vdjtools Documentation

Release SNAPSHOT

Mikhail Shugay

Jul 05, 2018

Contents

VDJtools is an open-source Java/Groovy-based framework designed to facilitate analysis of immune repertoire sequencing (RepSeq) data. VDJtools computes a wide set of statistics and is able to perform various forms of cross-sample analysis. Both comprehensive tabular output and publication-ready plots are provided.

For the period of VDJtools development, there were no other software tools able to perform a comprehensive RepSeq post-analysis. Therefore most of the analysis of this kind was done using in-house scripts, which definitely leads to "re-inventing the bicycle" problem and loss of analysis reproducibility.

The main aims of the **VDJtools Project** are:

- To ensure consistency between post-analysis methods and results
- To save the time of bioinformaticians analyzing RepSeq data
- To create an API framework facilitating development of new RepSeq analysis applications
- To provide a simple enough command line tool so it could be used by immunologists and biologists with little computational background

VDJtools source code and binaries are located here.

CHAPTER 1

Table of Contents

1.1 Introduction

1.1.1 Features and workflow

1.1.2 Prerequisites

Software

As the core framework is complied into Java™ bytecode, it could in theory be run on any platform that has a Java Runtime Environment installed. The software is distributed in a form of an executable JAR file.

Note that the graphical output requires R programming language and corresponding modules to be installed.

See the *Installing VDJtools* section for more details.

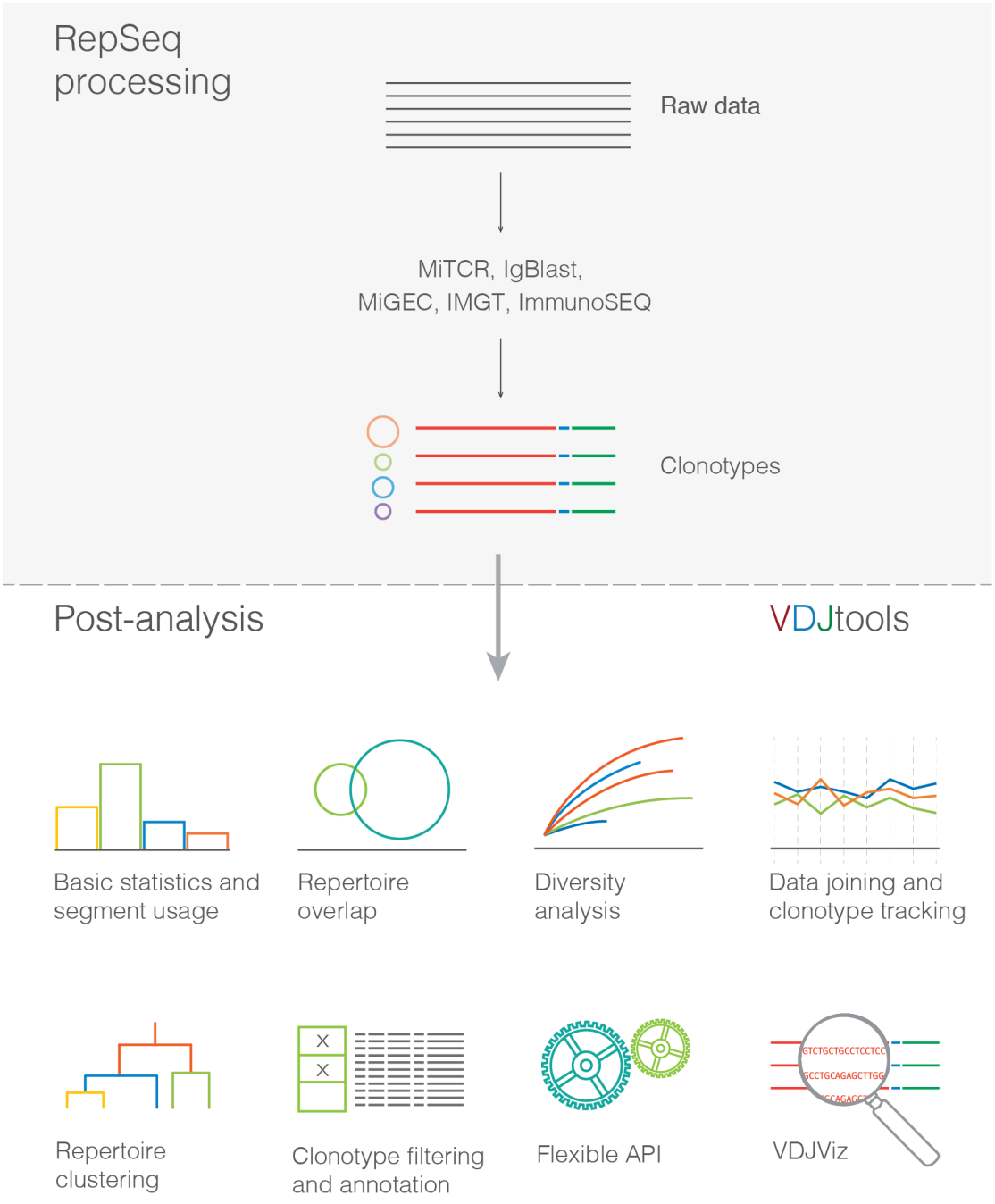
Hardware

VDJtools could be run on most of commodity hardware setups and is optimized to take advantage of parallel computations. The pipeline was successfully stress-tested using ~70 diverse samples containing repertoire snapshot of around 500,000 human T-cells on a UNIX server with 2x Intel Xeon E5-1620 and 64 Gb RAM.

Input

The framework is currently able to import and analyze the output of the following V-(D)-J junction mapping software and analysis platforms:

- MiTCR
- MiGEC
- IgBlast (via our MIGMAP wrapper)


RepSeq processing
Raw data
MiTCR, IgBlast, MiGEC, IMGT, ImmunoSEQ
Clonotypes
Post-analysis
VDJtools
Basic statistics and segment usage
Repertoire overlap
Diversity analysis
Data joining and clonotype tracking
Repertoire clustering
Clonotype filtering and annotation
Flexible API
VDJViz

- IMGT
- ImmunoSEQ
- VDJdb
- Vidjil
- RTCR
- MiXCR
- ImSEQ

For more details see the software section. VDJtools converts those files to its internal format (see *VDJtools format*) which is used throughout the pipeline.

Note: If this list is missing your favourite RepSeq processing software, please add a corresponding feature request Any contributions in form of pull requests are also welcome.

1.2 Installing VDJtools

1.2.1 Installing binaries

First make sure that you have installed Java Runtime Environment (JRE) v1.8 by running `java -version`. Any recent Linux distribution will provide it via its package manager. If not, or if your system is running MacOSX or Windows, download the JRE from Oracle.

Then download and unpack the VDJtools binaries from the latest release.

The program is then run by executing the following line:

```
java -jar path-to-vdjtools-X.X.X.jar
```

where `X.X.X` stands for the VDJtools version (omitted further for simplicity). This will bring up the list of available routines. To see the details (parameters, etc) for a specific routine execute

```
java -jar vdjtools.jar RoutineName -h
```

Windows

Dedicated VDJtools bundle can be downloaded from the release section and is marked with `.win.zip` suffix.

Linux

A VDJtools bundle can be downloaded from the release section which includes the required vdjtools.jar file.

All plotting is handled by R and will require several R packages some of which will be available via your distribution package manager. See *Setting up plotting routines* below.

MacOS

Installation can be performed using Homebrew package manager:

```
brew tap homebrew/science
brew tap mikessh/repseq
brew install vdjtools
```

Note that this sets `vdjtools` as a shortcut for `java -jar vdjtools-X.X.X.jar`. JVM arguments such as `-Xmx` can be still passed to the script, e.g. `vdjtools -Xmx20G CalcBasicStats ...`.

1.2.2 Setting up plotting routines

All plotting in VDJtools framework is performed via running R scripts. Therefore one needs to install R programming language and several of its packages. Make sure that

```
Rscript --version
```

runs successfully. Note that all R scripts were tested under R version 3.1.0.

The pre-compiled `*.win.zip` includes all the required R packages and the homebrew installation will install them automatically. In all other cases the required packages need to be manually installed.

These are the required packages:

CRAN package	Debian package
ape	r-cran-ape
circlize	
FField	
ggplot2	r-cran-ggplot2
gplots	r-cran-gplots
grid	
gridExtra	
MASS	r-cran-mass
plotrix	r-cran-plotrix
RColorBrewer	r-cran-rcolorbrewer
reshape	r-cran-reshape
reshape2	r-cran-reshape2
scales	r-cran-scales
VennDiagram	

If your Linux distribution includes pre-packaged versions of a package, those should be prefered. The following will install the existing for Debian and Debian based distributions such as Ubuntu and Mint:

```
apt-get install r-cran-ape r-cran-ggplot2 r-cran-gplots r-cran-mass \
  r-cran-plotrix r-cran-rcolorbrewer r-cran-reshape r-cran-reshape2 \
  r-cran-scales
```

while the other packages will have to be installed via R itself:

```
install.packages(c("circlize", "grid", "gridExtra", "VennDiagram"))
```

Alternatively, VDJtools has a ref:*Rinstall* routine:

```
java -jar vdjtools.jar Rinstall
```

This would also print the list of required R modules, so in case `Rinstall` fails, they could be installed manually by running the following command in R:

```
install.packages(c("reshape2", "FField", "reshape", "gplots",
                   "gridExtra", "circlize", "ggplot2", "grid",
                   "VennDiagram", "ape", "MASS", "plotrix",
                   "RColorBrewer", "scales"))
```

Note that most issues with package installation can be resolved by switching to correct CRAN mirror.

Dedicated windows binaries already have all R packages bundled, and the options summarized above should be considered only when troubleshooting R script execution issues.

1.2.3 Compiling from source

VDJtools could be compiled from source code using Apache Maven. Compilation should be performed under JRE v1.8 by running the following commands:

```
git clone https://github.com/mikessh/vdjtools.git
cd vdjtools/
mvn clean install
```

Binaries could then be found under the `vdjtools/target/` folder.

1.3 Usage

1.3.1 Command line usage

General way to execute VDJtools routines would be the following,

```
java -Xmx16G -jar vdjtools.jar RoutineName [arguments] -m metadata.txt output/prefix
```

Output prefix could be either an output directory name (if ended with `/`) or an output file prefix. Most VDJtools routines will append the prefix with an intuitive suffix and extension.

The `-m metadata.txt` argument specifies a metadata file with relative sample paths, sample names and any other information to provide this information later in analysis. For more details, see the *Metadata* section.

Alternatively, `-m` argument could be substituted with a space-separated list of files, e.g.

```
java -Xmx16G -jar vdjtools.jar RoutineName sample1.txt[.gz] sample2.txt[.gz] ...␣
↪output/prefix
```

Whether not explicitly used (such as in "...Plot" routines) and applicable, plotting is turned on with `-p` argument.

The `-h` argument will bring up help message for specified routine.

Warning: Consider allocating sufficient memory for Java Virtual Machine when running the pipeline. To do so, execute the java with the `-Xmx` argument, e.g.:

```
java -Xmx16G -jar vdjtools.jar RoutineName [arguments]
```

If insufficient amount memory is allocated, the Java Virtual Machine could drop with a *Java Heap Space Out of Memory* error.

Warning: Due to JAR loading overhead, running VDJtools for a batch of samples should be preferred to running VDJtools separately for each sample if possible. See *Metadata* section for more details.

Tip: Some routines could be memory demanding, especially when running sample intersection/joining/pooling with a high number of large (~1,000,000 clonotypes) datasets. Setting the `-Xmx` argument to 20-60Gb of memory should be enough for most purposes, e.g. 100 samples with 500,000 clonotypes on average.

Another way to work this around is to down-sample datasets to ~100,000 reads each using the *DownSample* routine.

1.3.2 Importing clonotype tables

In order to proceed with VDJtools analysis datasets should be converted to VDJtools format (see *VDJtools format*). To do this run either of the following commands:

```
java -Xmx16G -jar vdjtools.jar Convert -S software -m metadata.txt ... output_dir/
```

or

```
java -Xmx16G -jar vdjtools.jar Convert -S software sample1.txt[.gz] sample2.txt[.gz] .
↪.. output_dir/
```

An additional `-c` flag could be set to compress output files.

1.4 Examples

There are several data bundles and shell scripts that cover most of VDJtools usage scenarios available in the examples repository.

All of the examples refer to a folder with clonotype abundance tables (`samples/`). They contain a sample metadata file (`metadata.txt`, see *Metadata*) and a shell script `run.sh` that contains a line-by-line instructions to run various VDJtools routines. Sections below give a detailed explanation for post-analysis steps for the available example datasets.

For more details on individual VDJtools routines see the *Analysis modules* section.

Important: Samples in the repository are already converted to *VDJtools format*.

We assume that you have set the following variable pointing to VDJtools executable JAR file:

```
# Point to VDJtools executable and allocate enough memory for JVM
VDJTOOLS="java -Xmx20G -jar vdjtools.jar"
```

or in case the software was installed using Homebrew

```
VDJTOOLS="vdjtools -Xmx20G"
```

1.4.1 Aging

The aging experiment involving 39 healthy donors of various ages and both genders (see this paper for details). This example allows to have a look at how a diverse set of repertoire characteristics changes as we age. Post-analysis can be performed using the following commands:

```
# Basic analysis
# Generate summary tables
$VDJTOOLS CalcBasicStats -m metadata.txt out/0
$VDJTOOLS CalcSpectratype -m metadata.txt out/1
# -p for plotting, -f specifies metadata column for coloring,
# -n tells that factor is continuous
$VDJTOOLS CalcSegmentUsage -m metadata.txt -p -f age -n out/2
# the following routines run on a single sample
$VDJTOOLS PlotFancySpectratype ../samples/A4-i125.txt.gz out/3
$VDJTOOLS PlotSpectratypeV ../samples/A4-i125.txt.gz out/4
$VDJTOOLS PlotFancyVJUsage ../samples/A4-i125.txt.gz out/5

# Diversity estimation
# Plot clonality for a single sample
$VDJTOOLS PlotQuantileStats ../samples/A4-i125.txt.gz out/6
# Compute sample diversity estimates
$VDJTOOLS CalcDiversityStats -m metadata.txt out/7
# Perform rarefaction, -l specifies metadata column used as label
$VDJTOOLS RarefactionPlot -m metadata.txt -f age -n -l sample.id out/8

# Sample overlapping
# Overlap two replicate samples coming from the same donor
$VDJTOOLS OverlapPair -p ../samples/A4-i189.txt.gz ../samples/A4-i190.txt.gz out/9
# computes various metrics characterizing the similarity between repertoires
$VDJTOOLS CalcPairwiseDistances -m metadata.small.txt out/10
# plotting routine is separated from time-consuming batch intersection
# sample clustering is performed on this stage.
# Here we use relative sample overlap as metric and age as continuous factor
$VDJTOOLS ClusterSamples -p -f age -n -l sample.id out/10 out/10.age
# here we use Variable segment Jensen-Shannon divergence and sex as discrete factor
$VDJTOOLS ClusterSamples -p -e vJSD -f sex -l sample.id out/10 out/10.sex
```

(continues on next page)

(continued from previous page)

```
# Demonstrate sample operations and filtering
# Remove cross-sample contamination (-c produces compressed output)
$VDJTOOLS Decontaminate -m metadata.txt -c out/dec/
# Down-sample datasets to 10,000 reads
$VDJTOOLS Downsample -m metadata.txt -c -x 10000 out/ds/
# Filter non-coding clonotypes
$VDJTOOLS FilterNonFunctional -m metadata.txt -c out/nf/
# Join samples into a single clonotype abundance matrix
$VDJTOOLS JoinSamples -p -m metadata.small.txt out/12
# Pool samples together
$VDJTOOLS PoolSamples -m metadata.small.txt out/13

# Annotate each clonotype in each sample with insert size,
# total CDR3 hydrophobicity and other basic and amino acid properties
$VDJTOOLS Annotate -m metadata.txt out/annot/
```

The code block above shows example usage for nearly all available commands. Rarefaction plot in the aging case displays a strong age-related diversity decrease. If running on a server with ~24GB of available RAM one can try out repertoire clustering for the whole experiment (replace `metadata.small.txt` with `metadata.txt` for corresponding routines) which will show some interesting age-related trends.

