MiTCR User Manual

System requirements

Java version 7 or higher should be installed on the system: JRE for simple analysis or JDK for MiTCR API usage. Latest Java release could be downloaded from here:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Distributives

Windows

MiTCR is distributed in two formats for the Windows platform:

- **installer** for MiTCR application[1] (MiTCRInstaller.exe)
- **cross-platform jar file distributive** (mitcr.jar)

The advantage of **installer** is automatic detection of available memory in the system and setting the Xmx parameter of Java Virtual Machine appropriately.

Usage

MiTCR is a command line application. In case of installation via MiTCRInstaller.exe the following command line should be executed from console to run MiTCR:

```
$ mitcr <options> <input file name> <output file name>
```

The example command that will run MiTCR with default parameters (see “Command line parameters” section) on input fastq file and produce the tab-delimited result file is the following:

```
$ mitcr -pset flex in.fastq.gz result.txt
```

Cross-platform

The common way to run MiTCR on any platform with Java support is a direct execution from the jar. In this case the command line to run MiTCR is the following:

```
$ java -Xmx<M> -jar mitcr.jar <options> <input file name> <output file name>
```

, where ***M*** - sets the maximum available memory for MiTCR[2].
Approximate memory amounts needed for the analysis are:

- 2 Gb for average diversity samples (~ 100,000 clonotypes)
- 10 Gb for highly diverse samples (~ 2,000,000 clonotypes)

Here is an example command:

```
$ java -Xmx10g -jar mitcr.jar -pset flex in.fastq.gz result.txt
```

---

[1] produced by Launch4J, NSIS

[2] See official java documentation for details

Source

The source code for stable versions of MiTCR can be downloaded from the MiTCR web site (http://mitcr.milaboratory.com/). Most recent development snapshots of MiTCR can be obtained from our source code repository at Bitbucket (https://bitbucket.org/milaboratory/mitcr).

Using MiTCR

Input

MiTCR accepts sequencing data in the fastq format. It can also directly read the data from a gzip-compressed input file (file name must end with ".gz"). Sequence quality information in the fastq file can be coded with byte offset equal to 33 (Sanger) or 64 (Solexa), in the later case additional `-solexa` option should be added to the parameters list (see **Other useful command line options** below).

Output

MiTCR supports two output formats:

- `.cls` - this format is a binary file containing clonotype information to be viewed with MiTCR-Viewer. This output format will be used if output file name ends with ".cls", in all other cases a tab-delimited format will be used.
- `tab-delimited` - this format is a plain text file containing clonotype information formatted as a simple table with columns separated by a tab symbol. Additionally, if the file name ends with ".gz" (e.g. `cloneset.txt.gz`) it will be automatically compressed using gzip. Three verbosity level can be selected using `-level` option:

  `-level 1` - default output verbosity level. The following columns will be outputted:

  Read count - the count of reads assigned to this clonotype

  Percentage - the fraction of this clone in the sample (equals to the read count divided by the total number of used reads)

  CDR3 nucleotide sequence - the nucleotide sequence of complementarity determining region 3 (see `-cysphe` option)

  CDR3 amino acid sequence - the amino acid sequence of CDR3

  V segments - the list of possible V segments for this clonotype (the ambiguity is associated with the homology of some segments, as well as possibly insufficient size of sequenced region)

  J segments - the list of possible J segments for this clonotype

  D segments - the list of possible D segments for this clonotype

  `-level 2` - medium verbosity level. Adds the following columns to those listed for level 1:

  Last V nucleotide position - position of the last nucleotide of V segment (zero based)

  First D nucleotide position - position of the first nucleotide of D segment or -1 if D segment was not found in this CDR3 (zero based)

  Last D nucleotide position - position of the last nucleotide of D segment or -1 if D segment was not found in this CDR3 (zero based)

  First J nucleotide position - position of the first nucleotide of J segment (zero based)

  VD insertions - the number of inserted nucleotides between V and D segments or -1 if D segment was not found in this

DJ insertions - the number of inserted nucleotides between D and J segments or -1 if D segment was not found in this

Total insertions - the total number of inserted nucleotides. Sum of VD and DJ insertions, if D segment was found, or number of insertions between V and J segments, if not.

