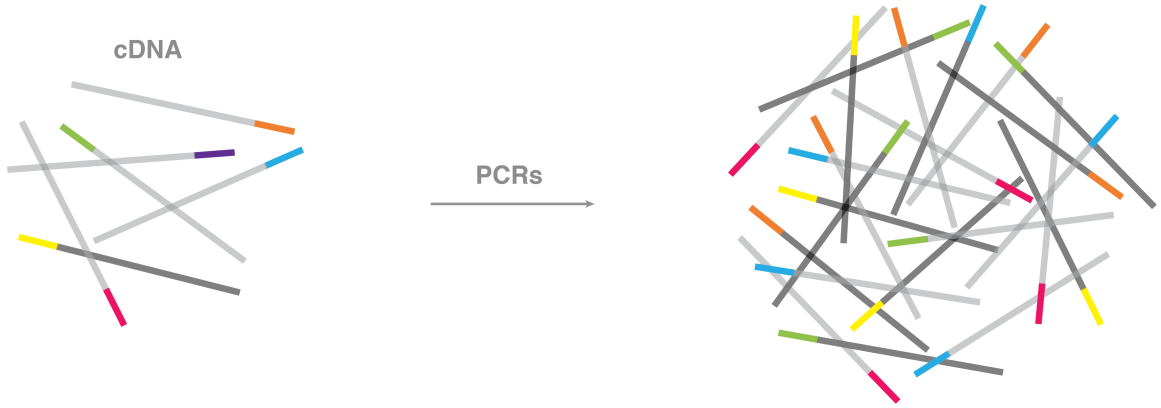
MIGEC Documentation

Release SNAPSHOT

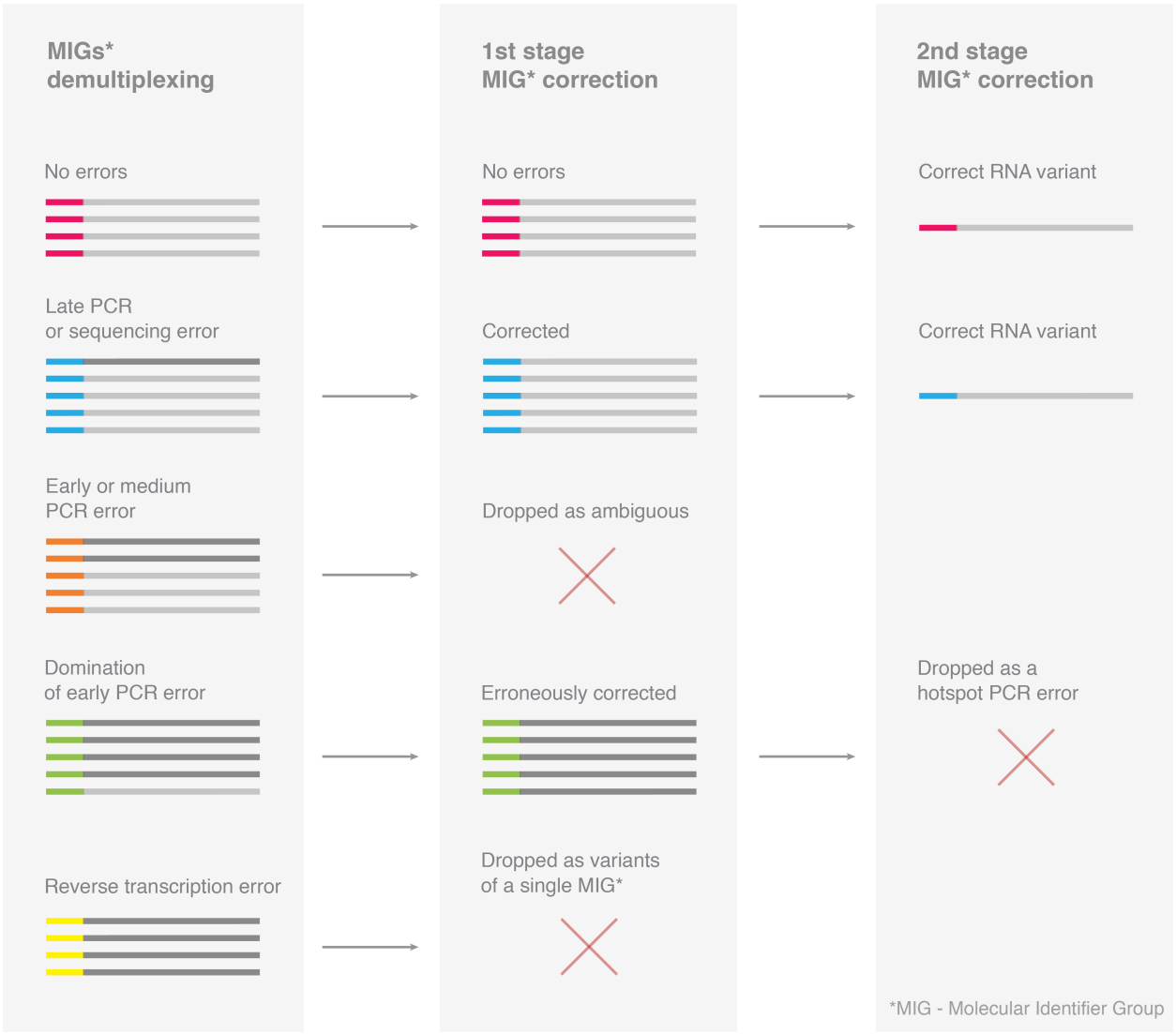
Mikhail Shugay

May 27, 2018

Contents



MIGEC pipeline



CHAPTER 1

FEATURES

- De-multiplexing, adapter trimming and read overlapping for NGS data. Extraction of UMI sequences
- Assembly of consensuses for original molecules that entered library preparation by grouping reads with identical molecular identifiers
- Mapping of V, D and J segments, extraction of CDR3 regions and clonotype assembly for all human and mouse immune receptors (TRA/TRB/TRG/TRD and IGH/IGL/IGK)
- Additional filtering of hot-spot errors in CDR3 sequence
- Flexible and straightforward batch processing
- Currently all species-gene pairs that have germline sequences (based on IMGT) that allow CDR3 identification are supported

Species	Gene
HomoSapiens	TRA, TRB, TRG, TRD, IGL, IGK, IGH
MusMusculus	TRB, TRG, TRD, IGL, IGK, IGH
MacacaMulatta	TRB, IGK, IGH
OryctolagusCuniculus	IGL, IGK, IGH
RattusNorvegicus	IGL, IGH
CanisLupusFamiliaris	TRB, TRG
SusScrofa	IGL, IGK
BosTaurus	TRD
MusSpretus	IGL
GallusGallus	TRB
AnasPlatyrhynchos	TRB

CHAPTER 2

Table of contents:

2.1 Installing and running

The pipeline is written in Groovy (a Java scripting language) and distributed as an executable JAR. To install it get the latest JRE and download the executable from releases section.

To ran a specific script from the pipeline, say **Checkout**, execute

```
java -jar MIGEC-$VERSION.jar Checkout [arguments]
```

Where `$VERSION` stands for pipeline version (e.g. 1.2.1), this notation is omitted in MIGEC routine documentation.

To view the list of available scripts execute:

```
java -jar MIGEC-$VERSION.jar -h
```

alternatively you can download the repository and compile it from source using Maven (requires Maven version 3.0)

```
git clone https://github.com/mikessh/MIGEC.git
cd MIGEC/
mvn clean install
java -jar target/MIGEC-$VERSION.jar
```

This should show you the list of available MIGEC routines.