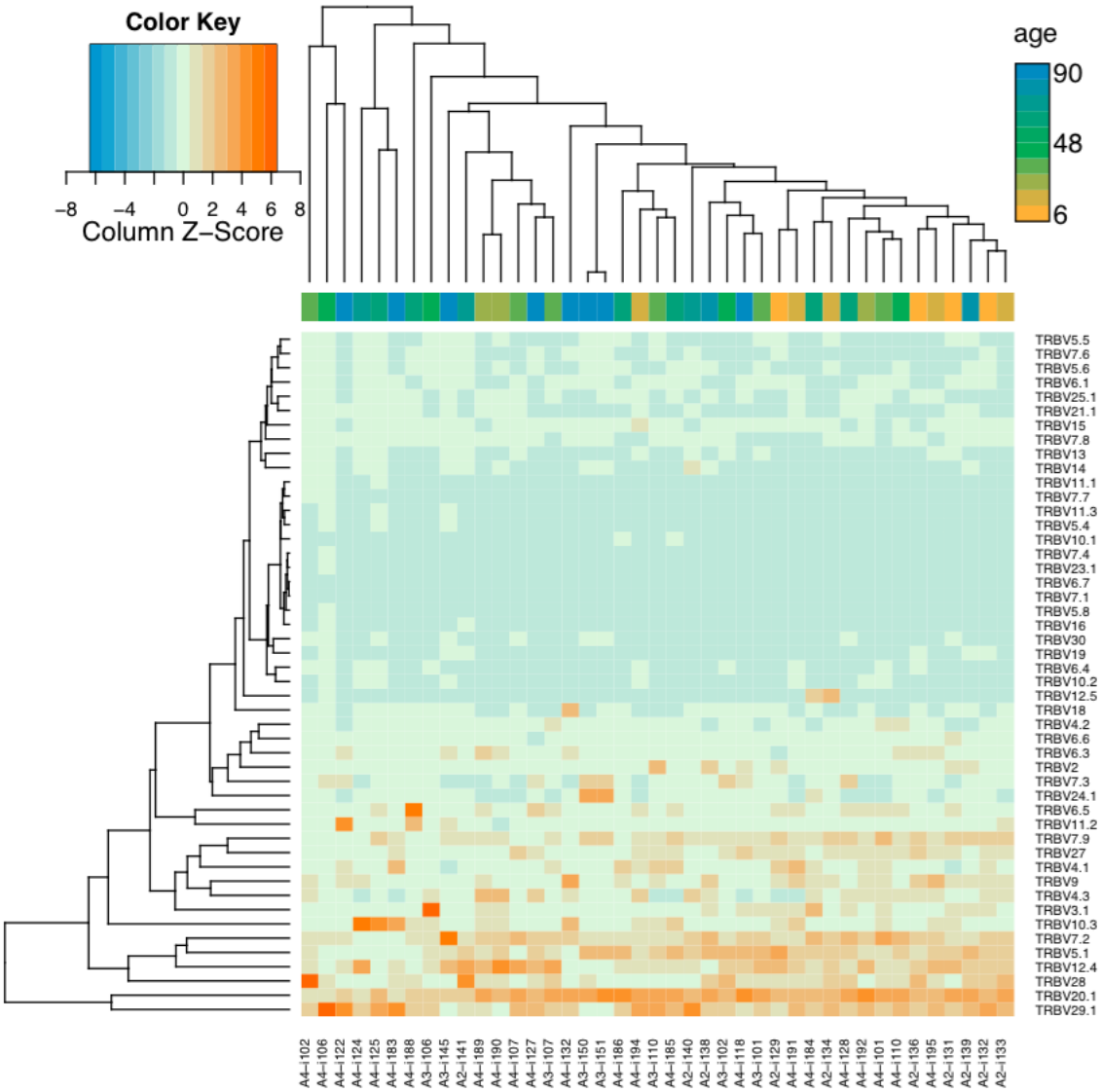
Variable segment usage in healthy donors of various age. Note non-random sample grouping within dendrogram which can be attributed to stochastic antigen-driven expansion of clonotypes as we age. See *CalcSegmentUsage* for a detailed description of this plot.

1.4.2 HSCT

Hematopoietic stem cell transfer (HSCT) is a great model for clonotype tracking and studying how the diversity of immune repertoire restores following myeloablation. Post-analysis can be performed using the following commands:

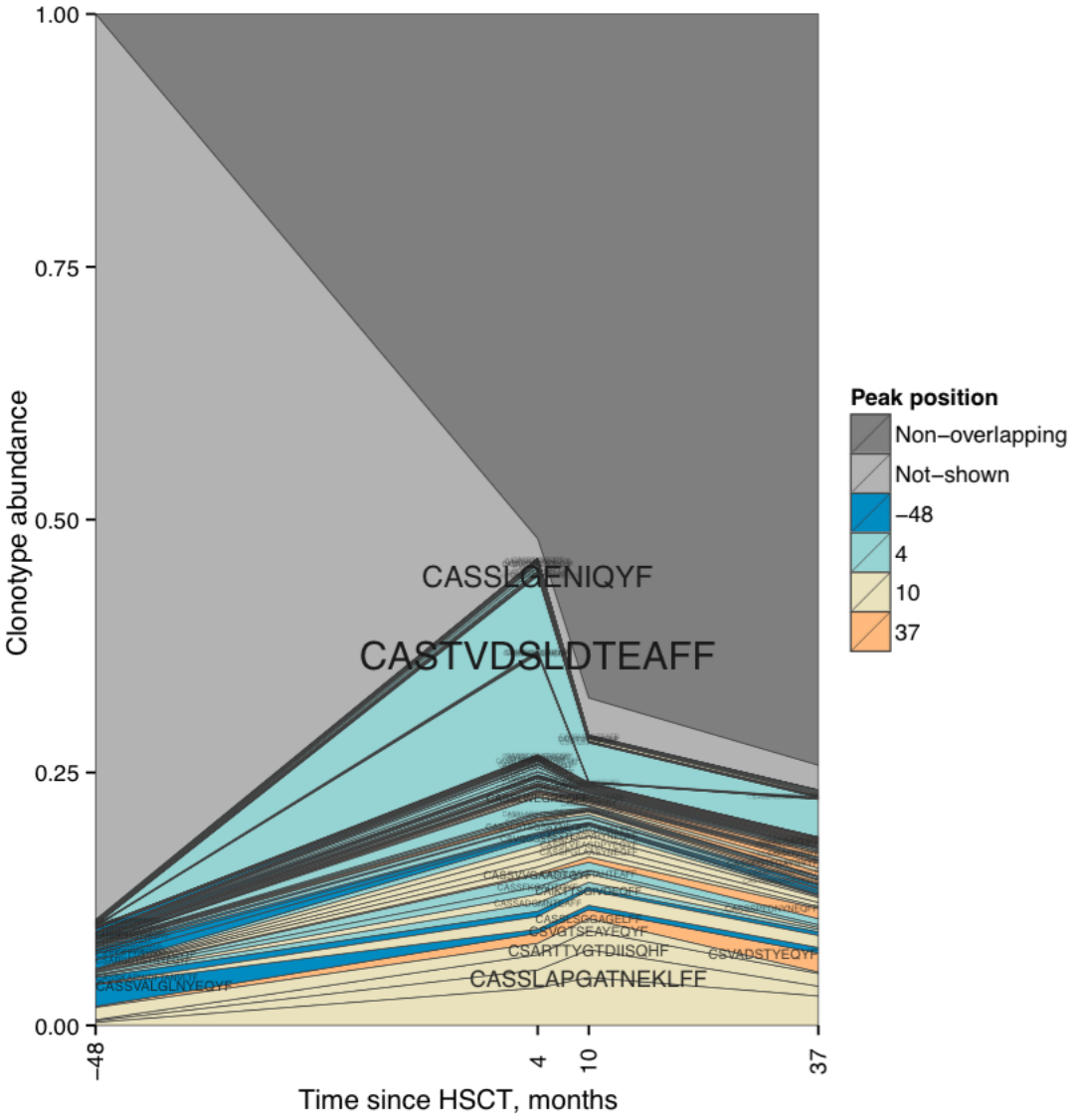
```
# Some basic analysis, same as above
$VDJTOOLS CalcBasicStats -m metadata.txt out/0
$VDJTOOLS CalcSpectratype -m metadata.txt out/1
$VDJTOOLS CalcSegmentUsage -m metadata.txt -p -f "Time post HSCT, months" -n out/2

# Diversity estimates
# Note that selecting the factor having spaces in its name requires using double␣
↪quotes
$VDJTOOLS CalcDiversityStats -m metadata.txt out/3
$VDJTOOLS RarefactionPlot -m metadata.txt -f "Time post HSCT, months" -n -l sample.id␣
↪out/4

# Clonotype tracking
# Show repertoire changes that happen directly after HSCT
$VDJTOOLS OverlapPair -p ../samples/minus48months.txt.gz ../samples/4months.txt.gz␣
↪out/5
# Next routine by default detects clonotypes that are present in 2 or more samples
# and builds a time course for them,
# but here we trace clonotypes from first time point setting -x 0
$VDJTOOLS TrackClonotypes -m metadata.txt -f "Time post HSCT, months" -x 0 -p out/6
```

RarefactionPlot output shows how repertoire diversity is lost and restored during post-HSCT period. The output of *ScanDatabase (DEPRECATED since v1.0.5, use VDJmatch)* displays that CMV- and EBV-specific clonotypes start


Color Key
−8 −4 0 2 4 6 8
Column Z−Score
age
90
48
6
TRBV5.5
TRBV7.6
TRBV5.6
TRBV6.1
TRBV25.1
TRBV21.1
TRBV15
TRBV7.8
TRBV13
TRBV14
TRBV11.1
TRBV7.7
TRBV11.3
TRBV5.4
TRBV10.1
TRBV7.4
TRBV23.1
TRBV6.7
TRBV7.1
TRBV5.8
TRBV16
TRBV30
TRBV19
TRBV6.4
TRBV10.2
TRBV12.5
TRBV18
TRBV4.2
TRBV6.6
TRBV6.3
TRBV2
TRBV7.3
TRBV24.1
TRBV6.5
TRBV11.2
TRBV7.9
TRBV27
TRBV4.1
TRBV9
TRBV4.3
TRBV3.1
TRBV10.3
TRBV7.2
TRBV5.1
TRBV12.4
TRBV28
TRBV20.1
TRBV29.1
A4−102 A4−106 A4−122 A4−124 A4−125 A4−183 A4−188 A3−106 A3−145 A2−141 A4−189 A4−190 A4−107 A4−127 A3−107 A4−132 A3−150 A3−151 A4−186 A4−194 A3−110 A4−185 A2−140 A2−138 A3−102 A4−118 A3−101 A2−129 A4−191 A4−184 A2−134 A4−128 A4−192 A4−101 A4−110 A2−136 A4−195 A2−131 A2−139 A2−132 A2−133


NEEDY

to dominate in the repertoire: they comprise ~4% of repertoire prior to HSCT, but increase more than 2-fold in post-HSCT period.



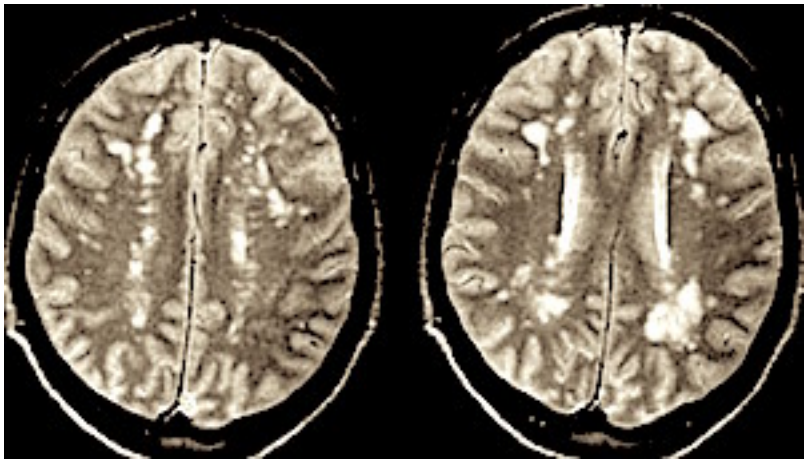
Clonotype abundance plot. Stacked abundance for top 100 clonotypes at different time points is shown.

1.4.3 Multiple sclerosis (MS)

MS is a complex autoimmune disorder that does not show a strong T-cell clonotype bias (see Turner et al.). Still some high-level repertoire features such as diversity and segment usage are distinct between affected persons and healthy donors.

```
# Diversity estimation
# Perform rarefaction analysis and compare repertoire diversity
# between MS patients and healthy donors
$VDJTOOLS RarefactionPlot -m metadata.txt -l sample_id -f state diversity/
```

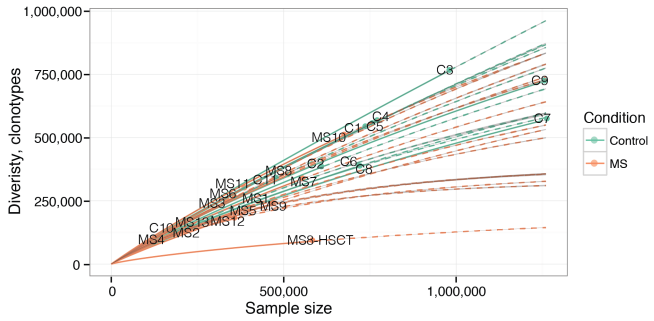
(continues on next page)



(continued from previous page)

```
$VDJTOOLS CalcDiversityStats -m metadata.txt diversity/

# Shows that MS cluster is not that compact as the
# cluster of healthy donors suggesting
# private nature of MS clonotypes
# -i aa!nt is used to discard CDR3 nucleotide sequence matches
    # (note the ! character should be escaped when running on Unix system: \!)
# and focus on amino-acid matches as strong cross-contamination is present
$VDJTOOLS CalcPairwiseDistances -i aa\!nt -m metadata.txt overlap/
$VDJTOOLS ClusterSamples -p -f state -i aa\!nt overlap/ overlap/state
$VDJTOOLS TestClusters -i aa\!nt overlap/state overlap/state

# Shows V usage level trends and cluster samples based on V usage profiles
$VDJTOOLS CalcSegmentUsage -m metadata.txt -p -f state vusage/

# Shows details of repertoire changes for MS8 patient that has
# undergone a HSCT (MS14 is a post-HSCT blood sample)
$VDJTOOLS OverlapPair -p ../samples/MS8.txt.gz ../samples/MS14.txt.gz overlap/
$VDJTOOLS PlotFancyVJUsage ../samples/MS8.txt.gz hsct/MS8
$VDJTOOLS PlotFancyVJUsage ../samples/MS14.txt.gz hsct/MS8-HSCT
```

Below is an example of *RarefactionPlot* graphical output.