`-level 3` - "full" verbosity level. If this level is used additional line containing technical information for deserialization using MiTCR API will be added to the header of output file. Adds following columns to those listed for level 1 and 2:

CDR3 nucleotide quality - PHRED quality string (33 based) composed of the maximum quality values for each nucleotide in the CDR3 sequence

Min quality - minimal nucleotide quality value (zero based, 2 is a minimum value if well formatted Illumina fastq files where used in analysis)

V alleles - the list of possible V alleles for this clonotype (the ambiguity is associated with the homology of some segments and alleles, as well as possibly insufficient size of sequenced region)

J alleles - the list of possible J alleles for this clonotype

D alleles - the list of possible D alleles for this clonotype

Command line parameters

MiTCR is highly customizable, there is a large number of parameters for each module in the analysis pipeline. There are several combinations of parameters aka presets (developed based on our experience), that perform well on datasets obtained using certain sequencing library preparation strategies. Therefore the parameter setting is performed in the following way:

The base parameter set is selected from the list of parameter presets, either built-in or previously stored by the user. After that, some parameters could be overridden through the command line options. For convenience the resulting parameter set could be stored as XML file to be used in the future.

Base parameter set is specified through `-pset` command line option. Here is the example of loading most commonly used (`flex`) preset:

```
$ mitcr -pset flex input.fastq.gz output.txt
```

this command will process the input file directly from gzipped FASTQ using built-in `flex` parameter preset and will create an `output.txt` file with calculated clones, formatted as tab-delimited table.

Built-in parameter sets

We provide two parameter presets that differ in configuration of J segment mapper and are fine tuned to give the best performance in certain datasets:

- `flex` - Alignment of V and J segments starts from 5-mer surrounding the conserved amino acid (cysteine and phenylalanine respectively). Then alignments are expanded in both directions, i.e. inside and outside of CDR3.
  Target datasets: with PCR primers designed such that regions upstream of conserved phenylalanine and downstream of conserved cysteine of CDR3 are not overlapped by primers
- `jprimer` - Begins alignments of V and J segments starting from 5-mer around the conserved amino acid (cysteine and phenylalanine respectively). Then the alignment for V is expanded in both (inside CDR3 and outside CDR3) directions and the alignment for J is expanded only inside CDR3.
  Target datasets: with sequence of J segment upstream conserved phenylalanine biased by PCR primer annealing.

Overriding of default parameters

Overriding of parameters form the initial set is performed using the following command-line options:

`-species <species>`	overrides target species **values:** `hs` = Homo sapiens `mm` = Mus musculus **default value for built-in parameters:** `hs`
`-gene <gene>`	overrides target gene **values:** `TRB` = beta chain of TCR `TRA` = alpha chain of TCR **default value for built-in parameters:** `TRB`
`-cysphe <0|1>`	overrides CDR3 definition. Determines whether to include bounding cysteir and phenylalanine into CDR3. **values:** `0` = do not include cys & phe into CDR3 `1` = include cys & phe into CDR3 **default value for built-in parameters:** `1`
`-ec <0|1|2>`	overrides the error correction level **values:** `0` = don't correct errors; for preliminary analysis `1` = correct low quality sequencing errors only *(see -quality and -lq options for details)*; for preliminary analysis `2` = also corrects PCR errors and high quality sequencing errors *(see -pcrec option)* **default value for built-in parameters:** `2`
`-quality <value>`	overrides the quality threshold value for segment alignment and low quality sequences correction algorithms. **values:** PHRED quality value

	`0` tells the program not to use quality information **default value for built-in parameters:** `25`
`-lq <drop | map>`	overrides low quality CDR3s processing strategy **values:** `drop` = filter off reads that contain low quality (PHRED quality value less than 25 by default or as specified by `-quality` parameter) nucleotides within CDR3 `map` = map reads that contain low quality (PHRED quality value less than 25 used by default or as specified by `-quality` parameter) nucleotides within CDR3 onto clonotypes created from the high quality CDR3s **note:** this option makes no difference if quality threshold (`-quality` option) is set to 0, or error correction level (`-ec`) is 0. **default value for built-in parameters:** `map`
`-pcrec <smd | ete>`	overrides the PCR and high quality sequencing errors correction algorithm **values:** `smd` = "save my diversity" corrects PCR errors and high quality sequencing errors in germline regions only (corrects 65-85% of all errors with minimal risk to lose real TCR diversity) `ete` = "eliminate these errors" maximal correction of errors (each sinlge mismatch within CDR3 is considered as possible error) with a risk of losing a minor portion of real TCR diversity **default value for built-in parameters:** `ete`

Here are several other examples of MiTCR usage:

```
$ mitcr -pset flex -species mm -gene TRA -quality 30 input.fastq output.cls
```

This command is applicable for dataset containing sequences of mouse CDR3s of alpha chain of T-cell receptor with relatively high average sequence quality (roughly speaking average nucleotide PHRED quality is higher than 35).

```
$ mitcr -pset jprimer -gene TRA -ec 0 input.fastq output.cls
```

