MiXCR v1.2 User Guide

1. Installation

1.1 System requirements

- Any Java-enabled platform (Windows, Linux, Mac OS X)
- Java version 7 or higher (download from Oracle web site)
- 1--16 Gb RAM (depending on number of clones in the sample)

1.2 Installation on Mac OS X / Linux

- check that you have Java 1.7+ installed on your system by typing `java -version`. Here is the example output of this command:

  ```
  java version "1.7.0_65"
  Java(TM) SE Runtime Environment (build 1.7.0_65-b17)
  Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
  ```

- unzip the archive with MiXCR

- add `bin` folder of MiXCR distribution to your `PATH` variable or add symbolic link to `mixcr` script to the user folder with executables (e.g. `~/bin/` in Ubuntu and many other popular linux distributions)

1.3 Installation on Windows

Currently there is no execution script or installer for Windows. Still MiXCR can easily be used by direct execution from the jar file.

- check that you have Java 1.7+ installed on your system by typing `java -version`. Here is the example output of this command:

  ```
  java version "1.7.0_65"
  Java(TM) SE Runtime Environment (build 1.7.0_65-b17)
  Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
  ```

- unzip the archive with MiXCR

- use `mixcr.jar` from the `jar` folder in the following way:

  ```
  java -Xmx4g -Xms3g -jar path_to_mixcr\jar\mixcr.jar ...
  ```

  For example:

  ```
  java -Xmx4g -Xms3g -jar C:\path_to_mixcr\jar\mixcr.jar align input.fastq.gz output.vdjca
  ```

To use mixcr from `jar` file one need to substitute `mixcr` command with `java -Xmx4g -Xms3g -jar path_to_mixcr\jar\mixcr.jar` in all examples from this manual.

2. Quick start

Typical workflow of MiXCR consists of three steps: alignment of raw sequencing reads with reference V, D, J genes, assembling of clones from aligned reads and exporting of necessary data columns for assembled clonotypes to a tab-delimited text file. There are many parameters that a user can change to adapt MiXCR for particular needs. While all these parameters are optional there is a set of parameters that are worth considering before running the analysis:

- `-OvParameters.geneFeatureToAlign` sets the gene feature of V gene used for alignment. Applied on the alignment stage. Choice of the value for this parameter depends on the type of starting material and library preparation strategy used. There are three options covering most of the cases (see Section 5 for the full list):

  - `VRegion` **(default)** is generally suitable for majority of use cases, on the other hand if you have some additional information about your library it is a good idea to use one of the values mentioned below instead of default. Don't change the default value if your library is prepared using multiplex PCR on the V gene side.
  - `VTranscript` if RNA was used as a starting material and some kind of non-template-specific technique was used for further amplification on the 5'-end of RNA (e.g. 5'RACE) (see Example E2). Using of this option is useful for increasing of sequencing information utilization from 5'-end of the molecule, which in turn helps to increase accuracy of V gene identification.
  - `VGene` if DNA was used as a starting material and 5' parts of V gene (including V intron, leader sequence and 5'UTR) are supposed to be present in your data. Using of this option is useful for increasing of sequencing information utilization from 5'-end of the molecule, which in turn helps to increase accuracy of V gene identification.

  Use one of the non-default values if you intend to analyse full-length sequence of T- or B- cell receptors.

- The `-OassemblingFeatures` parameter sets the region of TCR/BCR sequence which will be used to assemble clones. Applyed on the assembly stage. By default its value is `CDR3` which results in assembling of clones by the sequence of *Complementarity Determining Region 3*. To analyse full length sequences use `VDJRegion` as a value for the `assemblingFeatures` (see Section 5 for more details).

- Another important parameter is `--species`, it sets the organism from which the target sample was obtained. THis parameter is used on the alignment stage. Possible values are `hsa` (or `HomoSapiens`) and `mmu` (or `MusMusculus`). Default value is `hsa`. This parameter should be supplied on the alignment (`align`) stage. See Example E4.

The following sections describes common use cases

Examples

E1. Default workflow

MiXCR can be used with the default parameters in most cases by executing the following sequence of commands:

```
> mixcr align --loci IGH input_R1.fastq input_R2.fastq alignments.vdjca

... Building alignments

> mixcr assemble alignments.vdjca clones.clns

... Assembling clones

> mixcr exportClones clones.clns clones.txt

... Exporting clones to tab-delimited file
```

Here the only one parameter is set (`--loci IGH`) to tell MiXCR to search for IGH sequences, however even this parameter can be omitted (in this case MiXCR will search through all possible T-/B- cell receptor sequences: TRA, TRB, TRG, TRD, IGH, IGL, IGK). Omitting of `--loci` is not recommended.

The file produced (`clone.txt`) will contain a tab-delimited table with information about all clonotypes assembled by CDR3 sequence (clone abundance, CDR3 sequence, V, D, J genes, etc.). For full length analysis and other useful features see examples below.

This sequence of commnads is particularly suitable for analysis of **multiplex-PCR** selected fragments of T-/B- cell receptor genes.

E2. Analysis of data obtained using 5'RACE-based amplification protocols

Consider MiXCR workflow in more detail on analysis of paired-end sequenced cDNA library of IGH gene prepared using 5'RACE-based protocol (i.e. one read covers CDR3 with surroundings and another one covers 5'UTR and downstream sequence of V gene):

1. Align raw sequences to reference sequences of segments (V, D, J) of IGH gene:

```
> mixcr align --loci IGH -OvParameters.geneFeatureToAlign=VTranscript --report alignmentReport.log input_R1.fastq input_R2.fa
stq alignments.vdjca
```

Here we specified non-default value for gene feature used to align V genes (`-OvParameters.geneFeatureToAlign=VTranscript`) in order to utilize information from both reads, more specifically to let MiXCR align V gene's 5'UTRS and parts of coding sequence on 5'-end with sequence from read opposite to CDR3. MiXCR can also produce report file (specified by optional parameter `--report`) containing run statistics which looks like this:

```
Analysis Date: Mon Aug 25 15:22:39 MSK 2014
Input file(s): input_r1.fastq,input_r2.fastq
Output file: alignments.vdjca
Command line arguments: align --loci IGH --report alignmentReport.log input_r1.fastq input_r2.fastq alignments.vdjca
Total sequencing reads: 323248
Successfully aligned reads: 210360
Successfully aligned, percent: 65.08%
Alignment failed because of absence of V hits: 4.26%
Alignment failed because of absence of J hits: 30.19%
Alignment failed because of low total score: 0.48%
```

One can convert binary output produced by `align` (`output.vdjca`) to a human-readable text file using exportAlignments command.

2. Assemble clonotypes:

```
> mixcr assemble --report assembleReport.log alignments.vdjca clones.clns
```

This will build clonotypes and additionally correct PCR and sequencing errors. By default, clonotypes will be assembled by CDR3 sequences; one can specify another gene region by passing additional command line arguments (see assemble documentation). The optional report `assembleReport.log` will look like:

```
Analysis Date: Mon Aug 25 15:29:51 MSK 2014
Input file(s): alignments.vdjca
Output file: clones.clns
Command line arguments: assemble --report assembleReport.log alignments.vdjca clones.clns
Final clonotype count: 11195
Total reads used in clonotypes: 171029
Reads used, percent of total: 52.89%
Reads used as core, percent of used: 92.04%
Mapped low quality reads, percent of used: 7.96%
Reads clustered in PCR error correction, percent of used: 0.04%
Clonotypes eliminated by PCR error correction: 72
Percent of reads dropped due to the lack of clonal sequence: 2.34%
Percent of reads dropped due to low quality: 3.96%
Percent of reads dropped due to failed mapping: 5.87%
```

3. Export binary file with a list of clones (`clones.clns`) to a human-readable text file:

```
> mixcr exportClones clones.clns clones.txt
```

This will export information about clones with default set of fields, e.g.:

Clone count	Clone fraction	...	V hits	J hits	...	N. seq. CDR3	AA. seq. CDR3	...
4369	2.9E-3	...	IGHV4-39*00(1388)	IGHJ6*00(131)	...	TGTGTGAG...GACGTCTGG	CVRHKPMVQGGVDVW	...
3477	2.3E-3	...	IGHV4-34*00(1944)	IGHJ4*00(153)	...	TGTGCGAT...ATGACTTCTGG	CAIWDVGLRHDFW	...
...	...	...	...	...	...	...	...	...

where dots denote rows not shown here (for compactness). For the full list of available export options see export documentation.