Rarefaction analysis of MS and healthy donor repertoires. Note that rarefaction curves for MS patients are generally lower than those for healthy donors, indicating the presence of clonal expansion in former.

1.5 Input

1.5.1 Clonotype tables

The processing stage of RepSeq analysis starts with mapping of Variable, Diversity and Joining segments. Mapped reads are then assembled into clonotypes and stored as a clonotype abundance tables.

Clonotype

VDJtools **clonotype** specification includes the following fields:

- Variable (*V*) segment name.
- Diversity (*D*) segment name for some of the receptor chains (TRB, TRD and IGH). Set to . if not aplicable or D segment was not identified.
- Joining (*J*) segment name.
- Complementarity determining region 3 nucleotide sequence (*CDR3nt*). CDR3 starts with Variable region reference point (conserved Cys residue) and ends with Joining segment reference point (conserved PheTrp).
- Translated CDR3 sequence (*CDR3aa*).
- Somatic hypermutations (*SHMs*) in the variable segment (antibody only, **planned**).

Important: For ambiguous segment assignments encoded by a comma separated list of segment names only the first one is selected.

Hint: In case of non-coding CDR3 sequences, the convention is to translate in both directions: upstream from V segment reference point and downstream from J segment reference point. The resulting sequence (e.g. `CASSLA_TNEKFF`) is linked by a _ symbol that marks the incomplete codon.

Clonotype **abundance** data is represented by *count* and *frequency* fields:

- *Count*: number of reads or cDNA/DNA molecules in case UMIs are used.
- *Frequency*: the share of clonotype in the sample. While seemingly redundant, this property is left for compatibility with cases when the sample represents a subset of another one, e.g. clonotypes from PBMCs filtered by intersection with lymph node clonotypes.

The following fields are optional, but are used for computing various statistics and visualization:

- *Vend*, *Dstart*, *Dend* and *Jstart* - marking V, D and J segment boundaries within CDR3 nucleotide sequence (inclusive)

Tip: VDJtools accepts gzip-compressed files, such files should have an `.gz` suffix. Input data should be provided in a form of tab-delimited table.

1.5.2 VDJtools format

This is a core tabular format for VDJtools. All datasets should be converted to this format using the *Convert* routine prior to analysis. Columns 8-10 are optional.

col-umn1	col-umn2	column3	col-umn4	col-umn5	col-umn6	col-umn7	col-umn8	col-umn9	col-umn10	col-umn11
count	fre-quency	CDR3nt	CDR3aa	V	D	J	Vend	Dstart	Dend	Jstart
1176	9.90E-02	TGT-GCCAGC...AAGCTT	CAST...EAF	TRBV12-4	TRBD1	TRBJ1-1	11	14	16	23

All additional columns after column 10 will be considered as clonotype annotations and carried over unmodified during most stages of VDJtools analysis. This is especially useful when processing results of *Annotation* and *ScanDatabase (DEPRECATED since v1.0.5, use VDJmatch)* routines.

1.5.3 Formats supported for conversion

MiTCR

Output from MiTCR software (executable jar, documentation) in `full` mode can be used without any pre-processing. Corresponding table should start with **two header lines** (default MiTCR output stores processing options and version in the first line), followed by a clonotype list.

Run *Convert* routine with `-S mitcr` argument to prepare datasets in this format for VDJtools analysis.

MiGEC

MiGEC is a software for V/D/J mapping and CDR3 extraction that relies on BLAST algorithm for running alignments. MIGEC software additionally implements processing of unique molecular identifier (UMI)-tagged libraries for error correction and dataset normalization. Default output of MIGEC software can be directly used with VDJtools.

Run *Convert* routine with `-S migec` argument to prepare datasets in this format for VDJtools analysis.

IgBlast (MIGMAP)

As IgBlast doesn't compute a canonical clonotype abundance table, VDJtools supports output of MIGMAP, a versatile IgBlast wrapper. Note that currently no somatic hypermutation (SHM) information is imported by VDJtools, neither there are any dedicated VDJtools routines to analyze SHM profiles, but you check out post-analysis provided by MIGMAP.

Run *Convert* routine with `-S migmap` argument to prepare datasets in this format for VDJtools analysis.

ImmunoSEQ

One of the most commonly used RepSeq data format, more than 90% of recently published studies were performed using immunoSEQ assay. We have implemented a parser for clonotype tables as provided by Adaptive Biotechnologies.

- The resulting datasets for most studies that use ImmunoSEQ technology can be accessed and exported using the ImmunoSEQ Analyzer.
- Example datasets in this format could be found in the Supplementary Data section of Spreafico R et al. Ann Rheum Dis. 2014.
- Column header information was taken from **page 24** of the immunoSEQ Analyzer manual

- VDJtools will use V/J segment information only at the family level, as many of the clonotypes miss segment (*-X*) and allele (*-X*0Y*) information. The clonotype table is then collapsed to handle unique V/J/CDR3 entries.
- Raw clonotype tables in this format do not contain CDR3 nucleotide sequence. Instead, an entire sequencing read (first column) is provided. Therefore, we have implemented additional algorithms for CDR3 extraction and "virtual" translation to tell out-of-frame clonotypes from partially read ones.

Attention: Some of the clonotype entries will dropped during conversion as they contain an incomplete CDR3 sequence (lacking J segment), which is due to short reads used in immunoSEQ assay, see this blog post for details.

Run *Convert* routine with `-S immunoseq` argument to prepare datasets in this format for VDJtools analysis. Note that there are currently two possible ImmunoSEQ output formats that have different column naming:

- This option should be used in case you have selected

  `Export samples` option in the ImmunoSEQ analyzer.
- In case you have used the `Export samples v2` option you should pass the `-S immunoseqv2` argument to VDJtools Convert routine.

IMGT/HighV-QUEST

Another commonly used RepSeq processing tool is the IMGT/HighV-QUEST web server.

Please refer to the official documentation to see the description of output files and their formats.

Tip: The output for each submission consists of several files and only

```
3_Nt-sequences_${chain}_${sx}_${date}.txt
```

should be used as an input for VDJtools *Convert* routine.

Run *Convert* routine with `-S imgthighvquest` argument to prepare datasets in this format for VDJtools analysis.

VDJdb

VDJtools has native support for the analysis of clonotype tables annotated with VDJdb software. Note that as those tables can list the same clonotype several times with different annotation, they should not be used directly in most VDJtools routines (e.g. diversity statistics), check out VDJdb README for corresponding guidelines and workarounds.

Vidjil

VDJtools supports parsing output Json files produced by the Vidjil software. VDJtools will only use top clonotypes which have V/D/J detalization in the output.

RTCR

VDJtools supports parsing the `results.tsv` table with clonotype list generated by the RTCR software.

Run *Convert* routine with `-S rtcr` argument to prepare datasets in this format for VDJtools analysis.

MiXCR

Output from MiXCR software `export` routine in `full` (default) mode can be used without any pre-processing.

Run *Convert* routine with `-S mixcr` argument to prepare datasets in this format for VDJtools analysis.

IMSEQ

Output from IMSEQ software can be used if results are collapsed to nucleotide-level clonotypes using `-on` argument with IMSEQ.

Run *Convert* routine with `-S imseq` argument to prepare datasets in this format for VDJtools analysis.

1.5.4 Metadata

Most VDJtools routines will accept multiple sample files as command line arguments for batch processing. This should be always preferred over multiple calls to VDJTools with a single sample due to the initialisation time of VDJTools.

An alternative way to specify a sample batch is to pass the sample metadata file with `-m` option. The file should contain sample file paths, sample names. It can be also supplemented with optional metadata columns that will be appended to analysis results and can be used for plottings.

Additionally, for each step that involves modification of samples (e.g. converting or filtering non-functional rearrangements) a new metadata file will be created in the folder containing the processed sample batch.

Note:

- VDJtools will append metadata fields to its output tables to facilitate the exploration of analysis results.
- Metadata entries are used as a factor in some analysis routines and most plotting routines.
- When performing tasks that involve modifying clonotype abundance tables themselves, such as down-sampling, VDJtools will also provide a copy of metadata file pointing to newly generated samples.
- Newly generated metadata file would contain an additional `..filter..` column, which has a comma-separated list of filters that were applied. For example the *DownSample* routine run with `-n 50000` will append `ds:50000` to the `..filter..` column. Note that this column name is reserved and should not be modified.
- Some routines for working with metadata files can be found in *Utilities* section.

Below are the basic guidelines for creating a metadata file.

- Metadata file should be a tab-delimited table, e.g.

#file.name	sample.id	col.name	...
sample_1.txt	sample_1	A	...
sample_2.txt	sample_2	A	...
sample_3.txt	sample_3	B	...
sample_4.txt	sample_4	C	...
...	...	...	...

- Header is mandatory, first two columns should be named **file_name** and **sample_id**. Names of the remaining columns will be later used to specify metadata variable name
- First two columns should contain the file name and sample id respectively.

  - The file name should be either an absolute path (e.g. `/Users/username/somedir/file.txt`) or a path relative to the parent directory of metadata file (e.g. `../file.txt`)
  - Sample IDs should be unique
- Columns after **sample.id** are treated as metadata entries. There are also several cases when info from metadata is used during execution:
  - VDJtools plotting routines could be directed to use metadata fields for naming samples and creating intuitive legends. If column name contains spaces it should be quoted, e.g. `-f "patient id"`
  - Metadata fields are categorized as factor (contain only strings), numeric (contain only numbers) and semi-numeric (numbers and strings). Numeric and semi-numeric fields could be used for gradient coloring by plotting routines.

1.6 Analysis modules

1.6.1 Table of VDJtools modules

VDJtools software package contains a comprehensive set of immune repertoire post-analysis routines, which are subdivided into several analysis modules. Each module's section provides command line usage syntax and parameter descriptions for each of the routines, as well as output example and description.

Basic analysis

Summary statistics, spectratyping, etc

- *CalcBasicStats* Computes summary statistics for samples: read counts, mean clonotype sizes, number of non-functional clonotypes, etc
- *CalcSegmentUsage* Computes Variable (V) and Joining (J) segment usage profiles
- *CalcSpectratype* Computes spectratype, the distribution of clonotype abundance by CDR3 sequence length
- *PlotFancySpectratype* Plots spectratype explicitly showing top N clonotypes
- *PlotFancyVJUsage* Plots the frequency of different V-J pairings
- *PlotSpectratypeV* Plots distribution of V segment abundance by resulting CDR3 sequence length

Diversity estimation

Repertoire richness and diversity

- *PlotQuantileStats* Visualizes repertoire clonality
- *RarefactionPlot* Performs rarefaction analysis
- *CalcDiversityStats* Computes repertoire diversity estimates

Repertoire overlap analysis

Clonotype sharing between samples

- *OverlapPair* Computes intersection between a pair of samples
- *CalcPairwiseDistances* Computes pairwise intersections for a list of samples

- *ClusterSamples* Performs sample clusterization according to the results of batch intersection
- *TrackClonotypes* Time-course analysis for a sequence of samples

Pre-processing

Filtering and resampling

- *Correct* Performs a frequency-based erroneous clonotype correction
- *Decontaminate* Filters possible cross-sample contaminations in a set of samples
- *DownSample* Performs down-sampling, i.e. takes a subset of random reads from sample(s)
- *FilterNonFunctional* Filters non-functional clonotypes
- *SelectTop* Selects a fixed number of top (most abundant) clonotypes from sample(s)
- *FilterByFrequency* Filters clonotypes based on a specified frequency threshold.
- *ApplySampleAsFilter* Filters clonotypes that are present in a specified sample from sample(s)
- *FilterBySegment* Filters clonotypes according to their V/D/J segment

Operate on clonotype tables

Clonotype table operations

- *PoolSamples* Pools clonotypes from several samples together
- *JoinSamples* Joins a set of samples and generates clonotype abundance profiles

Annotation

Functional annotation of clonotype tables (antigen specificity, amino acid properties, etc)

- *CalcCdrAAProfile* Builds a profile of CDR3 regions (V germline, V-D junction, ...) using a set of amino-acid physical properties
- *Annotate* Computes a set of basic (insert size, ...) and amino acid physical properties (GRAVY, ...) for clonotypes
- *ScanDatabase (DEPRECATED since v1.0.5, use VDJmatch)* Queries a database containing clonotypes of known antigen specificity.

Utilities

Some useful utilities

- *FilterMetadata* Filters metadata file by values in specified column
- *SplitMetadata* Splits metadata file by specified columns
- *Convert* Converts from one software format to another
- *RInstall* Installs necessary R dependencies

Output

Each routine generates a comprehensive tabular output and some produce optional graphical output. In case of graphical output, the corresponding R script with specified arguments (at the beginning of the script, commented) will be stored to the analysis folder. Thus, user can uncomment the script arguments, modify the script and re-run it. This behavior be disabled by running VDJtools with `discard_scripts` argument prior to routine name.

By default, all graphical output is generated in PDF format, to generate PNG images use "`--plot-type png` option.

When running routines that output clonotype tables consider the following:

- Joint and pooled samples are stored in VDJtools fomat
- Samples produced using *ScanDatabase (DEPRECATED since v1.0.5, use VDJmatch)* or *Annotation* routine are in VDJtools format and include additional annotation columns. Annotation columns are retained when running most of VDJtools routines
- When loading a joint/pooled sample into VDJtools, clonotype abundance vectors, incidence counts, etc will be treated as clonotype level annotations
- Annotation columns will not be preserved when joining/pooling annotated samples, a workaround

here will be to use *ApplySampleAsFilter* routine

Attention: When exporting a table generated by one of VDJtools routines into R use the following command to parse the input correctly:

```
read.table("some_table.txt", header=T, quote="", sep = "\t")
```

1.6.2 Common parameters

There are several parameters that are commonly used among analysis routines:

Shorthand	Long name	Argument	Description
`-h`	`--help`		Brings up the help message for selected routine
`-m`	`--metadata`	path	Path to metadata file. Should point to a tab-delimited file with the first two columns containing sample path and sample id respectively, and the remaining columns containing user-specified data. See *Metadata* section
`-u`	`--unweighted`		If present as an option and not set, all statistics will be weighted by clonotype frequency
`-i`	`--intersection-type`	string	*Overlap type*, that specifies which clonotype features (CDR3 sequence, V/J segments, hypermutations) will be compared when checking if two clonotypes match. Allowed values: `strict`,`nt`,`ntV`,`ntVJ`,`aa`,`aaV`,`aaVJ` and `aa!nt`.
`-p`	`--plot`		[*plotting*] Enable plotting for routines that supports it.
	`--plot-type`	pdf/png	[*plotting*] Specifies whether to generate a PDF or PNG file. While latter could be easily embedded, PDF plots have superior quality.
`-f`	`--factor`	string	[*plotting*] Name of the sample metadata column that should be treated as factor. If the name contains spaces, the argument should be surrounded with double quotes, e.g. `-f "Treatment type"`
`-n`	`--factor-numeric`		[*plotting*] Treat the factor as numeric?
`-l`	`--label`	string	[*plotting*] Name of the sample metadata column that should be treated as label. If the name contains spaces, the argument should be surrounded with double quotes, e.g. `-l "Patient id"`
`-c`	`--compress`	path	Compress resulting clonotype tables using GZIP.