This command performs rather raw analysis of alpha subunits of T-cell receptor in human samples. The `jprimer` parameter set is intended on analysis of libraries amplified using multiplex PCR with primers on J region of TRA gene. The option `-ec 0` tells the program not to correct sequence errors, so all CDR3 sequences (including erroneous ones) extracted from reads will be present in the output. This strategy is useful to make some checks on the sample quality and performance of error correction machinery of MiTCR.
Quality parameter (`-quality`) which is a PHRED Quality Score threshold could be tuned to allow tradeoff between true yield and false-diversity. Quality parameter for the core clonotypes formation can be routinely defined as 25 to build clonotypes with the maximal confidence. Alternatively, allowed quality level for the core clonotypes can be decreased to yield maximal diversity. Minimal quality set to zero indicates that the user wishes to build all possible clonotypes without taking into account quality values (such analysis strategy is useful for preliminary raw data overview). Low quality mapper could also be turned off by user, in this case low quality CDR3s that were extracted from raw reads will be dropped (`-lq drop`).

Other useful command line options

`-limit <sequences>`	limits the number of input sequencing reads, use this parameter t normalize several datasets or to have a glance at the data
`-export <new name>`	use this option to export current parameters set to a local xml file (se "exporting parameters" section)
`-report <file name>`	turns on the reporting and sets the name of report file. Report contain information on bulk characteristics of dataset, resulting clone set an analysis process. **note:** The same file name could be used for several invocations of mitcr, i this case report information will be appended to the end of the fil This is the recommended usage of `-report` option, e.g. see th following shell script to analyze all fastq files in the current folde generating a single report file (only for *nix platforms) `$ for file in *.fastq.gz; do mitcr -pset flex -report report.txt $file ${file%.*.*}.txt; done;`
`-level < 1 | 2 | 3 >`	verbosity level for tab-delimited output (see "output formats" section fo details). Has no effect if cls is used as output format. **values:** `1` = simple `2` = medium `3` = full, clone sets in this format could be deserialized using mitcr API **default value:** `3`

`-solexa`	sets the input format of quality strings in fastq files to old illumina format (< Casava 1.3) with 64 offset
`-h`	prints help
`-v`	prints version information

XML parameters (advanced options)

Exporting parameters to file

For convenience sake the whole parameter set could be exported for further usage. This is done through `-export` command line option. Consider the following command:

```
$ mitcr -pset flex -species mm -gene TRA -quality 30 -export myflex
```

This command will create a new file named `myflex`, that will contain XML formatted list of analysis parameters, including all modifications that were made through command line options. To further use this preset simply add `-pset` option with a specified name:

```
$ mitcr -pset myflex input.fastq output.cls
```

Sample parameters file

Here is the content of exported `myflex` file from the previous section:

```
<?xml version="1.0" encoding="UTF-8"?>
<parameters>
  <gene>TRA</gene>
  <species>MusMusculus</species>
  <qualityInterpretationStrategy>
    <illumina>30</illumina>
  </qualityInterpretationStrategy>
  <cdr3Extractor>
    <v>
      <extensionDirections>both</extensionDirections>
      <seedFrom>-4</seedFrom>
      <seedTo>1</seedTo>
      <minAlignmentMatches>12</minAlignmentMatches>
      <lengthTolerance>3</lengthTolerance>
    </v>
    <j>
      <extensionDirections>both</extensionDirections>
      <seedFrom>-1</seedFrom>
      <seedTo>4</seedTo>
      <minAlignmentMatches>12</minAlignmentMatches>
      <lengthTolerance>2</lengthTolerance>
    </j>
    <d>
      <minLength>3</minLength>
      <searchForRC />
    </d>
```

```
    <upperCDR3LengthThreshold>100</upperCDR3LengthThreshold>
    <lowerCDR3LengthThreshold>10</lowerCDR3LengthThreshold>
    <strand>both</strand>
    <includeCysPhe />
   </cdr3Extractor>
   <cloneGenerator>
    <segmentInformationAggregationFactor>0.15</segmentInformationAggregationFactor>
    <maxErrorsInBadPoints>3</maxErrorsInBadPoints>
    <proportionalMapping />
   </cloneGenerator>
   <clusterizer>
    <type>oneMismatch</type>
    <maxClusterizationRatio>0.33</maxClusterizationRatio>
   </clusterizer>
  </parameters>
```

The editing of exported presets is a good way to tune the analysis pipeline. XML offers additional options compared to command line parameters. All XML encoded parameters are tightly connected to parameters in the Java API, detailed description of XML file structure is present in the next section.