Each of the above steps can be customized in order to adapt the analysis pipeline for a specific research task (see below).

E3. Full length IGH analysis

1. To build clonotypes based on the full-length sequence of variable part of IGH gene (not V gene only, but V-D-J junction with whole V Region and J Region) one need to obtain alignments fully covering V Region (like in E2). For example:

```
> mixcr align --loci IGH -OvParameters.geneFeatureToAlign=VTranscript input_R1.fastq input_R2.fastq alignments.vdjca
```

2. Then assemble clones with corresponding option (`-OassemblingFeatures=VDJRegion`):

```
> mixcr assemble -OassemblingFeatures=VDJRegion alignments.vdjca clones.clns
```

3. And export clones to a tab-delimited file:

```
> mixcr exportClones clones.clns clones.txt
```

Resulting file will contain assembled clonotypes with sequences of all regions (`CDR1` , `CDR2` , `CDR3` , `FR1` , `FR2` , `FR3` , `FR4`) for each clone.

E4. Assembling of CDR3-based clonotypes for mouse TRB sample

This example shows how to perform routine assembly of clonotypes (based on CDR3 sequence) for mouse TRB library (analysis for other genes can be performed by setting different value for the `--loci` parameter, or even omitting it to search for all possible genes - TRA/B/D/G and IGH/L/K).

```
> mixcr align --loci TRB --species mmu input_R1.fastq input_R2.fastq alignments.vdjca
```

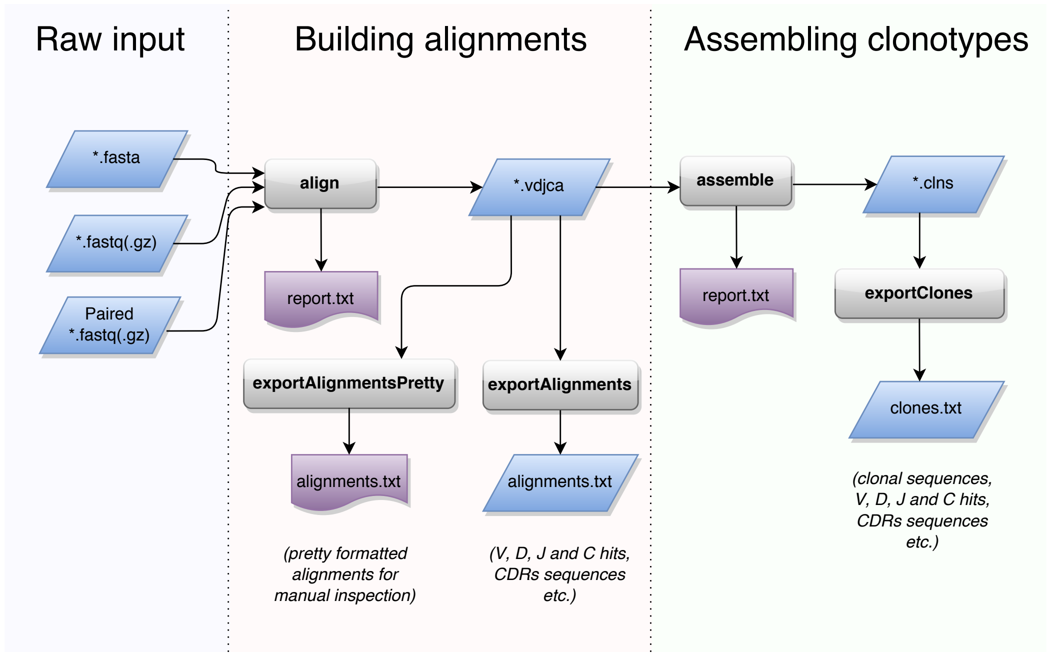
Other analysis stages can be executed without any additional parameters:

```
> mixcr assemble alignments.vdjca clones.clns

> mixcr exportClones clones.clns clones.txt
```

3. Overview

MiXCR is suitable for a wide range of research tasks in a field of adaptive immunity. This flexibility is achieved by making MiXCR modular. The following flowchart illustrates MiXCR workflow:



MiXCR workflow consists of three main processing steps:

- `align` : align sequencing reads to reference V, D, J and C genes of T- or B- cell receptors
- `assemble` : assemble clonotypes using alignments obtained on previous step (in order to extract specific gene regions e.g. CDR3)
- export: export alignment (`exportAlignments`) or clones (`exportClones`) to human-readable text file

MiXCR supports the following formats of sequencing data: `fasta` , `fastq` , `fastq.gz` , paired-end `fastq` and `fastq.gz` . As an output of each processing stage, MiXCR produces binary compressed file with comprehensive information about entries produced by this stage (alignments in case of `align` and clones in case of `assemble`). Each binary file can be converted to a human-readable/parsable tab-delimited text file using `exportAlignments` and `exportClones` commands.

The further sections covers each processing stage in detail.

4. MiXCR analysis steps

4.1 Align

The `align` command aligns raw sequencing reads to reference V, D, J and C genes of T- and B- cell receptors. It has the following syntax:

```
mixcr align [options] input_file1 [input_file2] output_file.vdjca
```

MiXCR supports `fasta` , `fastq` , `fastq.gz` and paired-end `fastq` and `fastq.gz` input. In case of paired-end reads two input files should be specified.

4.1.1 Command line parameters

The following table contains description of command line options for `align` :

Option	Default value	Description
`-h` , `--help`		Print help message.
`-r` , `--report`		Report file name. If this option is not specified, no report file be produced.
`-l` , `--loci`	`ALL`	Target immunological loci list separated by `,` . Available values: `IGH` , `IGL` , `IGK` , `TRA` , `TRB` , `TRG` , `TRD` , `IG` (for all immunoglobulin loci), `TCR` (for all T-cell receptor loci), `ALL` (for all loci) .
`-s` , `--species`	`HomoSapiens`	Species (organism). Possible values: `hsa` (or `HomoSapiens`) and `mmu` (or `MusMusculus`).
`-t` , `--threads`	number of available CPU cores	Number of processing threads.
`-n` , `--limit`		Limit number of sequences that will be analysed (only first `-n` sequences will be processed from input file(s)).
`-a` , `--save-description`		Copy read(s) description line from `.fastq` or `.fasta` to `.vdjca` file (can be then exported with `-descrR1` and `-descrR2` options in `exportAlignments` action).
`-Oparameter=value`		Overrides default value of aligner `parameter` (see next subsection).

All parameters are optional.

4.1.2 Aligner parameters

MiXCR uses a wide range of parameters that controls aligner behaviour. There are some global parameters and gene-specific parameters organized in groups: `vParameters` , `dParameters` , `jParameters` and `cParameters` . Each group of parameters may contain further subgroups of parameters etc. In order to override some parameter value one can use `-O` followed by fully qualified parameter name and parameter value (e.g. `-Ogroup1.group2.parameter=value`).

One of the key MiXCR features is ability to specify particular gene regions which will be extracted from reference and used as a targets for alignments. Thus, each sequencing read will be aligned to these extracted reference regions. The parameters responsible for target gene regions are:

Parameter	Default value	Description
`vParameters.geneFeatureToAlign`	`VRegion`	region in V gene which will be used as target in `align`
`dParameters.geneFeatureToAlign`	`DRegion`	region in D gene which will be used as target in `align`
`jParameters.geneFeatureToAlign`	`JRegion`	region in J gene which will be used as target in `align`
`cParameters.geneFeatureToAlign`	`CExon1`	region in C gene which will be used as target in `align`

It is important to specify these gene regions such that they will fully cover target clonal gene region which will be used in `assemble` (e.g. CDR3).