Overlap type

Some of VDJtools routines require to define clonotype matching strategy when computing clonotype sharing between samples. This parameter is also used when collapsing clonotype tables, e.g. a common situation is when one is interested in estimating the extent of convergent recombination, which is the number of distinct nucleotide CDR3 sequences per one CDR3 amino acid sequence. This requires to collapse clonotype table by identical CDR3aa field.

The list of strategies is defined below.

Shorthand	Rule	Note
strict	**CDR3nt** (AND) **V** (AND) **J** (AND) **SHMs**	Require full match for receptor nucleotide sequence
nt	**CDR3nt**	
ntV	**CDR3nt** (AND) **V**	
ntVJ	**CDR3nt** (AND) **V** (AND) **J**	
aa	**CDR3aa**	
aaV	**CDR3aa** (AND) **V**	
aaVJ	**CDR3aa** (AND) **V** (AND) **J**	
aa!nt	**CDR3aa** (AND)((NOT) **CDR3nt**)	Removes nearly all contamination bias from overlap results. Should not be used for samples from the same donor/tracking experiments

As somatic hypermutations (SHMs) are currently not supported by VDJtools, `strict` and `ntVJ` options are identical. See VDJtools *Clonotype* specification for details.

1.7 Basic analysis

1.7.1 CalcBasicStats

This routine computes a set of basic sample statistics, such as read counts, number of clonotypes, etc.

Command line usage

```
$VDJTOOLS CalcBasicStats \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-u`	`--unweighted`		If not set, all statistics will be weighted by clonotype frequency
`-h`	`--help`		Display help message

Tabular output

The following table with `.basicstats.txt` suffix is generated,

Column	Description
sample_id	Sample unique identifier
...	Metadata columns. See *Metadata* section
count	Number of reads in a given sample
diversity	Number of clonotypes in a given sample
mean_frequency	Mean clonotype frequency
geomean_frequency	Geometric mean of clonotype frequency
nc_diversity	Number of non-coding clonotypes
nc_frequency	Frequency of reads that belong to non-coding clonotypes
mean_cdr3nt_length	Mean length of CDR3 nucleotide sequence. Weighted by clonotype frequency
mean_insert_size	Mean number of inserted random nucleotides in CDR3 sequence. Characterizes V-J insert for receptor chains without D segment, or a sum of V-D and D-J insert sizes
mean_ndn_size	Mean number of nucleotides that lie between V and J segment sequences in CDR3
convergence	Mean number of unique CDR3 nucleotide sequences that code for the same CDR3 amino acid sequence

Graphical output

none

1.7.2 CalcSegmentUsage

This routine computes Variable (V) and Joining (J) segment usage vectors, i.e. the frequency of associated reads for each of V/J segments present in sample(s). If plotting is on, will also perform clustering for V/J usage vectors and samples *à la* gene expression analysis.

Command line usage

```
$VDJTOOLS CalcSegmentUsage \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-u`	`--unweighted`		Will compute the number of unique clonotypes with a given V/J segment. Counts the number of reads otherwise
`-p`	`--plot`		Turns on plotting. See *Common parameters*
`-f`	`--factor`	string	Specifies plotting factor. See *Common parameters*
`-n`	`--numeric`		Specifies if plotting factor is numeric. See *Common parameters*
`-l`	`--label`	string	Specifies label used for plotting. See *Common parameters*
`-h`	`--help`		Display help message

Tabular output

The following tables with `.segments.[unwt or wt depending on -u parameter].[V or J].txt` suffix are generated,

Column	Description
sample_id	Sample unique identifier
...	Metadata columns. See *Metadata* section
Segment name, e.g. TRBJ1-1	Segment frequency in a given sample
Next segment name, e.g. TRBJ1-2	...
...	...

Graphical output

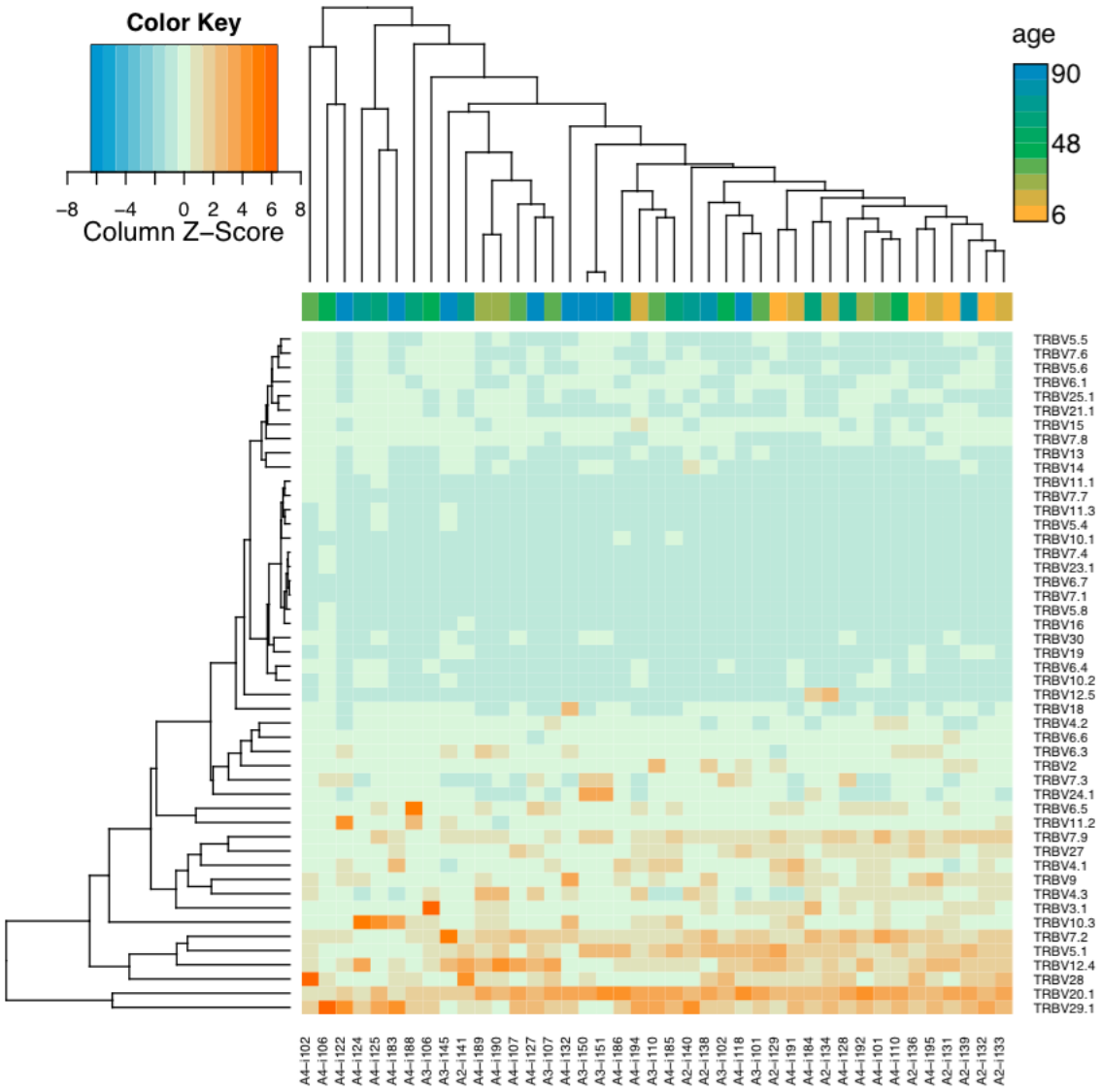
Images, having the same name as tables, with the exception of `.pdf` extension, are created if plotting is on. They display segment usage heatmap and hierarchical clustering for samples and segment.

This figure will be created using *heatmap.2* function from gplots R package with default clustering parameters.

Sample clustering based on Variable segment usage. Weighted Variable usage profiles are used, hierarchical clustering is performed using euclidean distance. A continuous factor is displayed (`-n -f age` argument).

1.7.3 CalcSpectratype

Calculates spectratype, that is, histogram of read counts by CDR3 nucleotide length. The spectratype is useful to detect pathological and highly clonal repertoires, as the spectratype of non-expanded T- and B-cells has a symmetric gaussian-like distribution.


Color Key
−8 −4 0 2 4 6 8
Column Z−Score
age
90
48
6
TRBV5.5
TRBV7.6
TRBV5.6
TRBV6.1
TRBV25.1
TRBV21.1
TRBV15
TRBV7.8
TRBV13
TRBV14
TRBV11.1
TRBV7.7
TRBV11.3
TRBV5.4
TRBV10.1
TRBV7.4
TRBV23.1
TRBV6.7
TRBV7.1
TRBV5.8
TRBV16
TRBV30
TRBV19
TRBV6.4
TRBV10.2
TRBV12.5
TRBV18
TRBV4.2
TRBV6.6
TRBV6.3
TRBV2
TRBV7.3
TRBV24.1
TRBV6.5
TRBV11.2
TRBV7.9
TRBV27
TRBV4.1
TRBV9
TRBV4.3
TRBV3.1
TRBV10.3
TRBV7.2
TRBV5.1
TRBV12.4
TRBV28
TRBV20.1
TRBV29.1
A4−102 A4−106 A4−122 A4−124 A4−125 A4−183 A4−188 A3−106 A3−145 A2−141 A4−189 A4−190 A4−107 A4−127 A3−107 A4−132 A3−150 A3−151 A4−186 A4−194 A3−110 A4−185 A2−140 A2−138 A3−102 A4−118 A3−101 A2−129 A4−191 A4−184 A2−134 A4−128 A4−192 A4−101 A4−110 A2−136 A4−195 A2−131 A2−139 A2−132 A2−133

Command line usage

```
$VDJTOOLS CalcSpectratype \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-u`	`--unweighted`		Instead of computing read frequency, will compute the number of unique clonotypes with specific a CDR3 length
`-a`	`--amino-acid`		Will use CDR3 amino acid sequences for calculation instead of nucleotide ones
`-h`	`--help`		Display help message

Tabular output

The following table with `.spectratype.[aa or nt depending on -a parameter].[unwt or wt depending on -u parameter].txt` suffix is generated,

Column	Description
sample_id	Sample unique identifier
...	Metadata columns. See *Metadata* section
CDR3 length, e.g. 22	Frequency of reads with a given CDR3 length in a given sample
Next CDR3 length, 23	...
...	...

Graphical output

none

1.7.4 PlotFancySpectratype

Plots a spectratype that also displays CDR3 lengths for top N clonotypes in a given sample. This plot allows to detect the highly-expanded clonotypes.

Command line usage

```
$VDJTOOLS PlotFancySpectratype [options] sample.txt output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-t`	`--top`	int	Number of top clonotypes to visualize. Should not exceed 20, default is 10
`-h`	`--help`		Display help message

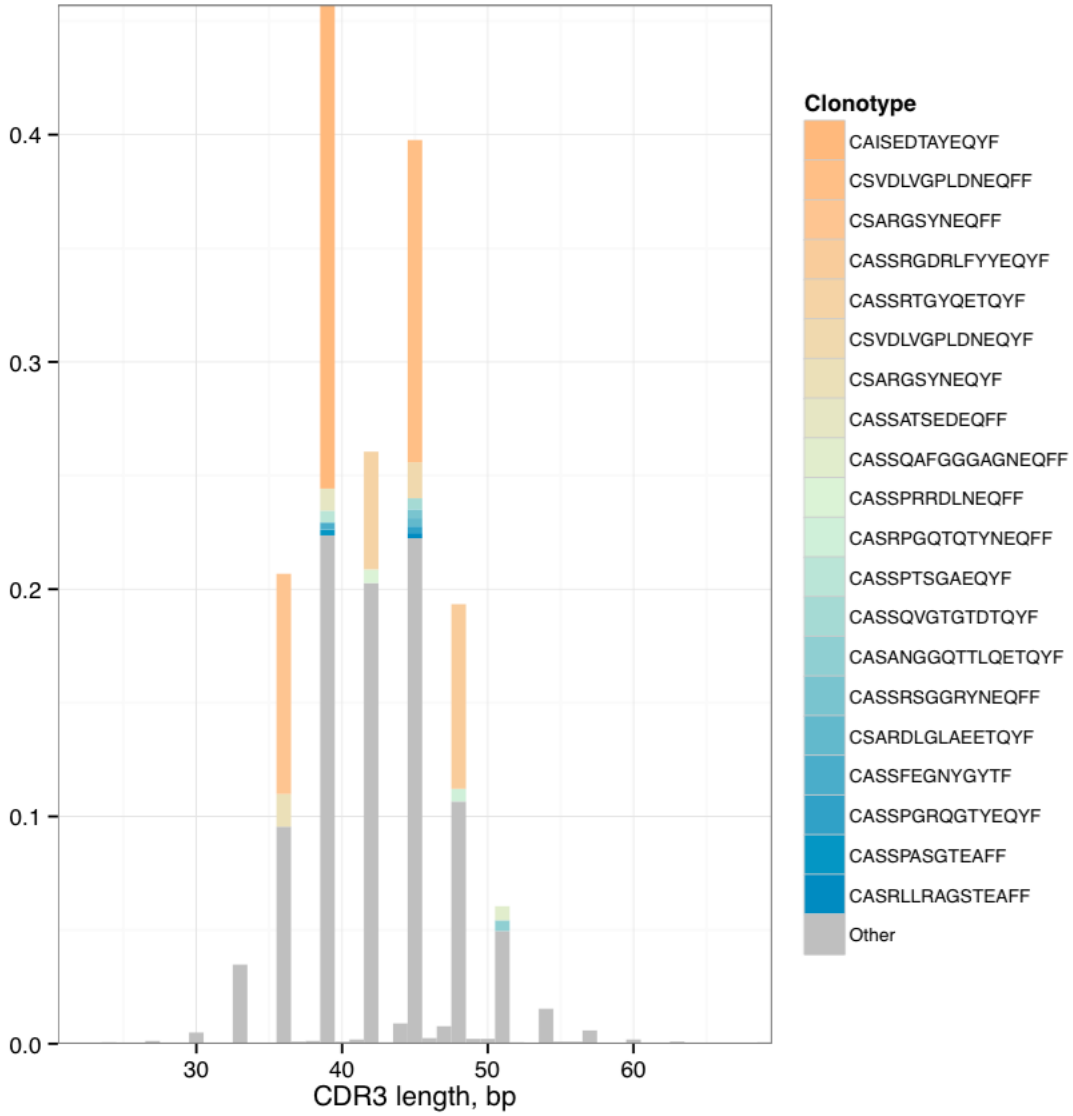
Tabular output

Following table with `.fancyspectra.txt` prefix is generated,

Column	Description
Len	Length of CDR3 nucleotide sequence
Other	Frequency of clonotypes with a given CDR3 length, other than top N
Clonotype#N, e.g. CASRLLRAG-STEAFF	Clonotype frequency, at the corresponding CDR3 length
Clonotype#N-1	...
...	...