Note: The data from 454 platform should be used with caution, as it contains homopolymer errors which (in present framework) result in reads dropped during consensus assembly. The 454 platform has a relatively low read yield, so additional read dropping could result in over-sequencing level below required threshold. If you still wish to give it a try, we would recommend filtering off all short reads and repairing indels with Coral, the latter should be run with options `-mr 2 -mm 1000 -g 3`.

Warning: NCBI-BLAST+ package is required. Could be directly installed on Linux using a command like $sudo apt-get ncbi-blast+ or downloaded and installed directly from here: ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

Warning: Consider providing sufficient memory for the pipeline, i.e. 8Gb for MiSeq or 36Gb for HiSeq sample, depending on sample sequence diversity and current script (CdrBlast requires has the highest memory requirements). To do so, execute the script with `-Xmx` argument: `java -Xmx8G -jar MIGEC-$VERSION.jar CdrBlast [arguments]`. If insufficient amount memory is allocated, the Java Virtual Machine could drop with a *Java Heap Space Out of Memory* error.

2.2 The pipeline

All routines in the pipeline are available in **manual** and **batch** variants. Batch variants are designed to automatically handle several input samples with minimal shell scripting glue between analysis steps, this is also the recommended way to use with MIGEC.

List of MIGEC batch routines:

- *Checkout-batch*
- *MIG statistics*
- *Assemble-batch*
- *CdrBlast-batch*
- *FilterCdrBlastResults-batch*

If the "barcodes" file is set properly, the entire pipeline can be written as following:

```
MIGEC="java -Xmx8G -jar MIGEC-$VERSION.jar"
$MIGEC CheckoutBatch -cu barcodes.txt checkout/
$MIGEC Histogram checkout/ histogram/
$MIGEC AssembleBatch -c checkout/ histogram/ assemble/
$MIGEC CdrBlastBatch -R TRB checkout/ assemble/ cdrblast/
$MIGEC FilterCdrBlastResultsBatch cdrblast/ cdrfinal/
```

2.2.1 Manual usage

List of MIGEC manual routines:

- *Checkout-manual*
- *MIG statistics*
- *Assemble-manual*
- *CdrBlast-manual*
- *FilterCdrBlastResults-manual*