One can override default gene regions in the following way:

```
mixcr align -OvParameters.geneFeatureToAlign=VTranscript input_file1 [input_file2] output_file.vdjca
```

Other global aligner parameters are:

Parameter	Default value	Description
`minSumScore`	`120.0`	Minimal total alignment score value of V and J genes.
`maxHits`	`5`	Maximal number of hits for each gene type: if input sequence align to more than `maxHits` targets, then only top `maxHits` hits will be kept.
`relativeMinVFR3CDR3Score` (*only for paired-end analysis*)	`0.7`	Relative minimal alignment score of `FR3+VCDR3Part` region for V gene. V hit will be kept only if its `FR3+VCDR3Part` part aligns with score greater than `relativeMinVFR3CDR3Score * maxFR3CDR3Score` , where `maxFR3CDR3Score` is the maximal alignment score for `FR3+VCDR3Part` region among all of V hits for current input reads pair.
`readsLayout` (*only for paired-end analysis*)	`Opposite`	Relative orientation of paired reads. Available values: `Opposite` , `Collinear` , `Unknown` .

One can override these parameters in the following way:

```
mixcr align -OmaxHits=3 input_file1 [input_file2] output_file.vdjca
```

V, J and C aligners parameters

MiXCR uses same types of aligners to align V, J and C genes (`KAligner` from MiLib; the idea of `KAligner` is inspired by this article). These parameters are placed in `parameters` subgroup and can be overridden using e.g. `-OjParameters.parameters.mapperKValue=7` . The following parameters for V, J and C aligners are available:

Parameter	Default V value	Default J value	Default C value	Description
`mapperKValue`	`5`	`5`	`5`	Length of seeds used in aligner.
`floatingLeftBound`	`true`	`true`	`false`	Specifies whether left bound of alignment is fixed or float: if `floatingLeftBound` set to false, the left bound of either target or query will be aligned. Default values are suitable in most cases.
`floatingRightBound`	`true`	`true`	`false`	Specifies whether right bound of alignment is fixed or float: if `floatingRightBound` set to false, the right bound of either target or query will be aligned. Default values are suitable in most cases. If your target molecules have no primer sequences in J Region (e.g. library was amplified using primer to the C region) you can change value of this parameter for J gene to `false` to increase J gene identification accuracy and overall specificity of alignments.
`minAlignmentLength`	`15`	`15`	`15`	Minimal length of aligned region.
`maxAdjacentIndels`	`2`	`2`	`2`	Maximum number of indels between two seeds.
`absoluteMinScore`	`40.0`	`40.0`	`40.0`	Minimal score of alignment: alignments with smaller score will be dropped.
`relativeMinScore`	`0.87`	`0.87`	`0.87`	Minimal relative score of alignments: if alignment score is smaller than `relativeMinScore * maxScore`, where `maxScore` is the best score among all alignments for particular gene type (V, J or C) and input sequence, it will be dropped.
`maxHits`	`7`	`7`	`7`	Maximal number of hits: if input sequence align with more than `maxHits` queries, only top `maxHits` hits will be kept.

These parameters can be overridden like in the following example:

```
mixcr align -OvParameters.parameters.minAlignmentLength=30 \
            -OjParameters.parameters.relativeMinScore=0.7 \
            input_file1 [input_file2] output_file.vdjca
```

Scoring used in aligners is specified by `scoring` subgroup of parameters. It contains the following parameters:

Parameter	Default value	Description
`subsMatrix`	`simple(match = 5, mismatch = -9)`	Substitution matrix. Available types: - `simple` --- a matrix with diagonal elements equal to `match` and other elements equal to `mismatch` - `raw` --- a complete set of 16 matrix elements should be specified; for example: `raw(5,-9,-9,-9,-9,5,-9,-9,-9,-9,5,-9,-9,-9,-9,5)` (*equivalent to the default value*)
`gapPenalty`	`-12`	Penalty for gap.

Scoring parameters can be overridden in the following way:

```
mixcr align -OvParameters.parameters.scoring.gapPenalty=-20 input_file1 [input_file2] output_file.vdjca
```

```
mixcr align -OvParameters.parameters.scoring.subsMatrix=simple(match=4,mismatch=-11) \
             input_file1 [input_file2] output_file.vdjca
```

D aligner parameters

The following parameters can be overridden for D aligner:

Parameter	Default value	Description
`absoluteMinScore`	`30.0`	Minimal score of alignment: alignments with smaller scores will be dropped.
`relativeMinScore`	`0.85`	Minimal relative score of alignment: if alignment score is smaller than `relativeMinScore * maxScore`, where `maxScore` is the best score among all alignments for particular sequence, it will be dropped.
`maxHits`	`3`	Maximal number of hits: if input sequence align with more than `maxHits` queries, only top `maxHits` hits will be kept.

One can override these parameters like in the following example:

```
mixcr align -OdParameters.absoluteMinScore=10 input_file1 [input_file2] output_file.vdjca
```

Scoring parameters for D aligner are the following:

Parameter	Default value	Description
`type`	`affine`	Type of scoring. Possible values: `affine`, `linear`.
`subMatrix`	`simple(match = 5, mismatch = -9)`	Substitution matrix. Available types: - `simple` --- a matrix with diagonal elements equal to `match` and other elements equal to `mismatch` - `raw` --- a complete set of 16 matrix elements should be specified; for example: `raw(5,-9,-9,-9,-9,5,-9,-9,-9,-9,5,-9,-9,-9,-9,5)` (*equivalent to the default value*)
`gapOpenPenalty`	`-10`	Penalty for gap opening.
`gapExtensionPenalty`	`-1`	Penalty for gap extension.

These parameters can be overridden in the following way:

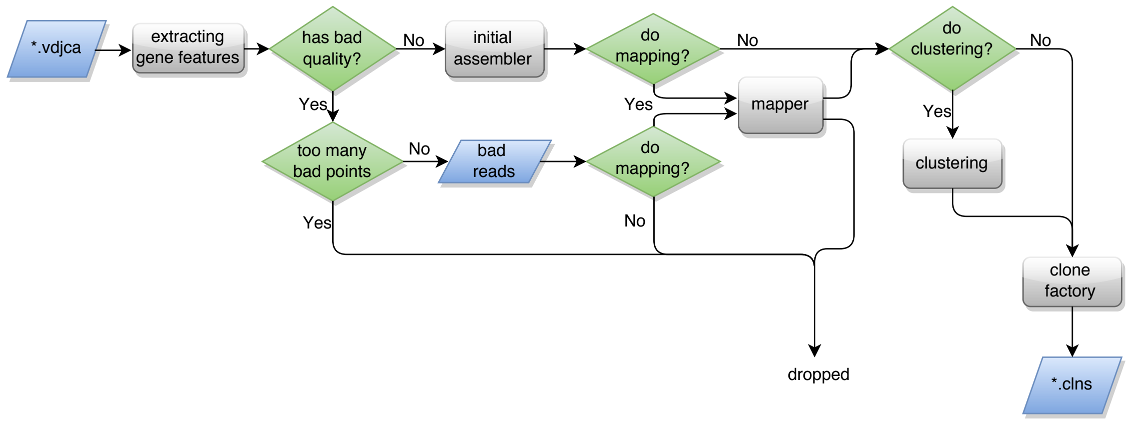
```
mixcr align -OdParameters.scoring.gapExtensionPenalty=-5 input_file1 [input_file2] output_file.vdjca
```

4.2 Assemble clones

The `assemble` command builds a set of clones using alignments obtained with [`align`] command in order to extract specific gene regions (e.g. CDR3). The syntax of `assemble` is the following:

```
mixcr assemble [options] alignments.vdjca output.clns
```

The following flowchart shows the pipeline of `assemble`:



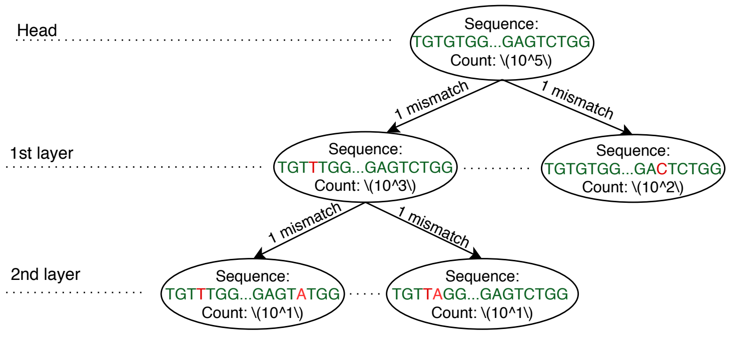
This pipeline consists of the following steps:

1. The assembler sequentially processes records (aligned reads) from input `.vdjca` file produced by [`align`]. On the first step, assembler tries to extract gene feature sequences from aligned reads (called *clonal sequence*) specified by `assemblingFeatures` parameter (`CDR3` by default); the clonotypes are assembled with respect to *clonal sequence*. If aligned read does not contain clonal sequence (e.g. `CDR3` region), it will be dropped.

