TMAP: the flow sequence 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 `http://ioncommunity.iontorrent.com` 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 Common Mapping Options

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

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.

1.2.1 Global Options

`-f 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 FILE`

Specifies the file name of the reads to mapped. The input reads can be in FASTA, FASTQ, or SFF format.

`-F 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.

`-A INT`

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

`-M INT`

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

`-O 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 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.

`-X INT`

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

`-w INT`

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

`-g 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

`-W INT`

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

`-B INT`

Specifies the window in bases in which to group seeds.

`-q INT`

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

`-n INT`

Specifies the number of threads to run.

`-a 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 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.

`-Y`

Specifies that SFF specific tags should be added to the output SAM file.

`-G`

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

`-j, -z`

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.

`-J, -Z`

Specifies that the output should be bzip2 (`-J`) or gzip (`-Z`) compressed.

`-k INT`

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

`-v`

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

`-h`

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

`-u 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 `-u` and `-U`) each algorithm.

`-U 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 `-u` and `-U`) each algorithm.

1.3 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 (**??**). 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 1.2 for common options that are in use in this command.

1.3.1 Usage

`-o INT`

Specifies the occurence interval size o, storing only every oth occurrence interval. This must be one, or a multiple of 16.

`-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 multiple of 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.

`--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.

1.3.2 IUPAC Ambiguity Codes

Please see section 3.1 for more details.

1.4 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 (**?**).

See section 1.2 for common options that are in use in this command.

1.4.1 Usage

`-l INT`

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

`-s INT`

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

`-L INT`

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

`-p NUM`

Specified the maximum number of edits or false-negative probability assuming the maximum error rate (`-P`). 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.

`-P NUM`

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

`-m 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;

`-o 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;

`-e 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;

`-d INT`

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

`-i INT`

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

`-b INT`

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

`-Q INT`

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

1.5 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 (**?**).

See section 1.2 for common options that are in use in this command.

1.5.1 Usage

`-c 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`).

`-w INT`

Specifies the band width in the banded local alignment.

`-T INT`

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

`-S INT`

Specifies the maximum seeding interval size for extending a mapping.

`-b INT`

Specifies the number of mappings to retain at during extension of the seed.

`-N INT`

Specifies the number seeds supporting the given mapping needed to skip the reverse alignment.

1.6 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 (**?**).

See section 1.2 for common options that are in use in this command.

1.6.1 Usage

`-l 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.

`-S INT`

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

`-w INT`

Specifies the band width in the banded local alignment.

`-T INT`

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

`-H INT`

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

`-V FLOAT`

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

1.7 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 1.2 for common options that are in use in this command.

1.7.1 Usage

There are no algorithm specific options.

1.8 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 two-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 first stage that pass filters, then the algorithms in the second 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 1.7 for more details.

See section 1.2 for common options that are in use in this command.

1.8.1 Overview

First, a set of global options is specified that will be used for the algorithms to be applied at both stages. Global options should not be given in the options for a specific algorithm. Next, the algorithms and their associated options will be specified for each stage. An algorithm to be applied in the first stage should be specified using its name in lowercase. The options specific to this algorithm should be specified directly thereafter. An algorithm to be applied in the second stage should be specified using its name in uppercase. The options specific to

this algorithm should be specified directly thereafter. The order of the specified algorithms does not matter, only the case (upper versus lower) in determining which algorithm is applied at which stage.

An example would be:
`tmap mapall -f ref.fasta -r reads.fastq -g -M 3 map1 -l 12 -s 4 MAP2 -b 5 MAP3 -S 10.`
In this case, the `map1` algorithm will be applied with the options `-l 12 -s 4 -M 3` in the first stage. If no mapping is found for a read, the `map2` algorithm with the options `-b 5 -g -M 3` and the `map3` algorithm with the options `-g -M 3 -S 10` will be applied in the second stage. Notice how the global options `-g -M 3` are applied to all the algorithms where applicable. It is possible to have a given algorithm run in both stages with different options.

1.8.2 Usage

`-I`

Specifies to apply the output filter (`-a`) and duplicate removal (`-W`) separately for each algorithm. This is useful for comparing the mappings across all the algorithms.

`-C INT`

Specifies the number of multiples of the match score (`-A`) for the minimum scoring threshold for the first stage. An alignment is filtered in the first stage if it has an alignment score less than this threshold, and there are algorithms in the second stage.

`-D INT`

Specifies the mapping quality threshold for the first stage. An alignment is filtered in the first stage if it has a mapping quality less than this threshold, and there are algorithms in the second stage.

`-K`

Specifies not to keep mappings from the first stage for the second stage. If this option is not given, the mappings from the first stage are added to any mappings from the second stage as candidates.

1.9 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 1.2 for common options that are in use in this command.

1.9.1 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 (**?**). 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` 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 an SFF file and the `-Y` option is used, then a comment line will be placed in the header with the following tags: RG, FO, and KS. The RG tag in the comment is the associated read group ID. 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

RG

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

PG

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

MD

This tag stores the MD array. The goal of this array is to follow the conventions in SAMTools (**?**). 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.

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.

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.

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.

XG

This tag stores the number of indel extensions (gap extensions) 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.

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.

XE

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

XI

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

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.

XZ

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

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.

Chapter 3

Commonly Asked Questions

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 unless the `-a 1` option and multiple threads (`-n`) are used. Using the `-a 1` option will output random hits for reads with multiple equaly likely mapping locations. TMAP uses a deterministic random seed so that when using only one thread the same random choice will be made. Nevertheless, when using multiple threads (`-n`), the order in which the random function call is made by the threads is determined by the operating system's scheduling of the threads. Therefore, the random function may return a different random mapping for the same read upon a second execution of TMAP.

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.

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.