Graphical output

The following image file with `.fancyspectra.pdf` suffix,



Spectratype with additional detalization. Most abundant clonotypes are explicitly shown.

1.7.5 PlotFancyVJUsage

Plots a circos-style V-J usage plot displaying the frequency of various V-J junctions.

Command line usage

```
$VDJTOOLS PlotFancyVJUsage [options] sample.txt output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-u`	`--unweighted`		Instead of computing read frequency, will compute the number of unique clonotypes with specific V-J junctions
`-h`	`--help`		Display help message

Tabular output

A matrix with rows corresponding to different J segments and columns corresponding to different V segments. Each cells contains the frequency of a give V-J junction. The file has `.fancyvj.[unwt or wt depending on -u parameter].txt` suffix.

Graphical output

An image having the same name as the output table, with the exception of `.pdf` extension, is generated. The plot is built using circlize R package.

1.7.6 PlotSpectratypeV

Plots a detailed spectratype containing additional info displays CDR3 length distribution for clonotypes from top N Variable segment families. This plot is useful to detect type 1 and type 2 repertoire biases, that could arise under pathological conditions.

Command line usage

```
$VDJTOOLS PlotSpectratypeV [options] sample.txt output_prefix
```

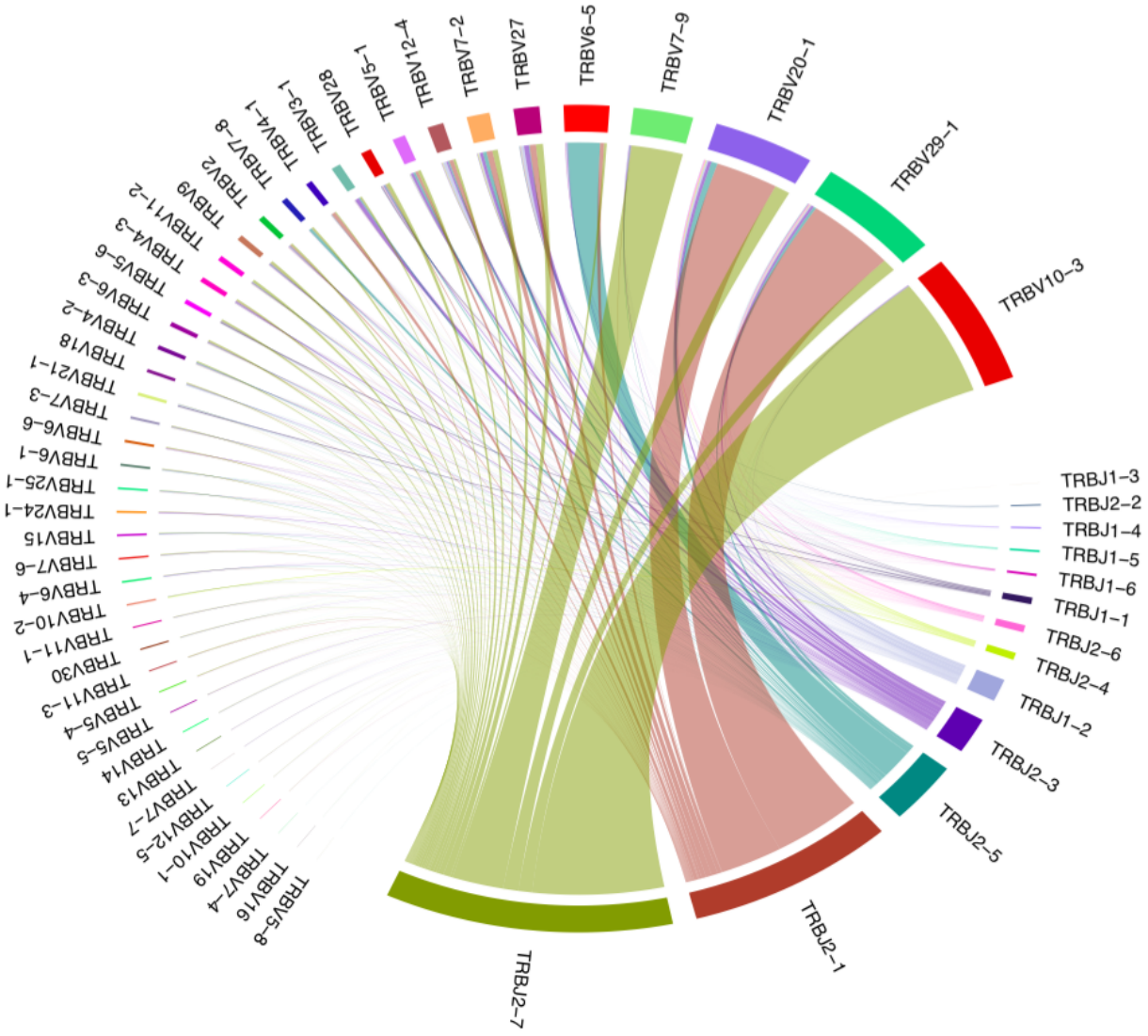


Fig. 1: **V-J junction circos plot for a single sample.** Arcs correspond to different V and J segments, scaled to their frequency in sample. Ribbons represent V-J pairings and their size is scaled to the pairing frequency (weighted in present case).

Parameters

Shorthand	Long name	Argument	Description
`-t`	`--top`	int	Number of top (by frequency) V segments to visualize. Should not exceed 12 default is 12
`-u`	`--unweighted`		Instead of counting read frequency, will count the number of unique clonotypes
`-h`	`--help`		Display help message

Tabular output

Following table with `.spectraV.[unwt or wt depending on -u parameter].txt` prefix is generated,

Column	Description
Len	Length of CDR3 nucleotide sequence
Other	Frequency of clonotypes with a given CDR3 length, having V segments other than the top N
Segment#N, e.g. TRBV10-1	Frequency of clonotypes with a given V segment at the corresponding CDR3 length
Segment#N-1	...
...	...

Graphical output

The following image file with `.spectraV.[unwt or wt depending on -u parameter].pdf` suffix,

Stacked spectratypes by Variable segment for a single sample. Most frequent Variable segments are highlighted.

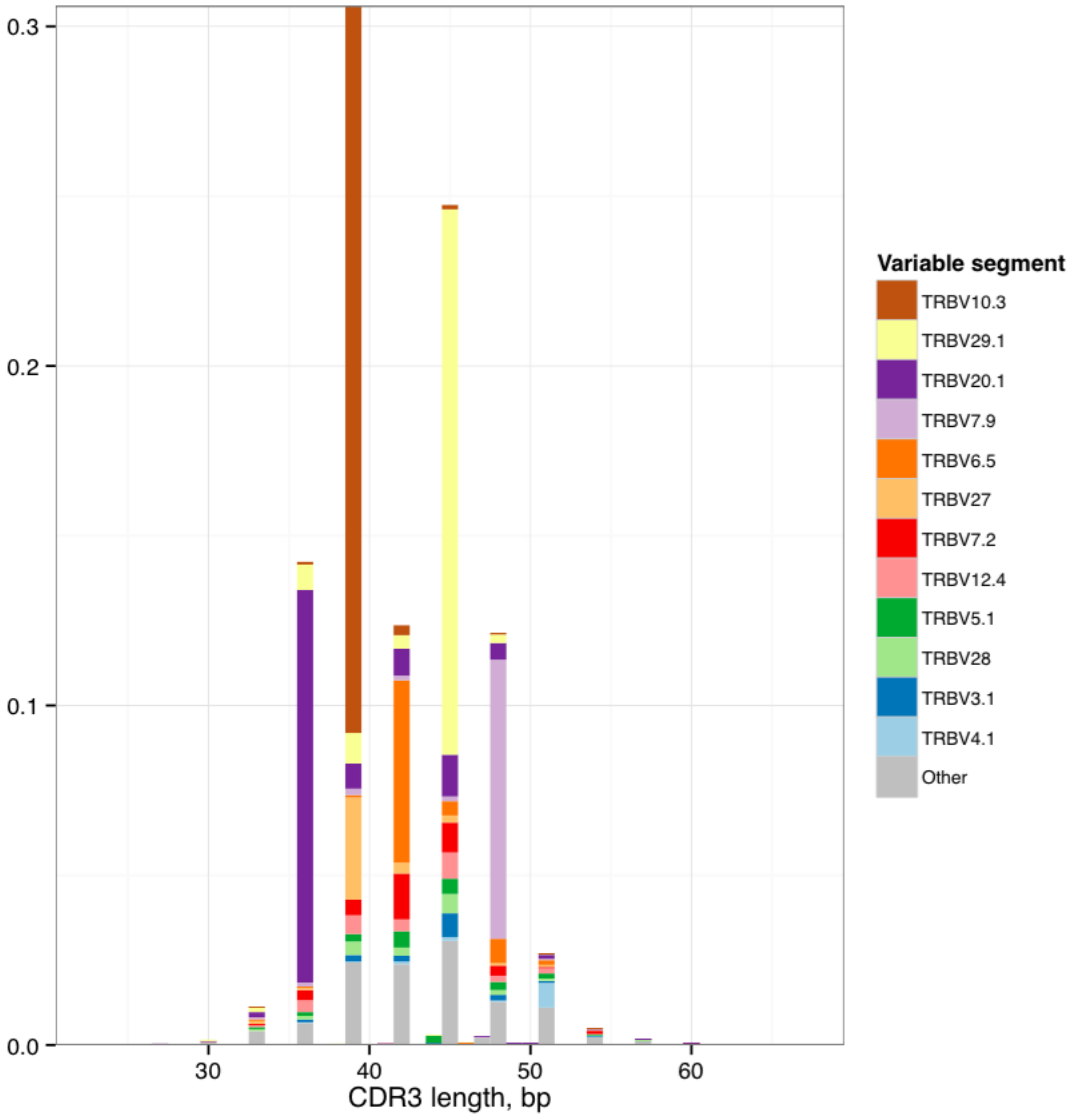
1.8 Diversity estimation

Note: Application of routines from this section is described in the following tutorial.

1.8.1 PlotQuantileStats

Plots a three-layer donut chart to visualize the repertoire clonality.

- First layer ("set") includes the frequency of singleton ("1", met once), doubleton ("2", met twice) and high-order ("3+", met three or more times) clonotypes. Singleton and doubleton frequency is an important factor in estimating the total repertoire diversity, e.g. Chao1 diversity estimator (see Colwell et al). We have also recently shown that in whole blood samples, singletons have very nice correlation with the number of naive T-cells, which are the backbone of immune repertoire diversity.
- The second layer ("quantile"), displays the abundance of top 20% ("Q1"), next 20% ("Q2"), ... (up to "Q5") clonotypes for clonotypes from "3+" set. In our experience this quantile plot is a simple and efficient way to display repertoire clonality.
- The last layer ("top") displays the individual abundances of top N clonotypes.


0.3
0.2
0.1
0.0
30
40
50
60
CDR3 length, bp
Variable segment
TRBV10.3
TRBV29.1
TRBV20.1
TRBV7.9
TRBV6.5
TRBV27
TRBV7.2
TRBV12.4
TRBV5.1
TRBV28
TRBV3.1
TRBV4.1
Other

Command line usage

```
java -Xmx4G -jar vdjtools.jar PlotQuantileStats [options] sample.txt output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-t`	`--top`	int	Number of top clonotypes to visualize. Should not exceed 10, default is 5
`-h`	`--help`		Display help message

Tabular output

Following table with `.qstat.txt` prefix is generated,

Column	Description
Type	Detalization level: `set`, `quantile` or `top`
Name	Variable name: "1", "Q1", "CASSLAPGATNEKLFF", etc
Value	Corresponding relative abundance

Graphical output

Following plot with `.qstat.pdf` prefix is generated,

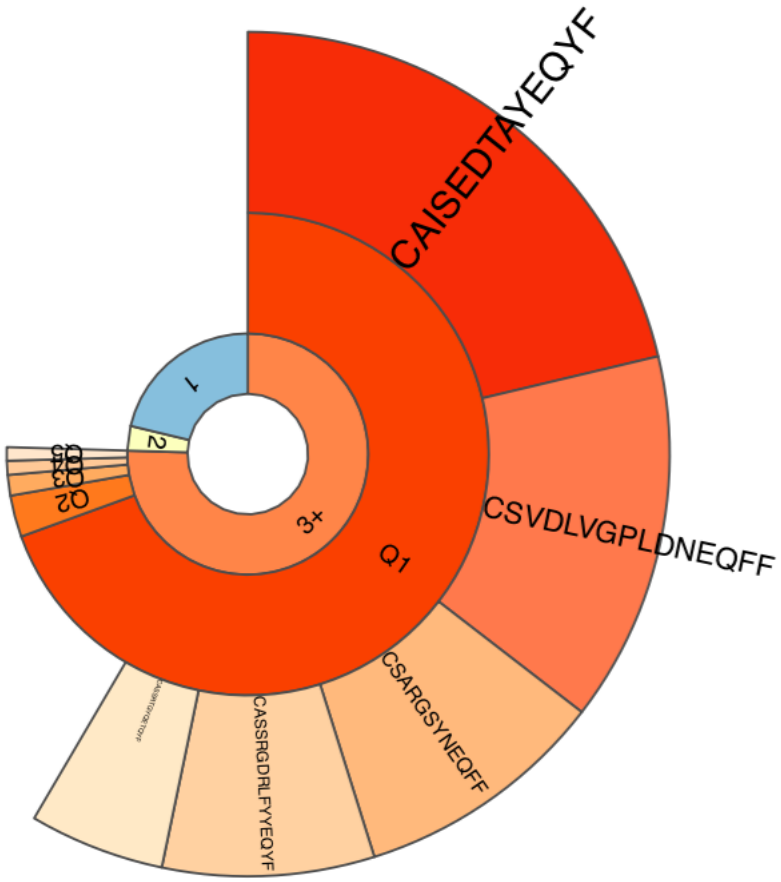
Sample clonality plot. See above for the description of plot structure.

1.8.2 RarefactionPlot

Plots rarefaction curves for specified list of samples, that is, the dependencies between sample diversity and sample size. Those curves are interpolated from 0 to the current sample size and then extrapolated up to the size of the largest of samples, allowing comparison of diversity estimates. Interpolation and extrapolation are based on multinomial models, see Colwell et al for details.