2. If clonal sequence contains at least one nucleotide with low quality (less than `badQualityThreshold` parameter value), then this record will be deferred for further processing by *mapping procedure*. If percent of low quality nucleotides in deferred record is greater than `maxBadPointsPercent` parameter value, then this record will be finally dropped. Records with clonal sequence containing only good quality nucleotides are used to build core clonotypes by grouping records by equality of clonal sequences (e.g. CDR3). Each core clonotype has two main properties: clonal sequence and `count` --- a number of records aggregated by this clonotype.

3. After the core clonotypes are built, MiXCR runs *mapping procedure* that processes records deferred on the previous step. *Mapping* is aimed on rescuing of quantitative information from low quality reads. For this, each deferred record is mapped onto already assembled clonotypes: if there is a fuzzy match, then this record will be aggregated by the corresponding clonotype; in case of several matched clonotypes, a single one will be randomly chosen with weights equal to clonotype counts. If no matches found, the record will be finally dropped.

4. After clonotypes are assembled by initial assembler and mapper, MiXCR proceeds to *clustering*. The clustering algorithm tries to find fuzzy matches between clonotypes and organize matched clonotypes in hierarchical tree (*cluster*), where each child layer is highly similar to its parent but has significantly smaller `count`. Thus, clonotypes with small counts will be attached to highly similar "parent" clonotypes with significantly greater count. The typical cluster looks as follows:

   After all clusters are built, only their heads are considered as final clones. The maximal depths of cluster, fuzzy matching criteria, relative counts of parent/childs and other parameters can be customized using `clusteringStrategy` parameters described below.

5. The final step is to align clonal sequences to reference V,D,J and C genes. Since the `assemblingFeatures` are different from those used in `align`, it is necessary to rebuild alignments for clonal sequences. This alignments are built by more accurate aligner (since all hits are known in advance); thus, better alignments will be built for each clonal sequence.

6. The result is written to the binary output file (`.clns`) with a comprehensive information about clones.

4.2.1 Command line parameters

The command line options of `assemble` are the following:

Option	Default value	Description
`-h` , `--help`		Print help message.
`-r` , `--report`		Report file name. If this option is not specified, no report file be produced.
`-t` , `--threads`	number of available CPU cores	Number of processing threads.
`-Oparameter=value`		Overrides default value of assembler `parameter` (see next subsection).

All parameters are optional.

4.2.2 Assembler parameters

MiXCR uses a wide range of parameters that controls assembler behaviour. There are some global parameters and parameters organized in groups for each stage of assembling: `cloneClusteringParameters` and `cloneFactoryParameters` . Each group of parameters may contain further subgroups of parameters etc. In order to override some parameter value one can use `-O` followed by fully qualified parameter name and parameter value (e.g. `-Ogroup1.group2.parameter=value`).

One of the key MiXCR features is ability to assemble clonotypes by sequence of custom gene region (e.g. `FR3+CDR3`); target clonal sequence can even be disjoint. This region can be specified by `assemblingFeatures` parameter, as in the following example:

```
mixcr assemble -OassemblingFeatures="[V5UTR+L1+L2+FR1,FR3+CDR3]" alignments.vdjca output.clns
```

(**note**: `assemblingFeatures` must cover `CDR3`).

Other global parameters are:

Parameter	Default value	Description
`badQualityThreshold`	`20`	Minimal value of sequencing quality score: nucleotides with lower quality are considered as "bad". If sequence contains at least one "bad" nucleotide, it will be deferred at initial assembling stage, for further processing by mapper.
`maxBadPointsPercent`	`0.7`	Maximal allowed percent of "bad" points in sequence: if sequence contains more than `maxBadPointsPercent` "bad" nucleotides, it will be dropped.
`addReadsCountOnClustering`	`false`	Aggregate cluster counts when assembling final clones: if `addReadsCountOnClustering` is `true` , then all children clone counts will be added to the head clone; thus head clone count will be a total of its initial count and counts of all its children.

One can override these parameters in the following way:

```
mixcr assemble -ObadQualityThreshold=10 alignments.vdjca output.clns
```

In order to prevent mapping of low quality reads (filter them off) one can set `maxBadPointsPercent` to zero:

```
mixcr assemble -OmaxBadPointsPercent=0 alignments.vdjca output.clns
```

Clustering strategy

Parameters that control clustering procedure are placed in `cloneClusteringParameters` parameters group:

Parameter	Default value	Description
`searchDepth`	`2`	Maximum number of cluster layers (not including head).
`allowedMutationsInNRegions`	`1`	Maximum allowed number of mutations in N regions (non-template nucleotides in VD, DJ or VJ junctions): if two fuzzy matched clonal sequences will contain more than `allowedMutationsInNRegions` mismatches in N-regions, they will not be clustered together (one cannot be a direct child of another).
`searchParameters`	`oneMismatch`	Parameters that control fuzzy match criteria between clones in adjacent layers. Available predefined values: `oneMismatch` , `oneIndel` , `oneMismatchOrIndel` . By default, `oneMismatch` allows only one mismatch between two adjacent clones (parent and direct child).
`clusteringFilter` `.specificMutationProbability`	`1E-3`	Probability of a single nucleotide mutation in clonal sequence which has non-hypermutation origin (i.e. PCR or sequencing error). This parameter controls relative counts between two clones in adjacent layers: a smaller clone can be attached to a larger one if its count smaller than count of parent multiplied by `(clonalSequenceLength * specificMutationProbability) ^ numberOfMutations` .

One can override these parameters in the following way:

```
mixcr assemble -OcloneClusteringParameters.searchParameters=oneMismatchOrIndel alignments.vdjca output.clns
```

In order to turn off clustering one should use the following parameters:

```
mixcr assemble -OcloneClusteringParameters=null alignments.vdjca output.clns
```

Clone factory parameters

Parameters which control final alignment of clonal sequences are placed in `cloneFactoryParameters` group. These parameters includes separate groups for V, D, J and C aligners: `vParameters` , `dParameters` , `jParameters` and `cParameters` . The D aligner is the same as used in `align` and thus all its parameters and their default values are the same as described for D aligner in `align` . One can override these parameters in the following way:

```
mixcr assemble -OcloneFactoryParameters.dParameters.absoluteMinScore=10 alignments.vdjca output.clns
```

```
mixcr assemble -OcloneFactoryParameters.dParameters.scoring.gapOpenPenalty=-10 alignments.vdjca output.clns
```

The aligners used to build alignments with V, J and C genes are different from those used by `align` .

V, J and C aligner parameters

The following table lists parameters of V, J and C aligners:

Parameter	Default V value	Default J value	Default C value	Description
`featureToAlign`	`VTranscript`	`JRegion`	`CExon1`	Gene region used to build alignments.
`relativeMinScore`	`0.8`	`0.8`	`0.8`	Relative minimal score of hit: hits with score less than `relativeMinScore * maxScore` (`maxScore` is score of best hit) will be dropped.

