for confident mapping).
5. The error mode of the read (mismatches versus indel error modes).
6. The algorithms and parameters chosen (algorithm choice and parameter choice can be tuned to run slower but have higher sensitivity).

3.12 How long must a read be to map?

There is no lower or upper limit to how long a read must be to map. However, depending on the genome size, we need more evidence for a read to map. Therefore, for larger genomes, for example human, reads must be longer or of higher quality. This is intimately related to section 3.11.

3.13 Are mismatches favored over indels when making an alignment?

The favoring of indels versus mismatches is determined by the alignment scoring parameters specified in TMAP. These include the `-A`/`-M`/`-O`/`-E` options. Using these parameters, we favor indels over mismatches, or vice versa. By default, mismatches are favored over indels as SNPs (biological mismatches) are more likely in human (a main application) than indels.

3.14 How do I reconcile choices made by TMAP regarding whether to create an indel or mismatch when doing variant calling?

The final step of TMAP is producing the alignment with Smith Waterman. This algorithm is may find multiple optimal alignments in the same small window, but only one is reported by default. For example, justifying indels does not change the alignment score but the alignment. Furthermore, TMAP does not use flow space information during mapping, but outputs it to the SAM file. This information is more informative when combining multiple observations (reads) within variant calling. Therefore, performing variant calling in a flow space aware manner is critical to the accurate detection of variants. Due to the ambiguity of selecting between multiple optimal alignments, it is difficult to select consistent alignments across reads without examining them all in aggregate, which is effectively variant calling.

Chapter 4

Appendix

Bibliography

Burrows, M. and Wheeler, D. (1994). A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation.

Ferragina, P. and Manzini, G. (2000). Opportunistic data structures with applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 390–, Washington, DC, USA. IEEE Computer Society.

Jones, N. and Pevzner, P. (2004). *An Introduction to Bioinformatics Algorithms*. MIT Press, Cambridge.

Li, H. (2012). Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. *Bioinformatics*.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li, H. and Durbin, R. (2010). Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, **26**, 589–595.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Ning, Z., Cox, A., and Mullikin, J. (2001). SSAHA: a fast search method for large DNA databases. *Genome Res.*, **11**, 1725–1729.