Command line usage

```
java -Xmx4G -jar vdjtools.jar RarefactionPlot \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```


CAISEDTAYEQYF
CSVDLVGPLDNEQFF
CSARGSYNEQFF
CASSRGDRLFYYEQYF
1
2
3+
Q1
Q2

Parameters

Short-hand	Long name	Ar-gu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-i`	`--intersect-type`	string	Set the intersection type used to collapse clonotypes before computing diversity. Defaults to `strict` (don't collapse at all). See *Common parameters*
`-s`	`--steps`	inte-ger	Set the total number of points in the rarefaction curve, default is `101`
`-f`	`--factor`	string	Specifies plotting factor. See *Common parameters*
`-n`	`--numeric`		Specifies if plotting factor is numeric. See *Common parameters*
`-l`	`--label`	string	Specifies label used for plotting. See *Common parameters*
	`--wide-plot`		Set wide plotting area
	`--label-exact`		If set to true, will position sample labels exactly at observed samle size, will use the extrapolated sample size otherwise
`-h`	`--help`		Display help message

Tabular output

The following table with `rarefaction.[intersection type shorthand].txt` is generated:

Column	Definition
sample_id	Sample unique identifier
...	Sample metadata columns, see *Metadata* section
x	Subsample size, reads
mean	Mean diversity at given size
ciL	Lower bound of 95% confidence interval
ciU	Upper bound of 95% confidence interval
type	Data point type: `0=interpolation`, `1=exact`, `2=extrapolation`

Graphical output

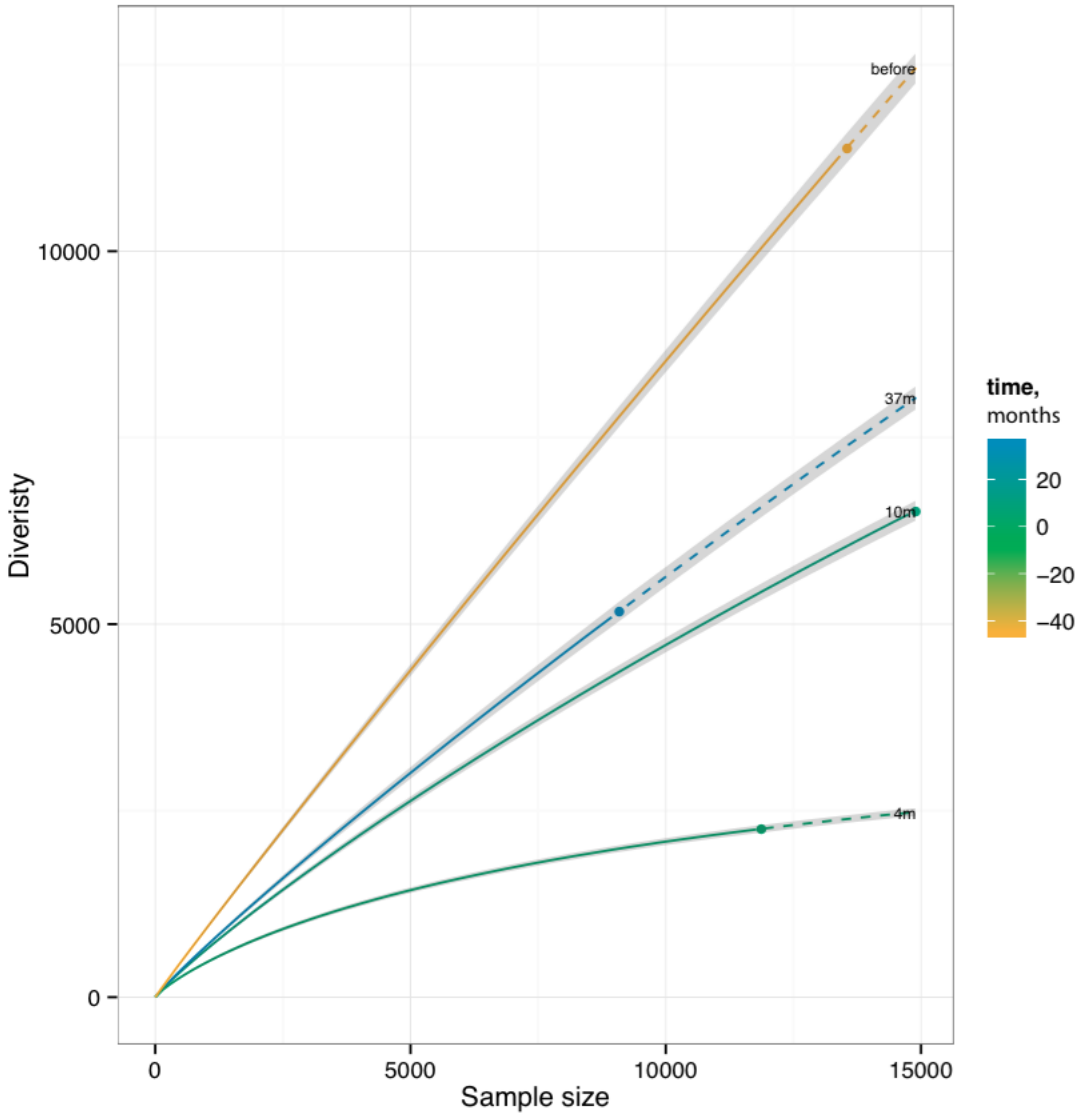
A figure with the same suffix as output table and `.pdf` extension is provided.

Rarefaction plot. Solid and dashed lines mark interpolated and extrapolated regions of rarefaction curves respectively, points mark exact sample size and diversity. Shaded areas mark 95% confidence intervals.

1.8.3 CalcDiversityStats

Computes a set of diversity statistics, including

- Observed diversity, the total number of clonotypes in a sample
- Lower bound total diversity (LBTD) estimates
  - Chao estimate (denoted *chao1*)
  - Efron-Thisted estimate
- Diversity indices


before
37m
10m
4m
10000
5000
0
Diveristy
0
5000
10000
15000
Sample size
time,
months
20
0
−20
−40

  - Shannon-Wiener index. The exponent of clonotype frequency distribution entropy is returned.
  - Normalized Shannon-Wiener index. Normalized (divided by `log[number of clonotypes]`) entropy of clonotype frequency distribution. Note that plain entropy is returned, not its exponent.
  - Inverse Simpson index
- Extrapolated Chao diversity estimate, denoted *chaoE* here.
- The d50 index, a recently developed immune diversity estimate

Diversity estimates are computed in two modes: using original data and via several re-sampling steps (usually down-sampling to the size of smallest dataset).

- The estimates computed on original data could be biased by uneven sampling depth (sample size), of those only `chaoE` is properly normalized to be compared between samples. While not good for between-sample comparison, the LBTD estimates provided for original data are most useful for studying the fundamental properties of repertoires under study, i.e. to answer the question how large the repertoire diversity of an entire organism could be.
- Estimates computed using re-sampling are useful for between-sample comparison, e.g. we have successfully used the re-sampled (normalized) observed diversity to measure the repertoire aging trends (see this paper).

Hint: In our recent experience the observed diversity and LBTD estimates computed on re-sampled data provide best results for between-sample comparisons.

Command line usage

```
java -Xmx4G -jar vdjtools.jar CalcDiversityStats \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-i`	`--intersection-type`	string	Set the intersection type used to collapse clonotypes before computing diversity. Defaults to `strict` (don't collapse at all). See *Common parameters*
`-x`	`--downsample-to`	integer	Set the sample size to interpolate the diversity estimates via resampling. Default = size of smallest sample. Applies to diversity estimates stored in `.resampled.txt` table
`-X`	`--extrapolate-to`	integer	Set the sample size to extrapolate the diversity estimates. Default = size of largest sample. Currently, only applies to `chaoE` diversity estimate.
	`--resample-iter`	integer	Number of resamples for corresponding estimator. Default = 3
`-h`	`--help`		Display help message

Tabular output

Two tables with `diversity.[intersection type shorthand].txt` and `diversity.[intersection type shorthand].resampled.txt` are generated, containing diversity estimates computed on original and down-sampled datasets respectively.

Note that `chaoE` estimate is only present in the table generated for original samples. Both tables contain means and standard deviations of diversity estimates. Also note that standard deviation and mean values for down-sampled datasets are computed based on N=3 re-samples.

Here is an example column layout, similar between both output tables:

Column	Definition
sample_id	Sample unique identifier
...	Sample metadata columns, see *Metadata* section
reads	Number of reads in the sample
diversity	Diversity of the original sample (after collapsing to unique clonotypes according to `-i` parameter)
extrapolate_reads / resample_reads	The reads used to extrapolate or re-sample in order to compute present diversity estiamtes
<*name*>_mean	Mean value of the diversity estimate <*name*>
<*name*>_std	Standard deviation of the diversity estimate <*name*>
...	

Graphical output

none

1.9 Repertoire overlap analysis

1.9.1 OverlapPair

Performs a comprehensive analysis of clonotype sharing for a pair of samples.

Command line usage

```
$VDJTOOLS OverlapPair [options] sample1.txt sample2.txt output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-i`	`--intersect-type`	string	Sample intersection rule. Defaults to `strict`. See *Common parameters*
`-t`	`--top`	int	Number of top clonotypes to visualize explicitly on stack are plot and provide in the collapsed joint table. Should not exceed 100, default is 20
`-p`	`--plot`		Turns on plotting. See *Common parameters*
	`--plot-area-v2`		Alternative plotting mode, clonotype CDR3 sequences are shown at plot sides and connected to corresponding areas with lines.
`-h`	`--help`		Display help message

Tabular output

Two joint clonotype abundance tables with `paired.[intersection type shorthand].table.txt` and `paired.[intersection type shorthand].table.collapsed.txt` suffices are generated. Tables are

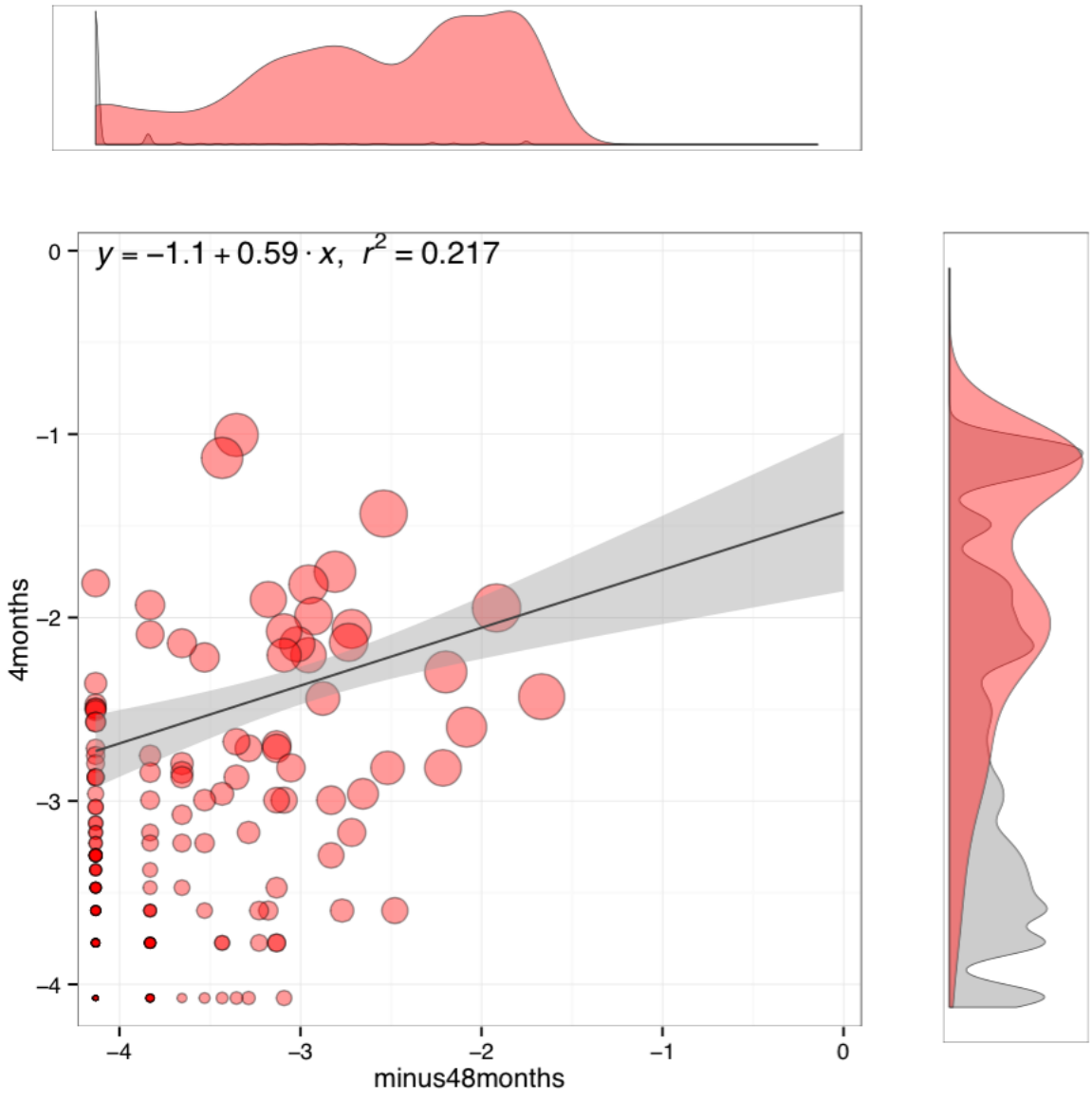
written in *VDJtools format*. Collapsed table contains rows corresponding to top N clonotypes and summary abundances for non-overlapping and hidden clonotypes.

See *Joint clonotype abundance table structure* for a detailed description of table columns.

A summary table (`paired.[intersection type shorthand].summary.txt` suffix) containing information on sample overlap size, etc, is also provided. See tabular output in *CalcPairwiseDistances* section below for details.

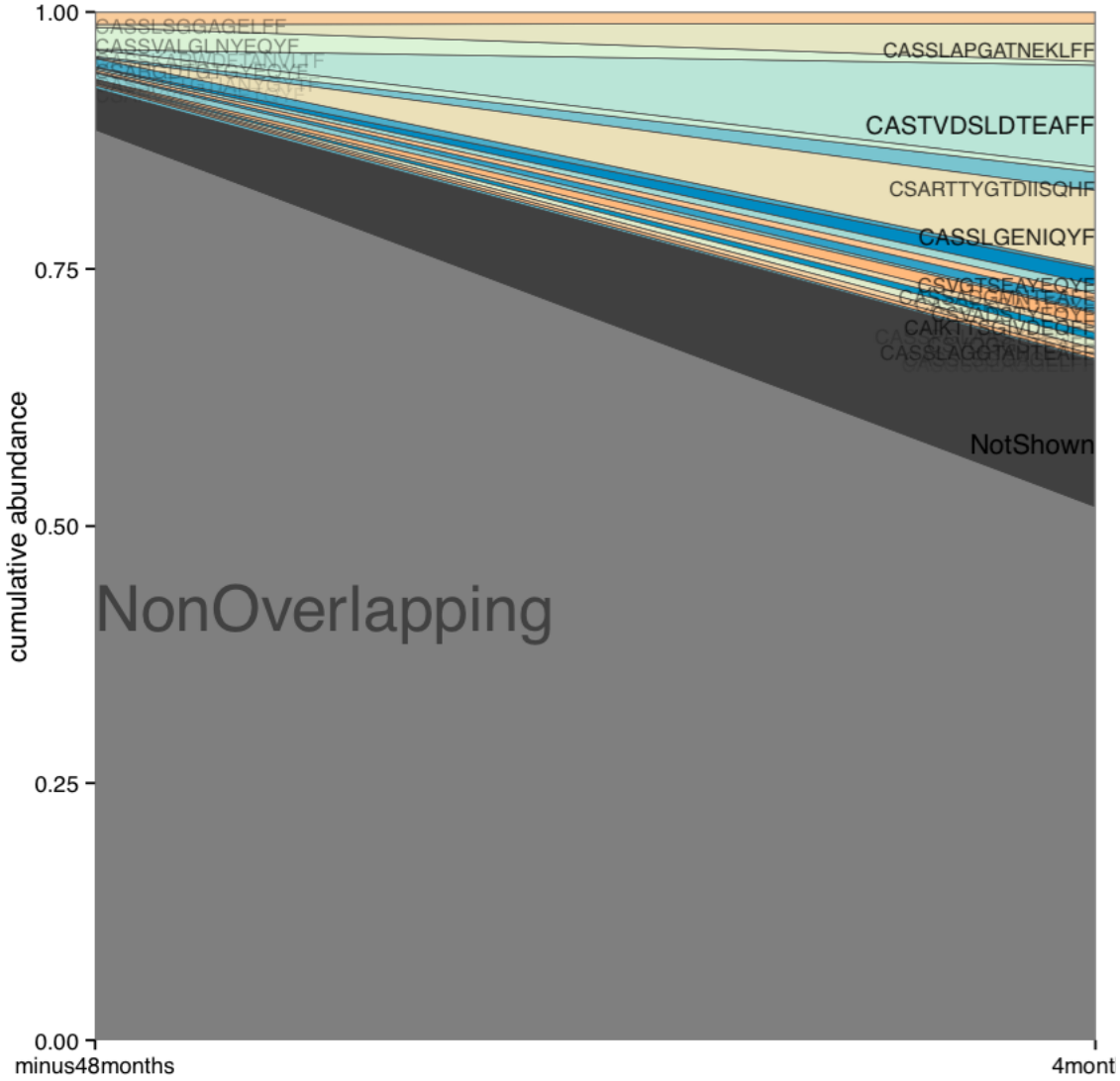
Graphical output

A composite scatterplot plot having `paired.[intersection type shorthand].scatter.pdf` suffix is generated. The second plot file with `.paired.[intersection type shorthand].table.collapsed.pdf` suffix contains a clonotype stack area plot.



Clonotype scatterplot. Main frame contains a scatterplot of clonotype abundances (overlapping clonotypes only) and a linear regression. Point size is scaled to the geometric mean of clonotype frequency in both samples. Scatterplot axes

represent log10 clonotype frequencies in each sample. Two marginal histograms show the overlapping (red) and total clonotype (grey) abundance distributions in corresponding sample. Histograms are weighted by clonotype abundance, i.e. they display read distribution by clonotype size.



Shared clonotype abundance plot. Plot shows details for top 20 clonotypes shared between samples, as well as collapsed ("NotShown") and non-overlapping ("NonOverlapping") clonotypes. Clonotype CDR3 amino acid sequence is plotted against the sample where the clonotype reaches maximum abundance.

1.9.2 CalcPairwiseDistances

Performs an all-versus-all pairwise overlap for a list of samples and computes a set of repertoire similarity measures. At least 3 samples should be provided. Note that this is one of most the memory-demanding routines, as it will load all samples into memory at once (unless used with `--low-mem` option).

Repertoire similarity measures include

- Pearson correlation of clonotype frequencies, restricted only to the overlapping clonotypes

$$R_{ij} = \frac{\sum_{k=1}^{N} \left(\phi_{ik} - \bar{\phi_i}\right) \left(\phi_{jk} - \bar{\phi_j}\right)}{\sqrt{\sum_{k=1}^{N} \left(\phi_{ik} - \bar{\phi_i}\right)^2 \sum_{k=1}^{N} \left(\phi_{jk} - \bar{\phi_j}\right)^2}}$$

  where $k = 1..N$ are the indices of overlapping clonotypes, ϕ_{ik} is the frequency of clonotype k in sample i and $\bar{\phi_i}$ is the average frequency of overlapping clonotypes in sample i.

- Relative overlap diversity, computed with the following normalization

$$D_{ij} = \frac{d_{ij}}{d_i d_j}$$

  where d_{ij} is the number of clonotypes present in both samples and d_i is the diversity of sample i. See this paper for the rationale behind normalization.

- Geometric mean of relative overlap frequencies

$$F_{ij} = \sqrt{f_{ij} f_{ji}}$$

  where $f_{ij} = \sum_{k=1}^{N} \phi_{ik}$ is the total frequency of clonotypes that overlap between samples i and j in sample i.

- lonotype-wise sum of geometric mean frequencies

$$F2_{ij} = \sum_{k=1}^{N} \sqrt{\phi_{ik} \phi_{jk}}$$

  Note that this measure performs similar to F and provides slightly more robust results in case cross-sample contamination is present.

- Jensen-Shannon divergence between Variable segment usage profiles (will be moved to *CalcSegmentUsage* in near future).
- Jaccard index.
- Morisita-Horm index.

ClusterSamples routine can be additionally run for CalcPairwiseDistances results.

Command line usage