Structure of parameters file

Majority of items in the XML parameters file are grouped according to the structure of analysis pipeline while first three elements determine global parameters of analysis process. Structure of parameters file is summarized in the following tables:

`<gene>`	Sets the chain of T-cell receptor to analyse Allowed values: • `TRA` (for alpha chain of TCR) • `TRB` (for beta chain of TCR)
`<species>`	Sets the species from which the sample is derived Allowed values: • `HomoSapiens` • `MusMusculus`
`<qualityInterpretationStrategy>`	Tells MiTCR how to process quality information provided with sequencing read Allowed values: • `<dummy>` (do not take into account quality information) • `<illumina>` **Q** `</illumina>` (set the quality threshold, where 0 < Q < 45)
`<cdr3Extractor>`	Parameters of CDR3 extractor (see the corresponding section for details)

`<cloneGenerator>`	Parameters of clonotypes generator (see the corresponding section for details)
`<clusterizer>`	Parameters of clonotypes clusterizer aka PCR error correction (see the corresponding section for details)

<cdr3Extractor> options

`<v>`	Sets parameters of algorithm used to build alignments with V gene segment (see “V/J alignment parameters” section)
`<j>`	Sets parameters of algorithm used to build alignments with J gene segment (see “V/J alignment parameters” section)
`<d>`	Sets parameters of algorithm used to build alignments with D gene segment (see “D alignment parameters” section)
`<upperCDR3LengthThreshold>`, `<lowerCDR3LengthThreshold>`	Sets the upper and lower limits for CDR3 length (in nucleotides). If length of extracted CDR3 will be outside this limits such CDR3 will be discarded.
`<strand>`	Determines in what strand(s) of sequencing read MiTCR should search for CDR3s Allowed values: • `forward` • `reverseComplement` • `both`
`<includeCysPhe />`	If this option is provided bounding Cys and Phe will be included into CDR3 sequence.

V/J alignment parameters

`<extensionDirections>`	Sets the direction(s) of alignment extension performed after the seed sequence is found Allowed values: • `both` • `insideCDR3` • `outsideCDR3`
`<seedFrom>, <seedTo>`	Sets positions of the seed region inside corresponding gene segment. The position is set in coordinates relative to the reference nucleotide. **The following convention of reference nucleotide is used for segment alleles throughout MiTCR API:** Reference nucleotide for V segment is the next nucleotide after codon of conserved Cys; for J segment is the nucleotide previous to codon of conserved Phe.

<minAlignmentMatches>	Sets the minimal number of matches for the alignment
<lengthTolerance>	(internally used parameter) Allowed value: 3 if extensionDirections = both 2 if extensionDirections != both

D alignment parameters

<minLength>	Minimal number of nucleotides to determine D region
<searchForRC />	Add this parameter to search for the reverse-complement sequence of D gene segment

<cloneGenerator> options

<segmentInformationAggregationFactor>	(internally used parameter) Allowed value: 0.15
<maxErrorsInBadPoints>	Sets the maximal number of mismatches in bad quality nucleotide (having quality value less then threshold specified in <qualityInterpretationStrategy>) to map bad CDR3 read onto the core clonotype. If this option is omitted mapping of bad CDR3s will be turned off.
<proportionalMapping>	(internally used parameter) Should always be present for correct clonotype formation
<filterOffLQReads />	If this option is provided, all bad CDR3s will be dropped.

<clusterizer> options

<type>	Allowed values: ● oneMismatch - equivalent of ete from the command line -pcrec option ● v2d1j2t3Explicit - equivalent of smd from the command line -pcrec option ● none
<maxClusterizationRatio>	Sets maximal ratio between read counts of two clonotypes to be clusterized together

Sample Output

Here are several lines from a file obtained by running MiTCR on sample dataset (http://files.milaboratory.com/sample.fastq.gz) using the following command:

```
$ mitcr -pset flex -level 1 sample.fastq.gz sample_output.txt
```

Read count	Percent	CDR3 nucleotide sequence	CDR3 amino acid sequence	V segments	J segments	D segments
11763	0.131	TGTGCCAGCAGCTTAGGGGAAAACATTCAGTACTTC	CASSLGENIQYF	TRBV13	TRBJ2-4	
4635	0.052	TGTGCCAGCACCGTGGACAGTCTGGACACTGAAGCTTTCTTT	CASTVDSLDTEAFF	TRBV12-4, TRBV12-3	TRBJ1-1	TRBD1
4470	0.050	TGCAGCGTTGAAATTTGGGATAGCTCCTACAATGAGCAGTTCTTC	CSVEIWDSSYNEQFF	TRBV29-1	TRBJ2-1	
2333	0.026	TGTGCCAGCAGCTTAGCGCCGGGAGCAACTAATGAAAAACTGTTTTTT	CASSLAPGATNEKLFF	TRBV7-6	TRBJ1-4	TRBD2
1448	0.016	TGTGCCATCAAGACGACTAGCGGGATTGTGGATGAGCAGTTCTTC	CAIKTTSGIVDEQFF	TRBV10-3	TRBJ2-1	TRBD2
1423	0.016	TGTGCCAGCAGGAACACCTACGAGCAGTACTTC	CASRNTYEQYF	TRBV27	TRBJ2-7	
...	...	...	...	...	...	...