An example for a 300bp paired-end MiSeq run of IGH library on a 16Gb RAM Unix server. Such sequencing read length allows complete IGH sequencing, thus mate pairs overlap. First *barcodes.txt* should be created containing adapter sequences, see the section below for guidelines. Then, assuming that the corresponding FASTQ files are *IGH_SAMPLE_R1.fastq.gz* and *IGH_SAMPLE_R2.fastq.gz*, UMI- and multiplex index-containing adapter is near

5'UTR of V segment (so the CDR3 is in mate#2 after reads are oriented) and NCBI-BLAST+ is installed, run all 5 stages of the pipeline using the following command:

```
$MIGEC Checkout -cute --overlap barcodes.txt IGH_S1-10_R1.fastq.gz IGH_S1-10_R2.fastq.
↪gz checkout/
$MIGEC Histogram checkout/ histogram/
$MIGEC Assemble -c --mask 0:0 checkout/S1_R12.fastq.gz . assembly/
$MIGEC CdrBlast -R IGH checkout/S1_R12.fastq.gz cdrblast/S1_raw.txt
$MIGEC CdrBlast -a -R IGH assembly/S1_R12.fastq.gz cdrblast/S1_asm.txt
$MIGEC FilterCdrBlastResults cdrblast/S1_asm.txt cdrblast/S1_raw.txt cdrfinal/S1.txt
```

Note: As dot (.) is reserved by MIGEC to specify empty metadata fields and unused arguments, use *./* in case you want to point to current directory.

2.2.2 Full-length immunoglobulin data analysis

MIGEC can be used as pre-processing tool to assemble full-length consensuses for further post-analysis with HIgBlast tool. Note that due to typically poor quality of MiSEQ 300+300bp reads, the `--overlap` mode of Checkout routine is not guaranteed to perform well. Instead, we recommend to assemble consensuses first and then perform overlapping using external tools. For example, MiTools `merge` action can be used with the `--same-strand` option specified, the latter is critical as assembled consensuses are on the same strand in output in contrast to normal orientation of Illumina reads.

Consensus quality and overlap efficiency can be greatly improved using the `--only-first-read` option of Histogram and Assemble routines. If set, this option instructs routines to use only the first read that typically has higher quality than the second one. This applies to non-oriented reads and works better for asymmetric sequencing design, e.g. 400+200bp reads.

2.3 De-multiplexing

2.3.1 Checkout-batch

Description

A script to perform de-multiplexing and UMI tag extraction for a set of FASTQ files that were previously split using Illumina sample indices.

Usage

General:

```
java -jar migec.jar CheckoutBatch [options] barcodes_file output_dir
```

The barcodes file specifies sample multiplexing and UMI (NNN.. region) extraction rules. It has the same structure as for "manual" Checkout (see section below), with additional two columns that specify input FASTQ file names.

Sample ID	Master barcode sequence	Slave barcode sequence	Read#1 FASTQ	Read#2 FASTQ
S0	acgtacgtAGGTTAcadkgag			
S1	acgtacgtGGTTAAcadkgag	ctgkGTTCaat	ILM1_R1_L001.fastq.gz	ILM1_R2_L001.fastq.gz
S1	acgtacgtAAGGTTcadkgagNNNNNN		ILM2_R1_L001.fastq.gz	ILM2_R2_L001.fastq.gz
S3	acgtacgtTAAGGTcadkgagNNNNNN	NNNNNNctgkGTTCaat	ILM1_R1_L001.fastq.gz	ILM1_R2_L001.fastq.gz

The following rules apply:

- All specified FASTQ files are sequentially processed using Checkout
- If no FASTQ file is specified for a given barcode, it will be searched in all FASTQ files
- CheckoutBatch will properly aggregate reads from multiple FASTQ files that have the same sample id
- Still there should not be the case when a FASTQ file has the same barcode specified more than once

Parameters

Same as in manual version of Checkout, see below.

Output format

The Checkout routine produces files in the FASTQ format that have a specific `UMI` field added to the header. Each read successfully matched by Checkout will be output as follows:

```
@ILLUMINA_HEADER UMI:NNNN:QQQQ
ATAGATTATGAGTATG
+
##II#IIIIIIIIII
```

The original read header (`ILLUMINA_HEADER` here) is preserved, the appended `UMI:NNNN:QQQQ` contains the sequence of the UMI tag (`NNNN` bases) and its quality string (`QQQQ`).

2.3.2 Checkout-manual

Description

A script to perform de-multiplexing and UMI tag extraction

Usage

General:

```
java -jar migec.jar Checkout [options] barcodes_file R1.fastq[.gz] [. or R2.fastq[.
↪gz]] output_dir
```

For paired-end data:

```
java -jar migec.jar Checkout -cute barcodes.txt R1.fastq.gz R2.fastq.gz ./checkout/
```

For unpaired library:

```
java -jar migec.jar Checkout -cute barcodes.txt R.fastq.gz . ./checkout/
```

For overlapping paired reads:

```
java -jar migec.jar Checkout -cute --overlap barcodes.txt R1.fastq.gz R2.fastq.gz .␣
↪checkout/
```

accepted *barcodes.txt* format is a tab-delimited table with the following structure:

Sample ID	Master barcode sequence	Slave barcode sequence
S0	acgtacgtAGGTTAcadkgag	
S1	acgtacgtGGTTAAcadkgag	ctgkGTTCaat
S2	acgtacgtAAGGTTcadkgagNNNNNN	
S3	acgtacgtTAAGGTcadkgagNNNNNN	NNNNNNctgkGTTCaat

A sequencing read is scanned for master adapter and then, if found, its mate is reverse-complemented to get on the same strand as master read and scanned for slave adapter.