One can override these parameters in the following way

```
mixcr assemble -OcloneFactoryParameters.jParameters.featureToAlign=JRegion(-6,0) alignments.vdjca output.clns
```

The scoring parameters are placed in group `alignmentParameters.scoring` :

Parameter	Default value (same for V, J, C)	Description
`subsMatrix`	`simple(match = 5, mismatch = -9)`	Substitution matrix. Available types: - `simple` --- a matrix with diagonal elements equal to `match` and other elements equal to `mismatch` - `raw` --- a complete set of 16 matrix elements should be specified; for example: `raw(5,-9,-9,-9,-9,5,-9,-9,-9,-9,5,-9,-9,-9,-9,5)` (*equivalent to the default value*)
`gapPenalty`	`-12`	Penalty for gap.

One can override these parameters in the following way

```
mixcr assemble -OcloneFactoryParameters.vParameters.alignmentParameters.scoring.gapPenalty=-5 \
               alignments.vdjca output.clns
```

4.3 Export

In order to export result of alignment or clones from binary file (`.vdjca` or `.clns`) to a human-readable text file one can use `exportAlignments` and `exportClones` commands respectively. The syntax for these commands is:

```
mixcr exportAlignments [options] alignments.vdjca alignments.txt
```

```
mixcr exportClones [options] clones.clns clones.txt
```

The resulting tab-delimited text file will contain columns with different types of information. If no options specified, the default set of columns, which is sufficient in most cases, will be exported. The possible columns are (see below for details): aligned sequences, qualities, all or just best hit for V, D, J and C genes, corresponding alignemtns, nucleotide and amino acid sequences of gene region present in sequence etc. In case of clones, the additional columns are: clone count, clone fraction etc.

One can customize the list of fields that will be exported by passing parameters to export commands. For example, in order to export just clone count, best hits for V and J genes with corresponding alignments and `CDR3` amino acid sequence, one can do:

```
mixcr exportClones -count -vHit -jHit -vAlignment -jAlignment -aaFeature CDR3 clones.clns clones.txt
```

The columns in the resulting file will be exported in the exact same order as parameters in the command line. The list of available fields will be reviewed in the next subsections. For convenience, MiXCR provides two predefined sets of fields for exporting: `min` (will export minimal required information about clones or alignments) and `full` (used by default); one can use these sets by specifying `--preset` option:

```
mixcr exportClones --preset min clones.clns clones.txt
```

One can add additional columns to preset in the following way:

```
mixcr exportClones --preset min -qFeature CDR2 clones.clns clones.txt
```

One can also put all export fields in the file like:

```
-vHits
-dHits
-feature CDR3
...
```

and pass this file to export command:

```
mixcr exportClones --presetFile myFields.txt clones.clns clones.txt
```

4.3.1 Command line parameters

The list of command line parameters for both `exportAlignments` and `exportClones` is the following:

Option	Description
-h, --help	print help message
-f, --fields	list available fields that can be exported
-p, --preset	select predefined set of fields to export (`full` or `min`)
-pf, --presetFile	load file with a list of fields to export

4.3.2 Available fields

The following fields can be exported both for alignments and clones:

Field	Description
-vHit	Best V hit.
-dHit	Best D hit.
-jHit	Best J hit.
-cHit	Best C hit.
-vHits	All V hits.
-dHits	All D hits.
-jHits	All J hits.
-cHits	All C hits.
-vAlignment	Best V alignment.
-dAlignment	Best D alignment.
-jAlignment	Best J alignment.
-cAlignment	Best C alignment.
-vAlignments	All V alignments.
-dAlignments	All D alignments.
-jAlignments	All J alignments.
-cAlignments	All C alignments.
-nFeature [feature]	Nucleotide sequence of specified gene feature.
-qFeature [feature]	Quality of sequences of specified gene feature.
-aaFeature [feature]	Amino acid sequence of specified gene feature.
-avrgFeatureQuality [feature]	Average quality of sequence of specified gene feature.
-minFeatureQuality [feature]	Minimal quality of sequence of specified gene feature.

The following fields are specific for alignments:

Field	Description
-sequence	Aligned sequence (initial read), or 2 sequences in case of paired-end reads.
-quality	Initial read quality, or 2 qualities in case of paired-end reads.
-readId	Index of source read (in e.g. `.fastq` file) for alignment.
-targets	Number of targets, i.e. 1 in case of single reads and 2 in case of paired-end reads.
-descrR1	Description line from initial `.fasta` or `.fastq` file of the first read (only available if `--save-description` was used in `align` command).
-descrR2	Description line from initial `.fastq` file of the second read (only available if `--save-description` was used in `align` command).

The following fields are specific for clones:

Field	Description
-count	Clone count.
-fraction	Clone fraction.
-sequence	Clonal sequence (or several sequences in case of multi-featured assembling).
-quality	Clonal sequence quality (or several qualities in case of multi-featured assembling).
-targets	Number of targets, i.e. number of gene regions used to assemble clones.

4.3.3 Examples

Export only best V, D, J hits and best V hit alignment from `.vdjca` file:

```
mixcr exportAlignments -vHit -dHit -jHit -vAlignment input.vdjca test.txt
```

Best V hit	Best D hit	Best J hit	Best V alignment
IGHV4-34*00		IGHJ4*00	262|452|453|47|237|SC268GSC271ASC275G|956.1, 58|303|450|56|301|SG72TSA73CSG136TSA144CSA158CSG171T|331.0
IGHV2-23*00	IGHD2*21	IGHJ6*00	262|452|453|47|237|SC268GSC271ASC275G|956.1, 58|303|450|56|301|SG72TSA73CSG136TSA144CSA158CSG171T|331.0

The syntax of alignment is described in appendix.

4.4 Exporting well formatted alignments for manual inspection

MiXCR allows to export resulting alignments after align step as a pretty formatted text for manual analysis of produced alignments and structure of library to facilitate optimization of analysis parameters and libraray preparation protocol. To export pretty formatted alignments use `exportAlignmebtsPretty` command:

```
mixcr exportAlignmentsPretty --skip 1000 --limit 10 input.vdjca test.txt
```

this will export 10 results after skipping first 1000 records and place result into `test.txt` file. Skipping of first records is often useful because first sequences in fastq file may have lower quality then average reads, so first resulsts are not representative. It is possible to omit last paramenter with output file name to print result directly to standard output stream (to console), like this:

```
mixcr exportAlignmentsPretty --skip 1000 --limit 10 input.vdjca
```

Here is a summary of command line options:

Option	Description
-h, --help	print help message
-n, --limit	limit number of alignments; no more than provided number of results will be outputted
-s, --skip	number of results to skip
-t, --top	output only top hits for V, D, J nad C genes