```
$VDJTOOLS CalcPairwiseDistances \
[options] [sample1.txt sample2.txt sample3.txt ... if -m is not specified] output_
↪prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-i`	`--intersect-type`	string	Sample intersection rule. Defaults to `aa`. See *Common parameters*
	`--low-mem`		Low memory mode, will keep only a pair of samples in memory during execution, but run much slower.
`-p`	`--plot`		Turns on plotting. See *Common parameters*
`-h`	`--help`		Display help message

Tabular output

A table suffixed `intersect.batch.[intersection type shorthand].summary.txt` with a comprehensive information on sample pair intersections is generated. This table is non-redundant: it contains `N * (N - 1) / 2` rows corresponding to upper diagonal of matrix of possible pairs `(i,j)`. Table layout is given below in three parts.

General info

Column	Description
1_sample_id	First sample unique identifier
2_sample_id	Second sample unique identifier
div1	Total number of clonotypes in the first sample after identical clonotypes are collapsed based on intersection type `-i`
div2	Same as above, second sample
div12	Number of overlapping clonotypes
div21	Same as above
count1	Total number of reads in the first sample
count2	...
count12	For clonotypes **overlapping** between two samples: total number of reads they have in the **first** sample
count21	...
freq1	Total clonotype relative abundance for the first sample (should be 1.0 if sample is unaltered)
freq2	...
freq12	For clonotypes **overlapping** between two samples: their sum of relative abundances in the **first** sample
freq21	...

Overlap metrics

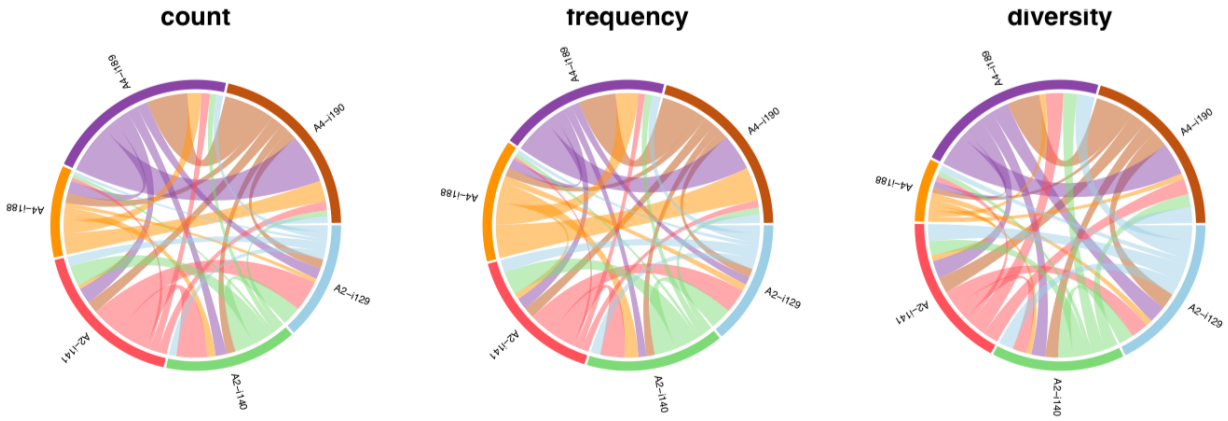
Column	Description
R	Pearson correlation
D	Relative overlap diversity
F	Geometric mean of relative overlap frequencies
F2	Sum of geometric means of overlapping clonotype frequencies.
vJSD	Jensen-Shannon divergence of Variable segment usage distributions
vjJSD	<*experimental*>
vj2JSD	<*experimental*>
sJSD	<*experimental*>
Jaccard	Jaccard index
MorisitaHorn	Morisita-Horn index

Sample metadata

Column	Description
1_...	First sample metadata columns. See *Metadata* section
2_...	Second sample metadata columns

Graphical output

Circos plots showing pairwise overlap are stored in a file suffixed `intersect.batch.[intersection type shorthand].summary.pdf`.



Pairwise overlap circos plot. Count, frequency and diversity panels correspond to the read count, frequency (both non-symmetric) and the total number of clonotypes that are shared between samples. Pairwise overlaps are stacked, i.e. segment arc length is not equal to sample size.

1.9.3 ClusterSamples

This routine provides additional cluster analysis (hierarchical clustering), multi-dimensional scaling (MDS) and plotting for *CalcPairwiseDistances* output.

Note that this routine requires the following parameter setting:

- Input file prefix (`input_prefix`) is set to the same value as the output prefix of *CalcPairwiseDistances*
- The `-i` argument setting is the same as in *CalcPairwiseDistances*

Command line usage

```
$VDJTOOLS ClusterSamples \
[options] input_prefix [output_prefix]
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-e`	`--measure`	string	Sample overlap metric, see **Overlap metrics** section of *CalcPairwiseDistances* tabular output for allowed values. Defaults to `F`
`-i`	`--intersect-type`	string	Intersection type, defaults to `aa`. See *Common parameters*
`-f`	`--factor`	string	Specifies metadata column with plotting factor (is used to color for sample labels and figure legend). See *Common parameters*
`-n`	`--numeric`		Specifies if plotting factor is continuous. See *Common parameters*
`-l`	`--label`	string	Specifies metadata column with sample labelslabel . See *Common parameters*
`-h`	`--help`		Display help message
`-p`	`--plot`		Turns on plotting. See *Common parameters*

Tabular output

Two output files are generated:

- Table suffixed `mds.[value of -i argument].[value of -e argument].txt` that contains coordinates of samples computed using multi-dimensional scaling (MDS), i.e. the coordinates of samples projected to a 2D plane in a manner that pairwise sample distances are preserved.
- A file in Newick format suffixed `hc.[value of -i argument].[value of -e argument].newick` is generated that contains sample dendrogram produced by hierarchical clustering.

Note: Hierarchical clustering and MDS are performed using `hclust()` and `isoMDS()` (MASS package) R functions. Default parameters are used for those algorithms.

Distances are scaled as `-log10(.)` and `(1-.)/2` for relative overlap and correlation metrics respectively; in case of Jensen-Shannon divergence, Jaccard and Morisita-Horn indices no scaling is performed.

Graphical output

Hierarchical clustering plot is stored in a file suffixed `hc.[value of -i argument].[value of -e argument].pdf`. MDS plot is stored in a file with `mds.[value of -i argument].[value of -e argument].pdf` suffix.

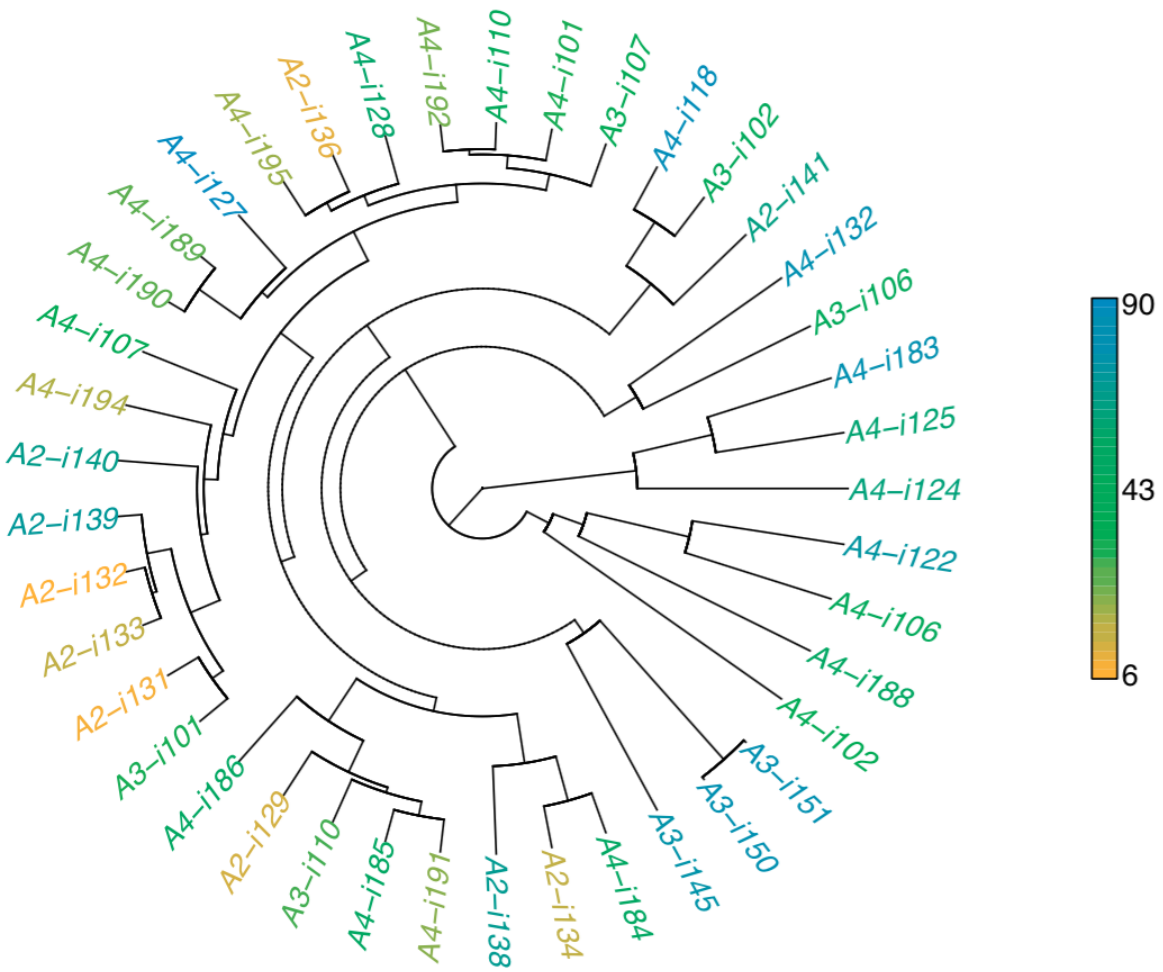
Hierarchical clustering. Dendrogram of samples, branch length shows the distance between repertoires. Node colors correspond to factor value, continuous scale is used in present case (`-n -f age` argument).

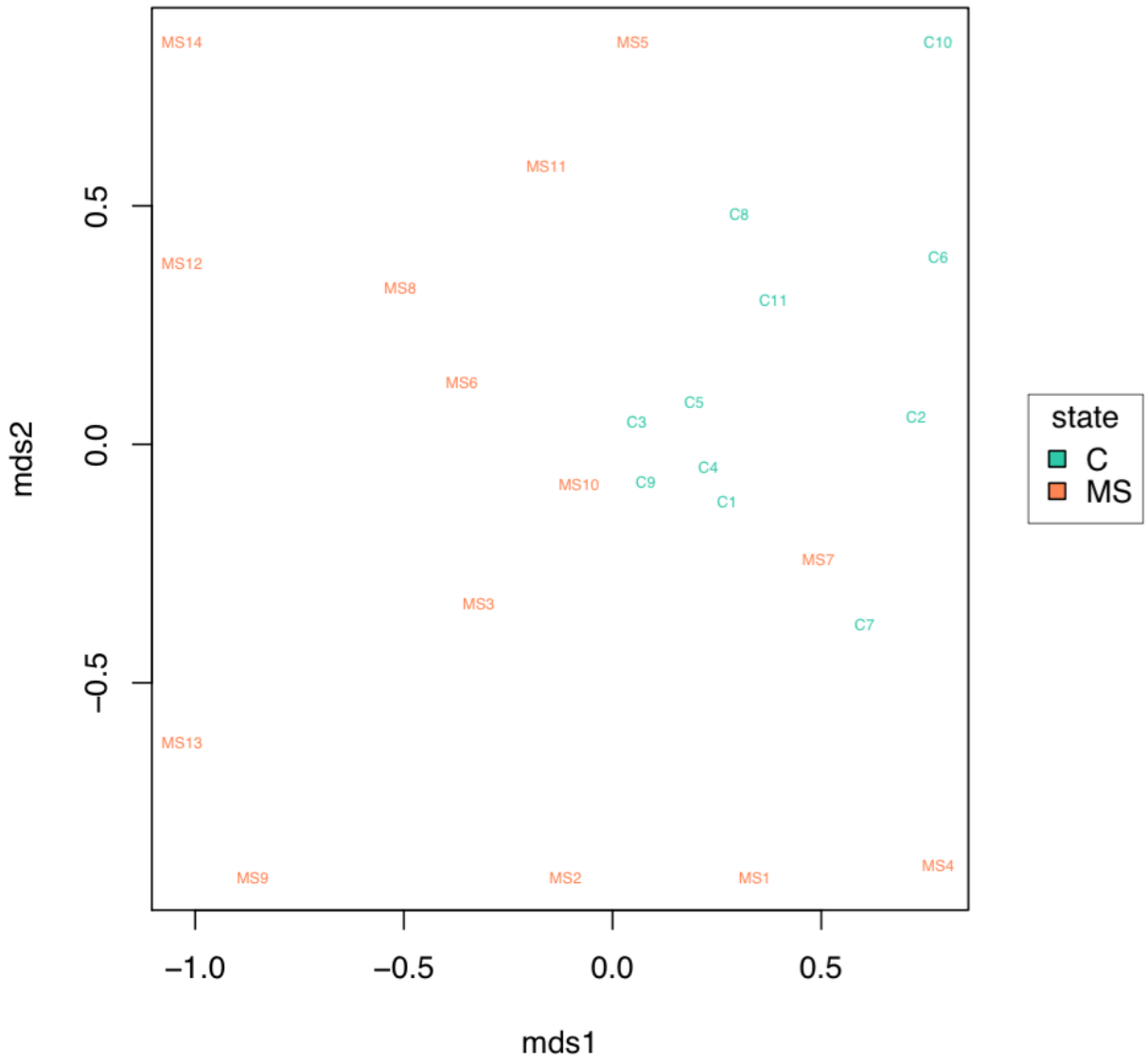
MDS plot. A scatterplot of samples. Euclidean distance between points reflects the distance between repertoires. Points are colored by factor value.

1.9.4 TestClusters

This routine allows to test whether a given factor influences repertoire clustering. It assesses compactness of samples that have the same factor level and separation between samples with distinct factor levels for the factor specified in *ClusterSamples*.

Performs post-hoc permutation testing based on MDS coordinates generated by *ClusterSamples* routine. Can only be performed if a discrete factor (`-f`) was specified in *ClusterSamples*.


A4–i192
A4–i110
A4–i101
A3–i107
A4–i118
A3–i102
A2–i141
A4–i132
A3–i106
A4–i183
A4–i125
A4–i124
A4–i122
A4–i106
A4–i188
A4–i102
A3–i151
A3–i150
A3–i145
A4–i184
A2–i134
A2–i138
A4–i191
A4–i185
A3–i110
A2–i129
A4–i186
A3–i101
A2–i131
A2–i133
A2–i132
A2–i139
A4–i140
A4–i194
A4–i107
A4–i190
A4–i189
A4–i127
A4–i195
A2–i136
A4–i128
90
43
6


MS14
MS5
C10
MS11
C8
MS12
C6
MS8
C11
MS6
C5
C3
C2
MS10
C9
C4
C1
MS7
MS3
C7
MS13
MS9
MS2
MS1
MS4
0.5
0.0
−0.5
mds2
−1.0
−0.5
0.0
0.5
mds1
state
C
MS

Note that this routine requires the following parameter setting:

- Input file prefix (`input_prefix`) is set to the same value as the output prefix of *ClusterSamples*
- The `-i` and `-e` argument setting is the same as in *ClusterSamples*

Command line usage