- Slave adapter sequence could be omitted.
- Adaptor sequence could contain any IUPAC DNA letters.
- Upper and lower case letters mark seed and fuzzy-search region parts respectively.
- *N* characters mark UMI region to be extracted.
- Multiple rows could correspond to the same sample
- In order to be able to run batch pipeline operations, all samples should contain UMI region of the same size

For example, in case *S2* **Checkout** will search for *AAGGTT* seed exact match, then for the remaining adapter sequence with two mismatches allowed and output the *NNNNNN* region to header. In case *S3* in addition the slave read is scanned for *GTTC* seed, fuzzy match to the rest of barcode is performed and *NNNNNN* region is extracted and concatenated with UMI region of master read.

Parameters

General:

`-c` compressed output (gzip compression).

`-u` perform UMI region extraction and output it to the header of de-multiplexed FASTQ files

`-t` trim adapter sequence from output.

`-e` also remove trails of template-switching (poly-G) for the case when UMI-containing adapter is added using reverse-transcription (cDNA libraries).

`--overlap` will try to overlap reads (paired-end data only), non-overlapping and overlapping reads will be placed to *_R1/_R2* and *_R12* FASTQ files respectively. While overlapping the nucleotide with higher quality will be taken thus improving overall data quality.

`--overlap-max-offset X` controls to which extent overlapping region is searched. **IMPORTANT** If the read-through extent is high (reads are embedded) should be set to ~40.

Barcode search:

`-o` speed up by assuming that reads are oriented, i.e. master adapter should be in R1

`-r` will apply a custom RC mask. By default it assumes Illumina reads with mates on different strands, so it reverse-complements read with slave adapter so that output reads will be on master strand.

`--rc-barcodes` also searches for both adapter sequences in reverse complement. Use it if unsure of your library structure.

`--skip-undef` will not store reads that miss adapter sequence to save drive space.

Note: When there is a huge number of unassigned/unused reads `--skip-undef` option greatly speeds up de-multiplexing. However, take care to carefully investigate the reasons behind low barcode extraction rate if it is a case.

Important: The `--overlap` option may not perform well for poor quality reads, which is a typical situation for 300+300bp MiSEQ sequencing. In this case, merging reads using external software after Assemble stage is recommended.

2.4 MIG statistics

Description

A script to generate consensus coverage statistics, i.e. molecular identifier group (MIG) size distribution.

Usage

General:

```
java -jar migec.jar Histogram checkout/ histogram/
```

Running this script will generate several files in *histogram* folder, the one important for basic data processing is *overseq.txt*. The header of table contains MIG sizes (in log2 scale), while each row corresponds to a de-multiplexed sample contains the number of reads in MIGs of a given size (cumulative abundance).

For a decent dataset the plot of cumulative abundance display a small peak at MIG size of 1 that could be attributed to erroneous MIGs and has an exponential decline, and a clear peak at MIG size of 10+ containing amplified MIGs. Those erroneous MIGs could arise as experimental artifacts, however the most common reason for their presence is an error event in UMI sequence itself. Note that the latter is only valid when number of distinct UMIs is far lower than theoretically possible UMI diversity (e.g. 4^12 for 12-letter UMI regions)!

MIG size cutoff in **Assemble** should be set to dissect erroneous MIGs while retaining amplified ones. If peaks overlap collision filtering should be considered.

A simple plotting routine written in R can facilitate visualization of MIG size distributions, available here.

Important: The `--only-first-read` option should be used if it is also specified in Assemble routine for consensus coverage estimates to be concordant.

2.5 Consensus assembly

2.5.1 Assemble-batch

Description

A script to perform UMI-guided assembly

Usage

General:

```
java -jar migec.jar AssembleBatch [options] checkout_output_folder/ histogram_output_
↪folder/ output_folder/
```

Performs a batch assembly for all FASTQ files produced by checkout, all assembly parameters are set according to **Histogram** output.

One can specify a default mask telling for paired-end reads which mate(s) to assemble. The mask is provided by `--default-mask <R1=[0,1]:R2=[0,1]>` argument, i.e. to assemble only second mate use `--default-mask 0:1`. This speeds-up the assembly. Also, by default the mask is `1:1`, so for each MIG an output consensus pair is created only if both consensuses are successfully assembled. In case of `0:0` mask will process only overlapped reads. Remember that during **Checkout** reads get re-oriented so they are on the same strand, corresponding to the strand of *Master* barcode and the read with *Master* barcode is assigned with *_R1* index.