Results produced by this command has the following structure:

```
>>> Read id: 12343    <--- Index of analysed read in input file

>>> Target sequences (input sequences):

Sequence0:   <--- Read 1 from paired-end read
Contains features: CDR1, VRegionTrimmed, L2, L, Intron, VLIntronL, FR1, Exon1,              <--- Gene features
VExon2Trimmed                                                                                    found in read 1

     0 TCTTGGGGGATTCGGTGATCAGCACTGAACACAGAGGACTCACCATGGAGTTTGGGCTGAACTGGGTTTTCCTCGTTGCT 79  <--- Sequyence & quality
       FGGEGGGGGDG8F78CFC6CEFF<,CFG9EED,6,CFCC<EEGFG,CE:CCAFFGGC87CEF?A?FBC@FGGFG>B,FC9          of read 1

    80 CTATTAAGAGGTGTCCAGTGTCAGGTGCAGCTGGTGGAGTCTGGGGGTGGCGTGTTCCAGCCTGGGGGGTCCGTGAGACT 159
       F9,A,95AFE,B?,E,C,9AC<FGA<EE5??,A,A<:=:E,=B8C7+++8,++@+,885=D7:@8E+:5*1**11**++<
   160 CTCCTGTGCAGCGTCGGGATGCACATCATGGAGCTATGGCCAGCCCTGGGTACGCCAGGCTACAGGCCACGGGCTGGAGG 239
       <++*++0++2A:ECE5EC5**2@C+:++++++22*2:+29+*2***25/79*0299))*/)*0*0*.75)7:)1)1/)))

   240 GGGTGCGTGGTAGATGGGAA 259
       )9:.)))*1)12***-/).)

Sequence1:   <--- Read 2 from paired-end read
Contains features: JCDR3Part, DCDR3Part, DJJunction, CDR2, JRegionTrimmed, CDR3, VDJunction,
VJJunction, VCDR3Part, ShortCDR3, FR4, FR3

     0 CGAGGCAAGAGGCTGGTGTGGGTGGCGGTTATATGGTATGGTGGAAGTAATAAACACTATGCAGACCCCGTGAAGGGCCG 79
       **0*0**)2**/**5D7<15*9<5:1+*0:GF:=C>6A52++*:2+++FF>>3<++++++302**:**/<+**;:/**2+

    80 ATTCACCATCGCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGAAGAGCCTGAGAGCCGAGGACACGGCTTTGT 159
       +++<0***C:2+9GGFB?,5,4,+,2F<>FC=*,,C:>,=,@,,;3<@=,3,,<3,CF?=**<>@,?3,<<:3,CC,E,@

   160 ATTACTGTGCGAGAGGTCAACAGGGTGACTATGTCTACGGTAGGGACGTCGGGGGCCAAGGGACCACGGTCACCGTCTCC 239
       ,@;FCF@+F@FGGF9FD,F>>+B:=,,=><GFCGGCFEGFF?+=B+7EF>+FFA,8F<E:,5+GDFFE,@F?,,7GGDFE

   240 TCAGGGAGTGCATCCGCCCCAACCCTTTTCCCCCTCTCTGCGTTGATACCACTGGCAGCTC 300
       C,FGGGEFCCGEEGGCFCC:8FGEGGGE@DFB-GFGGGGF@GFGFE<,GFCCFCAGC@CCC

>>> Gene features that can be extracted from this (paired-)read:                       <--- For paired-end reads
JCDR3Part, CDR1, VRegionTrimmed, L2, DCDR3Part, VDJTranscriptWithout5UTR, Exon2, L,          some gene features
DJJunction, Intron, FR2, CDR2, VDJRegion, JRegionTrimmed, CDR3, VDJunction, VJJunction,      can be extracted by
VLIntronL, FR1, VCDR3Part, ShortCDR3, Exon1, FR4, VExon2Trimmed, FR3                         merging sequence
                                                                                             information

>>> Alignments with V gene:

IGHV3-33*00 (total score = 1638.0) <--- Alignment of both reads with IGHV3-33
Alignment of Sequence0 (score = 899.0):   <--- Alignment of IGHV3-33 with read 1 from paired-end read
     65 ATTCGGTGATCAGCACTGAACACAGAGGACTCACCATGGAGTTTGGGCTGAGCTGGGTTTTCCTCGTTGCTCTTTTAAGA 144 <--- Germline
        ||||||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||||||||| ||||||
      9 ATTCGGTGATCAGCACTGAACACAGAGGACTCACCATGGAGTTTGGGCTGAACTGGGTTTTCCTCGTTGCTCTATTAAGA 88  <--- Read
        DG8F78CFC6CEFF<,CFG9EED,6,CFCC<EEGFG,CE:CCAFFGGC87CEF?A?FBC@FGGFG>B,FC9F9,A,95AF     <--- Quality score

    145 GGTGTCCAGTGTCAGGTGCAGCTGGTGGAGTCTGGGGGAGGCGTGGTCCAGCCTGGGAGGTCCCTGAGACTCTCCTGTGC 224
        |||||||||||||||||||||||||||||||||||||| |||||| ||||||||||| ||||| ||||||||||||||||
     89 GGTGTCCAGTGTCAGGTGCAGCTGGTGGAGTCTGGGGGTGGCGTGTTCCAGCCTGGGGGGTCCGTGAGACTCTCCTGTGC 168
```

```
        E,B?,E,C,9AC<FGA<EE5??,A,A<:=:E,=B8C7+++8,++@+,885=D7:@8E+:5*1**11****++<<++*++0++


    225 AGCGTCTGGATTCACCTTCA-GTAGCTATGGCATGCACTGGGTCCGCCAGGCTCCAGGCAAGGGGCTGGAGTGGGTG 300
        |||||| |||| || | ||| | |||||||||  || |||||| ||||||||| ||||| | ||||||||| |||||
    169 AGCGTCGGGATGCA-CATCATGGAGCTATGGCCAGCCCTGGGTACGCCAGGCTACAGGCCACGGGCTGGAGGGGGTG 244
        2A:ECE5EC5**2@ C+:++++++22*2:+29+*2***25/79*0299))*/)*0*0*.75)7:)1)1/))))9:.)

Alignment of Sequence1 (score = 739.0):   <--- Alignment of IGHV3-33 with read 2 from paired-end read
    279 AGGCAAGGGGCTGGAGTGGGTGGCAGTTATATGGTATGATGGAAGTAATAAATACTATGCAGACTCCGTGAAGGGCCGAT 358
        ||||||| |||||| ||||||||| ||||||||||||| ||||||||||||| ||||||||||| |||||||||||||||
      2 AGGCAAGAGGCTGGTGTGGGTGGCGGTTATATGGTATGGTGGAAGTAATAAACACTATGCAGACCCCGTGAAGGGCCGAT 81
        0*0**)2**/**5D7<15*9<5:1+*0:GF:=C>6A52++*:2+++FF>>3<++++++302**:**/<+**;:/**2+++


    359 TCACCATCTCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGAACAGCCTGAGAGCCGAGGACACGGCTGTGTAT 438
        |||||||| |||||||||||||||||||||||||||||||||||||||| |||||||||||||||||||||||| |||||
     82 TCACCATCGCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGAAGAGCCTGAGAGCCGAGGACACGGCTTTGTAT 161
        +<0***C:2+9GGFB?,5,4,+,2F<>FC=*,,C:>,=,@,,;3<@=,3,,<3,CF?=**<>@,?3,<<:3,CC,E,@,@


    439 TACTGTGCGAGAG 451
        |||||||||||||
    162 TACTGTGCGAGAG 174
        ;FCF@+F@FGGF9

IGHV3-30*00 (total score = 1582.0)  <--- Alternative hit for V gene
Alignment of Sequence0 (score = 885.0):
     65 ATTCGGTGATCAGCACTGAACACAGAGGACTCACCATGGAGTTTGGGCTGAGCTGGGTTTTCCTCGTTGCTCTTTTAAGA 144
        ||||||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||||||||||| ||||||
      9 ATTCGGTGATCAGCACTGAACACAGAGGACTCACCATGGAGTTTGGGCTGAACTGGGTTTTCCTCGTTGCTCTATTAAGA 88
        DG8F78CFC6CEFF<,CFG9EED,6,CFCC<EEGFG,CE:CCAFFGGC87CEF?A?FBC@FGGFG>B,FC9F9,A,95AF


    145 GGTGTCCAGTGTCAGGTGCAGCTGGTGGAGTCTGGGGGAGGCGTGGTCCAGCCTGGGAGGTCCCTGAGACTCTCCTGTGC 224
        |||||||||||||||||||||||||||||||||||||| |||||| ||||||||||| ||||| ||||||||||||||||
     89 GGTGTCCAGTGTCAGGTGCAGCTGGTGGAGTCTGGGGGTGGCGTGTTCCAGCCTGGGGGGTCCGTGAGACTCTCCTGTGC 168
        E,B?,E,C,9AC<FGA<EE5??,A,A<:=:E,=B8C7+++8,++@+,885=D7:@8E+:5*1**11****++<<++*++0++


    225 AGCCTCTGGATTCACCTTCA-GTAGCTATGGCATGCACTGGGTCCGCCAGGCTCCAGGCAAGGGGCTGGAGTGGGTG 300
        ||| || |||| || | ||| | |||||||||  || |||||| ||||||||| ||||| | ||||||||| |||||
    169 AGCGTCGGGATGCA-CATCATGGAGCTATGGCCAGCCCTGGGTACGCCAGGCTACAGGCCACGGGCTGGAGGGGGTG 244
        2A:ECE5EC5**2@ C+:++++++22*2:+29+*2***25/79*0299))*/)*0*0*.75)7:)1)1/))))9:.)

Alignment of Sequence1 (score = 697.0):
    279 AGGCAAGGGGCTGGAGTGGGTGGCAGTTATATCATATGATGGAAGTAATAAATACTATGCAGACTCCGTGAAGGGCCGAT 358
        ||||||| |||||| ||||||||| |||||||  |||| ||||||||||||| ||||||||||| |||||||||||||||
      2 AGGCAAGAGGCTGGTGTGGGTGGCGGTTATATGGTATGGTGGAAGTAATAAACACTATGCAGACCCCGTGAAGGGCCGAT 81
        0*0**)2**/**5D7<15*9<5:1+*0:GF:=C>6A52++*:2+++FF>>3<++++++302**:**/<+**;:/**2+++


    359 TCACCATCTCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGAACAGCCTGAGAGCTGAGGACACGGCTGTGTAT 438
        |||||||| |||||||||||||||||||||||||||||||||||||||| ||||||||||| |||||||||||| |||||
     82 TCACCATCGCCAGAGACAATTCCAAGAACACGCTGTATCTGCAAATGAAGAGCCTGAGAGCCGAGGACACGGCTTTGTAT 161
        +<0***C:2+9GGFB?,5,4,+,2F<>FC=*,,C:>,=,@,,;3<@=,3,,<3,CF?=**<>@,?3,<<:3,CC,E,@,@


    439 TACTGTGCGAGAG 451
        |||||||||||||
    162 TACTGTGCGAGAG 174
        ;FCF@+F@FGGF9

>>> Alignments with D gene:

IGHD4-17*00 (total score = 40.0)
Alignment of Sequence1 (score = 40.0):
      7 GGTGACTA 14
        ||||||||
    183 GGTGACTA 190
        :=,,=><G
```

```
IGHD4-23*00 (total score = 36.0)
Alignment of Sequence1 (score = 36.0):
      0 TGACTACGGT 9
        || |||||||
    191 TGTCTACGGT 200
        FCGGCFEGFF

IGHD2-21*00 (total score = 35.0)
Alignment of Sequence1 (score = 35.0):
     13 GGTGACT 19
        |||||||
    183 GGTGACT 189
        :=,,=><

>>> Alignments with J gene:

IGHJ6*00 (total score = 172.0)
Alignment of Sequence1 (score = 172.0):
     22 GGACGTCTGGGGCAAAGGGACCACGGTCACCGTCTCCTCA 61
        ||||||| ||||| ||||||||||||||||||||||||||
    203 GGACGTCGGGGGCCAAGGGACCACGGTCACCGTCTCCTCA 242
        =B+7EF>+FFA,8F<E:,5+GDFFE,@F?,,7GGDFEC,F

>>> Alignments with C gene:

No hits.
```