A sample metadata file could also be provided with `--sample-metadata <file_name>` argument to guide the batch assembly. This file should have the following tab-separated table structure:

Sample ID	File type	Mask
S0	paired	1:0
S0	overlapped	
S1	unpaired	
S2	paired	0:1

Note that *S0* is present with two file types, as when performing read overlap **Checkout** stores non-overlapped reads in *_R1/_R2* files, which could be then incorporated into data processing.

The `--force-overseq X` and `--force-collision-filter` will force a MIG size threshold of `X` and filtering of 1-mm UMI collisions for all samples being processed.

Warning: In most cases, the automatic MIG size threshold selected by Histogram routine is ok. However we strongly recommend manual inspection of Histogram output files and considering to manually specify an appropriate MIG size threshold for input samples. Our experience also shows that it is a good practice to set an identical size threshold for all samples in a batch.

Output format

The assembled consensuses are also stored in FASTQ format, however the read quality is now proportional to the relative frequency of the most frequent base at a given position. I.e. if 100% of bases in the position 5 are `A` the quality score will be Phred40 (`I`), while if only 27.1% of the bases are `A` and other bases have a frequency of 24.3% the quality will be Phred2 (`#`):

```
@MIG UMI:NNNN:X
ATAGATTATGAGTATG
+
##II#IIIIIIIIII
```

The UMI tag is also added to the header, the number `X` in the UMI tag field is the total number of reads that were assembled into a given consensus.

2.5.2 Assemble-manual

Description

A script to perform UMI-guided assembly

Usage

General:

```
java -jar migec.jar Assemble [options] R1.fastq[.gz] [. or R2.fastq[.gz]] output_
↪folder
```

Unpaired and overlapped FASTQ:

```
java -jar migec.jar Assemble -c checkout/S1_R0.fastq.gz . assembly/
```

Paired FASTQ:

```
java -jar migec.jar Assemble -c checkout/S1_R1.fastq.gz checkout/S1_R2.fastq.gz ./
↪assembly/
```

Paired FASTQ with only second read to be assembled:

```
java -jar migec.jar Assemble -c --mask 0:1 checkout/S1_R1.fastq.gz checkout/S1_R2.
↪fastq.gz assembly/
```

All reads are grouped by their UMI and then read groups (aka molecular identifier groups, MIGs) with >10 reads (default value, see **Histogram** section for details on setting it) are assembled. Multiple alignment is performed and consensus sequence is generated. Note that for paired reads both consensuses should be successfully assembled, otherwise the pair is dropped.

Automatic output file naming convention is used for compatibility with batch operations. Output file name will be appended with _R0 for unpaired FASTQ file, with either _R1 and _R2 for the corresponding paired FASTQ file and with _R12 for overlapped FASTQ file. Output file name will also include MIG size threshold used.

Settings

The `--mask <R1=[0,1]:R2=[0,1]>` parameter indicates FASTQ files to be assembled in paired-end data. By default both reads are assembled. In case of `0:0` mask will process only overlapped reads.

The `-c` option indicates compressed output.

The `-m` option sets minimum number of reads in MIG. This should be set according to Histogram script output to separate two peaks: over-sequenced MIGs and erroneous MIGs that cluster around MIG size of 1.

Note: To inspect the effect of such single-mismatch erroneous UMI sub-variants see "collisions" output of Histogram script. Such collision events could interfere with real MIGs when over-sequencing is relatively low. In this case collisions could be filtered during MIG consensus assembly using `--filter-collisions` option in **AssembleBatch** routine. When using **Assemble** routine use `--force-collision-filter` command to turn collision filter on. The child-to-parent ratio for collision filtering (max allowed size of larger and smaller UMIs that differ by a single mismatch) is controlled by the `--collision-ratio` parameter (default is `--collision-ratio 0.1`).

Important: The `--only-first-read` option can greatly improve assembly quality in case of poor second read quality and allows consensus assembly for asymmetric reads (e.g. 400+200bp sequencing design). If using this option, don't forget to set `--only-first-read` in Histogram util to correctly calculate MIG size threshold.

2.5.3 Summary statistics

MIG consensus assembly report will be stored in the `assemble.log.txt` file which contains basic information such as the fraction of assembled reads and the final number of assembled consensuses. It summarizes both filtering

based on MIG size and the consistency of read sequence within the same MIG:

- The `MIGS_*` counters show the number of MIGs that were processed (`MIGS_TOTAL`) and were successfully assigned with consensus sequences (`MIGS_GOOD_*`).
- In case of paired-end sequencing separate statistic is provided for both R1 and R2 (`MIGS_GOOD_FASTQ1` and `MIGS_GOOD_FASTQ2`). The total counter (`MIGS_GOOD_TOTAL`) reflects number of UMI tags for which both R1 and R2 MIGs were successfully assembled.
- The total number of reads in assembled MIGs and all MIGs is provided in `READS_GOOD_*` and `READS_TOTAL` columns respectively.

MIGs can be dropped from the assembly and marked as bad ones for the following reasons:

- MIGs with a size less then the specified size threshold value will be dropped (see `--force-overseq` and `-m` options), as well as MIGs that correspond to erroneous UMI variants (see `--filter-collisions` option).
- Reads that have too many mismatches when compared to the consensus sequence will be dropped, which is reflected by `READS_DROPPED_WITHIN_MIG` statistic. In case a high percentage of reads within MIG is dropped/final MIG size is less than the threshold the entire MIG will be dropped for the analysis.

Note: Additional pre-filtering of UMI tags identified by **Checkout** utility is performed by removing UMI tag sequences with a minimum Phred quality score below the one specified by the `-q` parameter (default is `15`). Thus, the `READS_TOTAL` can be somewhat smaller than the total number of reads for a given sample in the `checkout.log.txt`

2.6 V(D)J junction mapping

2.6.1 CdrBlast-batch

Description

A script to extract CDR3 sequences. Will properly combine reads coming from paired and overlapped data and perform analysis for both raw and assembled data.

Performs CDR3 extraction and V/J segment determination for both raw (**Checkout** output) and assembled-data. Gene parameter `-R` is required unless metadata (`--sample-metadata`) is provided that specifies gene for each sample; supported genes are *TRA*, *TRB*, *TRG*, *TRD*, *IGH*, *IGK* and *IGL*. If either of *assembly_output_folder* or *checkout_output_folder* is not specified, the processing will be done solely for the remaining input, this is useful e.g. if one wants quickly process the assembled data. Otherwise only samples and file types (paired, overlapped or single) that are present in both outputs will be used. Processing both raw and assembled data is required for second stage error correction (removal of hot-spot errors).

Usage

General:

```
java -jar migec.jar CdrBlastBatch [options] -R gene [checkout_output_folder/ or .]␣
↪[assemble_output_folder/ or .] output_folder
```

Several default **CdrBlast** parameters could be set,

`--default-mask <R1=[0,1]:R2=[0,1]>` - mask which specifies for which read(s) in paired-end data to perform CDR3 extraction. In case of `0:0` mask will process only overlapped reads `--default-species` - default species to be used for all samples, *human* (used by default) or *mouse* `--default-file-types` - default file types

(paired, overlapped or single) to be processed for each sample. If several file types are specified, the corresponding raw and assembled files will be combined and used as an input to CdrBlast

`--default-quality-threshold <Phred=[2..40],CQS=[2..40]>` - quality threshold pair, default for all samples. First threshold in pair is used for raw sequence quality (sequencing quality phred) and the second one is used for assembled sequence quality (CQS score, the fraction of reads in MIG that contain dominant letter at a given position) `--no-sort` - no sorting is performed for output files which speeds up processing. Could be safely used in full pipeline as FilterCdrBlastResults will provide final clonotype table in sorted format

A sample metadata file could also be provided with `--sample-metadata <file_name>` argument to guide the batch CDR3 extraction. This file should have the following tab-separated table structure:

Sample ID	Species	Gene	File types	Mask	Quality threshold pair
S0	human	TRA	paired, overlapped	1:0	25,30
S1	human	TRB	unpaired	•	25,30
S2	mouse	TRB,TRA	paired	0:1	20,25

See section below for more details.

Output format

The output of V-D-J mapping routines of MIGEC is a standard tab-delimited clonotype table with some information on the number of reads and UMI tags that correspond to a given clonotype.

Each clonotype is specified by count, fraction, V, D and J segment identifier list, CDR3 nucleotide and amino acid sequence.

The positions of last V nucleotide, first and last D nucleotide, and first J nucleotide specify the germline region markup within the hypervariable CDR3 sequence, they are given in 0-based coordinates where 0 marks the first base of CDR3.

The `total reads` and `good reads` fields contain the number of reads supporting a given clonotype prior to and after the quality filtering.

The `total events` and `good events` fields contain the number of UMI tags supporting a given clonotype prior to and after the quality filtering. For raw (unassembled) data these are equal to `total reads` and `good reads` respectively.

2.6.2 CdrBlast-manual

Description

A script to map V-(D)-J junctions, extract CDR3 sequences and assemble clonotypes.

Usage

General:

```
java -jar migec.jar CdrBlast [options] -R gene file1.fastq[.gz] [file2.fastq[.gz] ...
↪] output_file
```

Standard, assuming an example of a library containing T-cell Receptor Alpha Chain sequences

in case of MIG-assembled data:

```
java -jar migec.jar CdrBlast -a -R TRA assembly/S1_R2.fastq.gz cdrblast/S1_asm.
↪cdrblast.txt
```

for raw data:

```
java -jar migec.jar CdrBlast -R TRA checkout/S1_R2.fastq.gz cdrblast/S1_raw.cdrblast.
↪txt
```

to concatenate and process two or more FASTQ files at once:

```
java -jar migec.jar CdrBlast -R TRA checkout/S1_R2.fastq.gz checkout/S2_R2.fastq.gz␣
↪cdrblast/S12_raw.cdrblast.txt
```

Gene parameter `-R` is required, supported genes are *TRA*, *TRB*, *TRG*, *TRD*, *IGH*, *IGK* and *IGL*. Several chains can be specified, for example `-R TRA,TRB` or `-R IGH,IGL`. Species could be provided with `-S` parameter, by default uses *HomoSapiens*, supported species are *HomoSapiens*, *MusMusculus* and others. Assembled data should be passed to the script with `-a` option. `--same-sample` option should be used if several assembled files are provided from the same sample, so duplicate UMIs will be discarded and not counted twice.