5. Gene features and anchor points

There are several immunologically important parts of TCR/BCR gene (**gene features**). For example, such regions are three complementarity determining regions (`CDR1` , `CDR2` and `CDR3`), four framework regions (`FR1` , `FR2` , `FR3` and `FR4`) etc.

The key feature of MiXCR is the possibility to specify:

- regions of reference V, D, J and C genes sequences that are used in alignment of raw reads
- regions of sequence to be exported by `exportAlignments`
- regions of sequence to use as clonal sequence in clone assembly
- regions of clonal sequences to be exported by `exportClones`

For convenience, in MiXCR these regions can be specified in terms of above mentioned immunological gene features. The illustrated list of predefined gene features can be found below. The set of possible gene regions is not limited by this list:

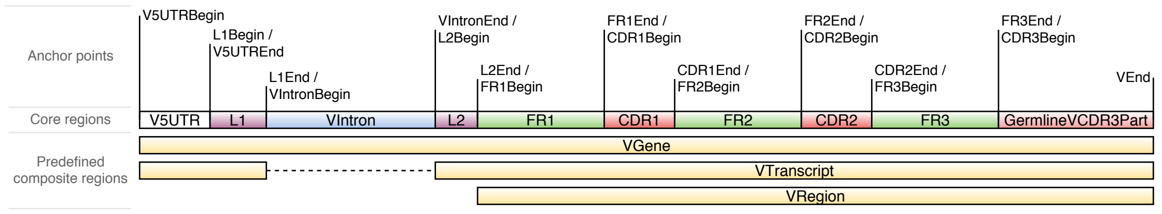
- boundary points of gene features (called **anchor points**) can be used to specify begin and end of custom gene regions
- gene features can be concatenated (e.g. `VTranscript = {V5UTRBegin:L1End}+{L2Begin:VEnd}`).
- offsets can be added or subtracted from original positions of **anchor points** to define even more custom gene regions
  (*for more detailed description see gene feature syntax*)

Naming of gene features is based on IMGT convention described in *Lefranc et al. (2003)*[1].

5.1 Germline features

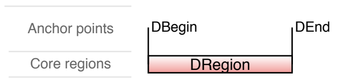
Features defined for germline genes are mainly used in `align` and export.

V Gene structure

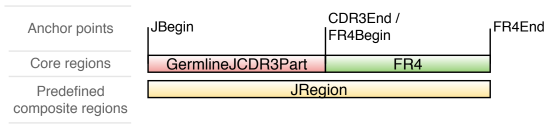


Additionally to core gene features in V region (like `FR3`) we introduce `VGene`, `VTranscript` and `VRegion` for convenience.

D Gene structure



J Gene structure

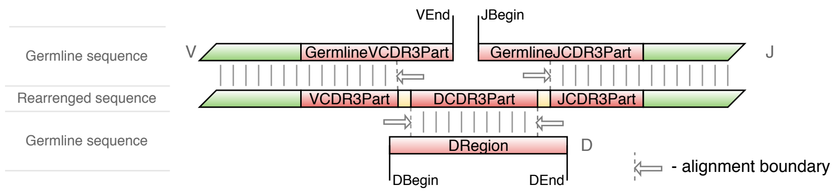


5.2 Mature TCR/BCR gene features

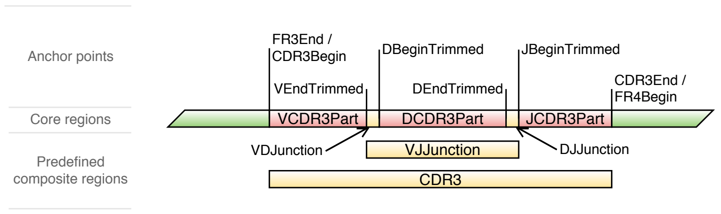
Features described here (like `CDR3`) cannot not be used for `align`, since they are not defined for germline genes.

V(D)J junction structure

Important difference between rearranged TCR/BCR sequence and germline sequence of its segments lies in the fact that during V(D)J recombination exact cleavage positions at the end of V gene, begin and end of D gene and begin of J gene varies. As a result in most cases actual `VEnd`, `DBegin`, `DEnd` and `JBegin` anchor positions are not covered by alignment:



In order to use actual V, D, J gene boundaries we introduce four additional anchor positions: `VEndTrimmed`, `DBeginTrimmed`, `DEndTrimmed` and `JBeginTrimmed` and several named gene features: `VDJunction`, `DJJunction` and `VJJunction`. On the following picture one can see the structure of V(D)J junction:



If D gene is not found in the sequence or is not present in target locus (e.g. TRA), `DBeginTrimmed` and `DEndTrimmed` anchor points as well as `VDJunction` and `DJJunction` gene features are not defined.