To get a sorted output use `-o` option, otherwise sorting will be performed at **FilterCdrBlastResults** step. Note that both raw and assembled data should be processed to apply the last step of filtration.

Note: In order to use all alleles, not just the major (*01 ones), use the `--all-alleles` option. To include non-coding segments (V segment pseudogenes) use the `--all-segments` option.

2.7 Result filtering

2.7.1 FilterCdrBlastResults-batch

Description

A script to filter erroneous CDR3 sequences produced due to hot-spot PCR and NGS errors. It can also use a hybrid error correction method that includes frequency-based filtering of singleton clonotypes (i.e. clonotypes represeted by a single MIG).

Usage

```
java -jar migec.jar FilterCdrBlastResultsBatch [options] cdrblast_batch_folder/␣
↪output_folder/
```

Perform hot-spot error filtration for data process with **CdrBlastBatch**. Options are the same as for manual version below.

2.7.2 FilterCdrBlastResults-manual

Usage

General:

```
java -jar migec.jar FilterCdrBlastResults [options] cdrblast_result_assembled_data␣
↪cdrblast_result_raw_data output_file
```

Example:

```
java -jar migec.jar FilterCdrBlastResults cdrblast/S1_asm.cdrblast.txt cdrblast/S1_
↪raw.cdrblast.txt final/S1.cdrblast.txt
```

The `-s` option tells to filter CDR3s represented by single MIGs. The rationale for this is that the deep repertoire profiling (at least with our protocol) can generate spurious singletons that are associated with reverse transcription errors and experimental artifacts. Filtering is a non-greedy procedure and filters single-MIG clonotypes only if a 1- or 2-mismatch parent clonotype exists at ratio 1:20 and 1:400 respectively. This is done to preserve diversity for samples with shallow sequencing, e.g. ran on MiSeq.

Other options:

- `-n` - output non-coding clonotypes that contain either a stop codon or a frameshift within CDR3 sequence.
- `-c` - include non canonical clonotypes that have a CDR3 region that does not start with conserved C residue, or end with a conserved F/W residue.
- `-r` - sets the read accumulation threshold (default is `1.0`) used for hot-spot error correction, see MiGEC paper for details.

Now the file *S1.cdrblast.txt* contains a filtered and sorted CDR3/V/J clonotype table.

2.8 MIGEC log structure

Below is the description of log files produced by various MIGEC routines.

Checkout

De-multiplexing, barcode extraction and overlapping:

- `INPUT_FILE_1` first input file containing R1 reads
- `INPUT_FILE_2` second input file containing R2 reads
- `SAMPLE` sample name
- `MASTER` number of reads where primary (master) barcode was detected
- `SLAVE` number of reads where secondary (slave) barcode was detected
- `MASTER+SLAVE` number of reads where both barcodes were
- `OVERLAPPED` number of succesfully overlapped reads

Histogram

The routine produces a number of histograms for UMI coverage, i.e. statistics of the number of reads tagged with a given UMI:

- `overseq.txt` contains sample id and sample type (single/paired/overlapped) in the header, followed by UMI coverage (MIG size). Each row has total read counts for UMIs corresponding to a given UMI coverage
- `overseq-units.txt` same as `overseq.txt`, but lists numbers of unique UMIs, not total read counts
- `estimates.txt` contains sample id, sample type, total number of reads (`TOTAL_READS`) and UMIs (`TOTAL_MIGS`) in the sample and selected thresholds: `OVERSEQ_THRESHOLD` - UMI coverage threshold, `COLLISION_THRESHOLD` - if greater or equal to `OVERSEQ_THRESHOLD` will search for UMIs that differ by a single mismatch and have a huge count difference and treat them as being the same UMI, `UMI_QUAL_THRESHOLD` - threshold for min UMI sequence quality, `UMI_LEN` - UMI length
- `collision1.txt` - same as `overseq.txt`, but lists only UMIs that are likely to be erroneous (i.e. have a 1-mismatch UMI neighbour with a substantially higher count)
- `collision1-units.txt` - same as `collision1.txt`, but lists numbers of unique UMIs, not total read counts
- `pwm.txt` and `pwm-units.txt` - a position weight matrix (PWM) representation of all UMI sequences

Assemble

Statistics of MIG (group of reads tagged with the same UMI) consensus sequence assembly. Note that it also contains summary of pre-filtering steps, e.g. UMIs with low coverage are filtered at this stage:

- `SAMPLE_ID` sample name
- `SAMPLE_TYPE` sample type (single/paired/overlapped)
- `INPUT_FASTQ1` first input file containing R1 reads
- `INPUT_FASTQ2` second input file containing R2 reads
- `OUTPUT_ASSEMBLY1` first output file containing R1 consensuses
- `OUTPUT_ASSEMBLY2` second output file containing R2 consensuses
- `MIG_COUNT_THRESHOLD` UMI coverage threshold used in assemble procedure
- `MIGS_GOOD_FASTQ1` number of succesfully assembled consensuses from R1
- `MIGS_GOOD_FASTQ2` same for R2
- `MIGS_GOOD_TOTAL` number of succesfully assembled consensuses that have both R1 and R2 parts
- `MIGS_TOTAL` total number of input UMIs prior to coverage filtering
- `READS_GOOD_FASTQ1` number of reads in succesfully assembled consensuses from R1
- `READS_GOOD_FASTQ2` same for R2
- `READS_GOOD_TOTAL` number of paired reads in succesfully assembled consensuses that have both R1 and R2 parts. If a given assembled consensus contains inequal number of reads in R1 and R2, an average number is added to this statistic
- `READS_TOTAL` total number of input reads prior to coverage filtering
- `READS_DROPPED_WITHIN_MIG_1` number of reads dropped during consensus assembly as they had high number of mismatches to the consensus in R1
- `READS_DROPPED_WITHIN_MIG_2` same for R2
- `MIGS_DROPPED_OVERSEQ_1` number of UMIs dropped due to insufficient coverage in R1
- `MIGS_DROPPED_OVERSEQ_2` same for R2
- `READS_DROPPED_OVERSEQ_1` number of reads in UMIs dropped due to insufficient coverage in R1
- `READS_DROPPED_OVERSEQ_2` same for R2
- `MIGS_DROPPED_COLLISION_1` number of UMIs dropped due to being an erroneous (1-mismatch) variant of some UMI with higher count in R1
- `MIGS_DROPPED_COLLISION_2` same for R2
- `READS_DROPPED_COLLISION_1` number of reads in UMIs dropped due to being an erroneous (1-mismatch) variant of some UMI with higher count in R1
- `READS_DROPPED_COLLISION_2` same for R2

CdrBlast

Statistics of V(D)J mapping with BLAST algorithm:

- `SAMPLE_ID` sample name
- `DATA_TYPE` raw reads (raw) or assembled consensuses (asm)
- `OUTPUT_FILE` output file name

- INPUT_FILES list of input files
- EVENTS_GOOD number of MIGs (group of reads tagged with the same UMI, equals to number of reads for raw data) that were V(D)J mapped and passed the quality threshold
- EVENTS_MAPPED number of MIGs that were V(D)J mapped
- EVENTS_TOTAL number of input MIGs
- READS_GOOD number of reads that were V(D)J mapped and passed the quality threshold
- READS_MAPPED number of reads that were V(D)J mapped
- READS_TOTAL number of input reads

FilterCdrBlastResults

Statistics of the second round of TCR/Ig clonotype filtering that considers the number of supporting reads before and after consensus assembly:

- SAMPLE_ID sample name
- OUTPUT_FILE output file name
- INPUT_RAW input file containing CdrBlast results for raw reads
- INPUT_ASM input file containing CdrBlast results for assembled consensuses
- CLONOTYPES_FILTERED number of clonotypes (unique TCR/Ig V+CDR3 nucleotide+J combinations) that were filtered
- CLONOTYPES_TOTAL number of input clonotypes
- EVENTS_FILTERED number of MIGs in filtered clonotypes
- EVENTS_TOTAL number of input MIGs
- READS_FILTERED number of reads in filtered clonotypes
- READS_TOTAL number of input reads
- NON_FUNCTIONAL_CLONOTYPES number of non-functional clonotypes that contain stop codon/frameshift in CDR3
- NON_FUNCTIONAL_EVENTS number of MIGs in non-functional clonotypes
- NON_FUNCTIONAL_READS number of reads in non-functional clonotypes

2.9 MIGEC analysis report

You can run the following command to generate

```
java -jar migec.jar Report [options] [output_path or .]
```

Command parameters set the checkout (-c), histogram (-h), assemble (-a), cdrblast (-b) and final results (-f) folders. Missing results will be excluded from the report.

> **Warning:** Works for **batch** analysis only.

Note: Running this command requires installing dependencies for R markdown compilation, see http://rmarkdown.rstudio.com/

Alternatively, you can install Rstudio and use this Rmd template to manually knit the report HTML.

2.10 Post-processing

You can additionally build a graph of hypermutations for the sample using

```
java -jar MIGEC.jar CreateCdrHypermGraph final/S1.cdr3blast.txt net
```

which will generate files that allow fast network construction using Cytoscape's network from table and import table routines for further data exploration.

Note that translated CDR3 sequences are obtained by simultaneously translating codons in two directions: from V and J segments to the middle of CDR3. If a frameshift is detected, the incomplete codon is added in lower case, with missing nucleotides marked as `?`; stop codons are marked by `*`. CDR3 that contain either frameshift or stop codon are non-functional and are filtered by default. To include them into your output use `-n` option.

We also recommend to try out our new VDJtools software suite to perform post analysis of clonotype tables generated by MIGEC.