5.3 Gene feature syntax

Syntax for gene features is the same everywhere. The best way to explain it is by example:

- to enter any gene feature mentioned above or listed in the next section just use its name: `VTranscript`, `CDR2`, `V5UTR` etc.
- to define a gene feature consisting of several concatenated features use `+`: `V5UTR+L1+L2+VRegion` is equivalent to `VTranscript`
- to create gene feature starting at anchor point `X` and ending at anchor point `Y` use `{X:Y}` syntax: `{CDR3Begin:CDR3End}` for `CDR3`.
- one can add or subtract offset from original position of anchor point using positive or negative integer value in brackets after anchor point name `AnchorPoint(offset)`: `{CDR3Begin(+3):CDR3End}` for `CDR3` without first three nucleotides (coding conserved cysteine), `{CDR3Begin(-6):CDR3End(+6)}` for `CDR3` with 6 nucleotides downstream its left bound and 6 nucleotides upstream its right bound.
- one can specify offsets for predefined gene feature boundaries using `GeneFeatureName(leftOffset, rightOffset)` syntax: `CDR3(3,0)`, `CDR3(-6,6)` - equivalents of two examples from previous item
- all syntax constructs can be combined: `{L1Begin(-12):L1End}+L2+VRegion(0,+10)}`.

5.4 List of predefined gene features

Gene Feature Name	Gene feature decomposition
VGene	{UTR5Begin:VEnd}
VDJTranscript	{UTR5Begin:L1End}+{L2Begin:FR4End}
V5UTR	{UTR5Begin:UTR5End}
VTranscript	{UTR5Begin:L1End}+{L2Begin:VEnd}
Exon1	{L1Begin:L1End}
L	{L1Begin:L1End}+{L2Begin:L2End}
VTranscriptWithout5UTR	{L1Begin:L1End}+{L2Begin:VEnd}
VLIntronL	{L1Begin:L2End}
VDJTranscriptWithout5UTR	{L1Begin:L1End}+{L2Begin:FR4End}
Intron	{VIntronBegin:VIntronEnd}
VExon2	{L2Begin:VEnd}
Exon2	{L2Begin:FR4End}
L2	{L2Begin:L2End}
VExon2Trimmed	{L2Begin:VEndTrimmed}
FR1	{FR1Begin:FR1End}
VRegionTrimmed	{FR1Begin:VEndTrimmed}
VRegion	{FR1Begin:VEnd}
VDJRegion	{FR1Begin:FR4End}
CDR1	{CDR1Begin:CDR1End}
FR2	{FR2Begin:FR2End}
CDR2	{CDR2Begin:CDR2End}
FR3	{FR3Begin:FR3End}
VCDR3Part	{CDR3Begin:VEndTrimmed}
CDR3	{CDR3Begin:CDR3End}

GermlineVCDR3Part	{CDR3Begin:VEnd}
ShortCDR3	{CDR3Begin(3):CDR3End(-3)}
VDJunction	{VEndTrimmed:DBeginTrimmed}
VJJunction	{VEndTrimmed:JBeginTrimmed}
DRegion	{DBegin:DEnd}
DCDR3Part	{DBeginTrimmed:DEndTrimmed}
DJJunction	{DEndTrimmed:JBeginTrimmed}
GermlineJCDR3Part	{JBegin:CDR3End}
JRegion	{JBegin:FR4End}
JRegionTrimmed	{JBeginTrimmed:FR4End}
JCDR3Part	{JBeginTrimmed:CDR3End}
FR4	{FR4Begin:FR4End}
CExon1	{CBegin:CExon1End}
CRegion	{CBegin:CEnd}

Appendix

A1. TCR/BCR refenrece sequences library

Default list and sequences of `V`, `D`, `J` and `C` genes used by MiXCR are taken from GenBank. Accession numbers of records used for each locus are listed in the following table:

Homo sapiens	TRA/TRD	NG_001332.2
	TRB	NG_001333.2
	TRG	NG_001336.2
	IGH	NG_001019.5
	IGK	NG_000834.1
	IGL	NG_000002.1
Mus musculus	TRA/TRD	NG_007044.1
	TRB	NG_006980.1
	TRG	NG_007033.1
	IGH	NG_005838.1
	IGK	NG_005612.1
	IGL	NG_004051.1

A2. Alignment and mutations encoding

MiXCR outputs alignments in `exportClones` and `exportAlignments` as a list of 7 fields separated by `|` symbol as follows:

`targetFrom` | `targetTo` | `targetLength` | `queryFrom` | `queryTo` | `mutations` | `alignmentScore` where

- `targetFrom` - position of first aligned nucleotide in **target sequence** (sequence of gene feature from reference V, D, J or C allele used in alignment; e.g. `VRegion` in TRBV12-2); this boundary is inclusive
- `targetTo` - next position after last aligned nucleotide in **target sequence**; this boundary is exclusive
- `targetLength` - length of **target sequence** (e.g. length of `VRegion` in TRBV12-2)
- `queryFrom` - position of first aligned nucleotide in **query sequence** (sequence of sequencing read or clonal sequence); this boundary is inclusive
- `queryTo` - next position after last aligned nucleotide in **query sequence**; this boundary is exclusive
- `mutations` - list of mutations from **target sequence** to **query sequence** (see below)
- `alignmentScore` - score of alignment

all positions are zero-based (i.e. first nucleotide has index 0)

Mutations are encoded as a list of single-nucleotide edits[2] (i.e. insertions, deletions or substitutions); if one apply these mutations to aligned subsequence of **target sequence**, one will obtain aligned subsequence of **query sequence**.

Each single mutation (single-nucleotide edit) is encoded in the following way (without any spaces; [] shows that corresponding fields are absent in some cases, see description):

`type` [`fromNucleotide`] `position` [`toNucleotide`]

- `type` of mutation (one letter):
  - `S` for substitution
  - `D` for deletion
  - `I` for insertion
- `fromNucleotide` is a nucleotide in **target sequence** affected by mutation (applicable only for substitutions and deletions; absent for insertions)
- `position` is a zero-based absolute position in **target sequence** affected by mutation; for insertions denotes position in **target sequence** right after inserted nucleotide
- `toNucleotide` nucleotide after mutation (applicable only for substitutions and insertions; absent for deletions)

Note, that for deletions and substitutions

```
targetSequence[position] == fromNucleotide
```

i.e. target sequence always have `fromNucleotide` at position `position`; for insertions `fromNucleotide` field is absent

Here are several examples of single mutations:

- `SA4T` - substitution of `A` at position `4` to `T`
- `DC12` - deletion of `C` at position `12`
- `I15G` - insertion of `G` before position `15`

Consider the following BLAST-like alignments encoded in MiXCR notation:

- Alignment without mutation

```
target = TTGTGCTGACAGATACCCC
query  = CGAGTGCTGACAGATACCGTCGATGCT

BLAST like alignment:

2 GTGCTGACAGATACC 16
  |||||||||||||||
3 GTGCTGACAGATACC 17

MiXCR alignment:

0|15|17|3|18||75.0
```

subsequence from `target` (from nucleotide 0 to nucleotide 15) was found to be identical to susequence from `query` (from nucleotide 3 to nucleotide 18).

- Alignment with mutation

  ```
  target = TTGTGCTGACAGATACCCC
  query  = CGAGTGCTATAGACTACCGTCGATGCT

  BLAST like alignment:

  2 GTGCTGACAGA-TACC 16
    ||||| | ||| ||||
  3 GTGCT-ATAGACTACC 17

  MiXCR alignment:

  0|15|17|3|18|DG7SC9TI13C|41.0
  ```

  so, to obtain subseqeunce from **query sequence** from 3 to 18 we need to apply the following mutations to subsequence of **target sequence** from 2 to 16:

  - deletion of `G` at position `7`
  - substitution of `C` at position `9` to `T`
  - insertion of `C` before at position `13`

---

1. Lefranc, Marie-Paule, et al. "IMGT unique numbering for immunoglobulin and T cell receptor variable domains and Ig superfamily V-like domains." Developmental & Comparative Immunology 27.1 (2003): 55-77.↩

2. Similar to what is used in definition of Levenshtein distance.↩