```
$VDJTOOLS TestClusters \
[options] input_prefix [output_prefix]
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-e`	`--measure`	string	Sample overlap metric, see **Overlap metrics** section of *CalcPairwiseDistances* tabular output for allowed values. Defaults to `F`
`-i`	`--intersect-type`	string	Intersection type, defaults to `aa`. See *Common parameters*

Tabular output

none

Graphical output

Permutation summary plot is generated having the `perms.[value of -i argument].[value of -e argument].pdf` suffix.

Testing compactness and separation of sample clustering for a given factor. Average repertoire similarity values for sample pairs in which both samples have the same (within panel) and different (between panel) factor levels. Each row correspond to a specific factor level. Red lines show observed values, histograms correspond to values generated by randomly permuting factor levels. Numbers near red lines indicate P-values for n=10000 permutations.

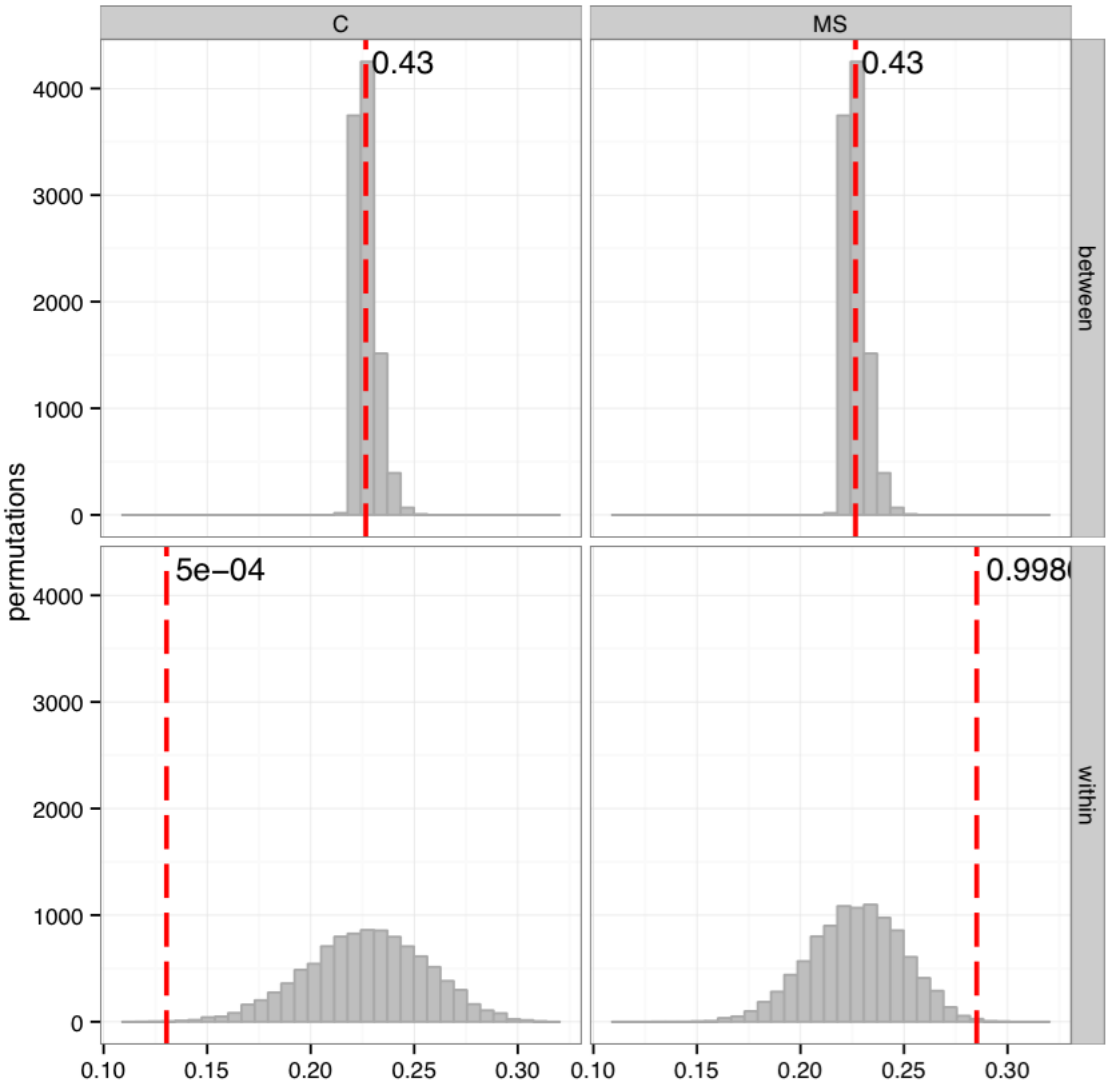
1.9.5 TrackClonotypes

This routine performs an all-vs-all intersection between an ordered list of samples for clonotype tracking purposes. User can specify sample which clonotypes will be traced, e.g. the pre-therapy sample.

Command line usage

```
$VDJTOOLS TrackClonotypes \
[options] [sample1.txt sample2.txt sample3.txt ... if -m is not specified] output_
↪prefix
```

Parameters:


C
MS
0.43
0.43
5e−04
0.998
between
within
permutations
4000
3000
2000
1000
0
0.10
0.15
0.20
0.25
0.30

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See See *Common parameters*
`-i`	`--intersect-type`	string	Sample intersection rule. Defaults to `strict`. See *Common parameters*
`-f`	`--factor`	string	Specifies factor that should be treated as `time` variable. Factor values should be numeric. If such column not set, time points are taken either from values provided with `-s` argument or sample order. See *Common parameters*
`-x`	`--track-sample`	integer	A zero-based index of time point to track. If not provided, will consider all clonotypes that were detected in 2+ samples
`-s`	`--sequence`	`[t1, t2,...]`	Time point sequence. Unused if `-m` is specified. If not specified, either `time` column values from metadata, or sample indexes (as in command line) are used.
`-t`	`--top`	integer	Number of top clonotypes to visualize explicitly on stack are plot and provide in the collapsed joint table. Should not exceed 100, default is 200
`-p`	`--plot`		Turns on plotting. See *Common parameters*
`-c`	`--compress`		Compressed output for clonotype table. See *Common parameters*
`-h`	`--help`		Display help message

Tabular output

Summary table suffixed `sequential.[value of -i argument].summary.txt` is created with the following columns.

Column	Description
1_sample_id	First sample unique identifier
2_sample_id	Second sample unique identifier
value	Value of the intersection metric
metric	Metric type: `diversity`, `frequency` or `count`. Metrics correspond to the number of unique clonotypes, total frequency and total read count for clonotypes overlapping between first and second sample. In case tracking is on (`-x`), only clonotypes present in tracked sample are counted.
1_time	Time value for the first sample
2_time	Time value for the second sample
1_...	First sample metadata columns. See *Metadata* section
2_...	Second sample metadata columns

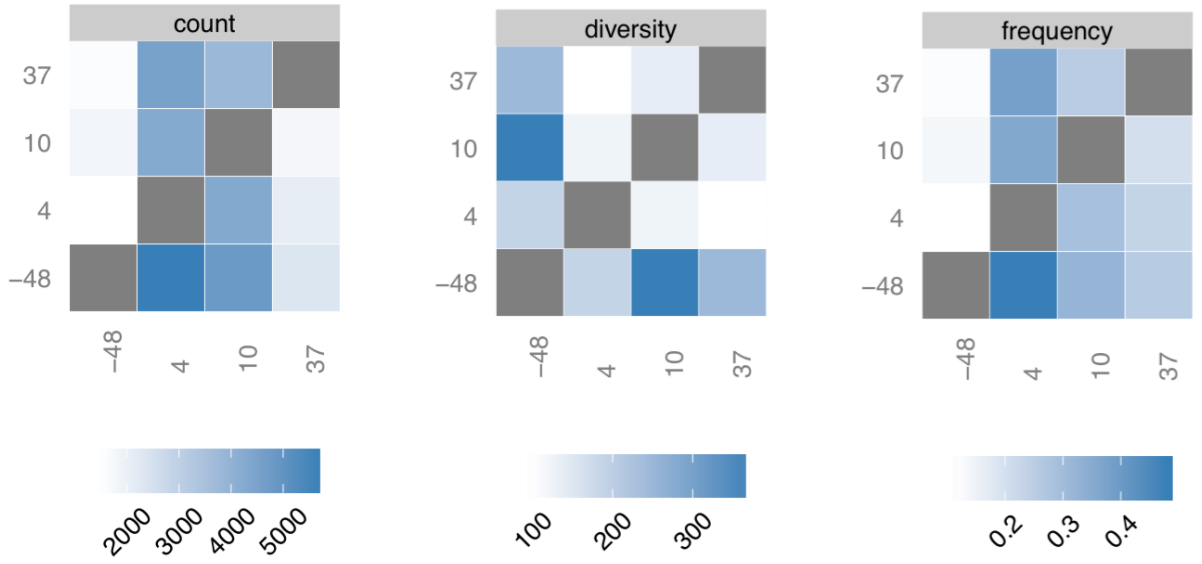
Two joint clonotype abundance tables with `sequential.[intersection type shorthand].table.txt` and `sequential.[intersection type shorthand].table.collapsed.txt` suffices are generated. The latter contains top `-t` clonotypes, with two additional rows containing summary count and frequency for non-overlapping and collapsed clonotypes.

See *Joint clonotype abundance table structure* for a detailed description of table columns.

Graphical output

Summary table is visualized in a plot file suffixed `sequential.[value of -i argument].summary.pdf`. A plot file with `.sequential.[value of -i argument].stackplot.pdf` suffix contains a clonotype abundance stack area plot. The same is also visualized using a heatmap in a file with `.sequential.[value of -i argument].heatplot.pdf`).

Clonotype tracking summary. Count, frequency and diversity panels correspond to the read count, frequency (both



non-symmetric) and the total number of clonotypes that are shared between samples. Rows and columns of each matrix are sorted according to time point sequence.

Clonotype tracking stackplot. Contains detailed profiles for top `-t` clonotypes, as well as collapsed ("NotShown") and non-overlapping ("NonOverlapping") clonotypes. Clonotype CDR3 amino acid sequence is plotted against the sample where the clonotype reaches maximum abundance. Clonotypes are colored by the peak position of their abundance profile.

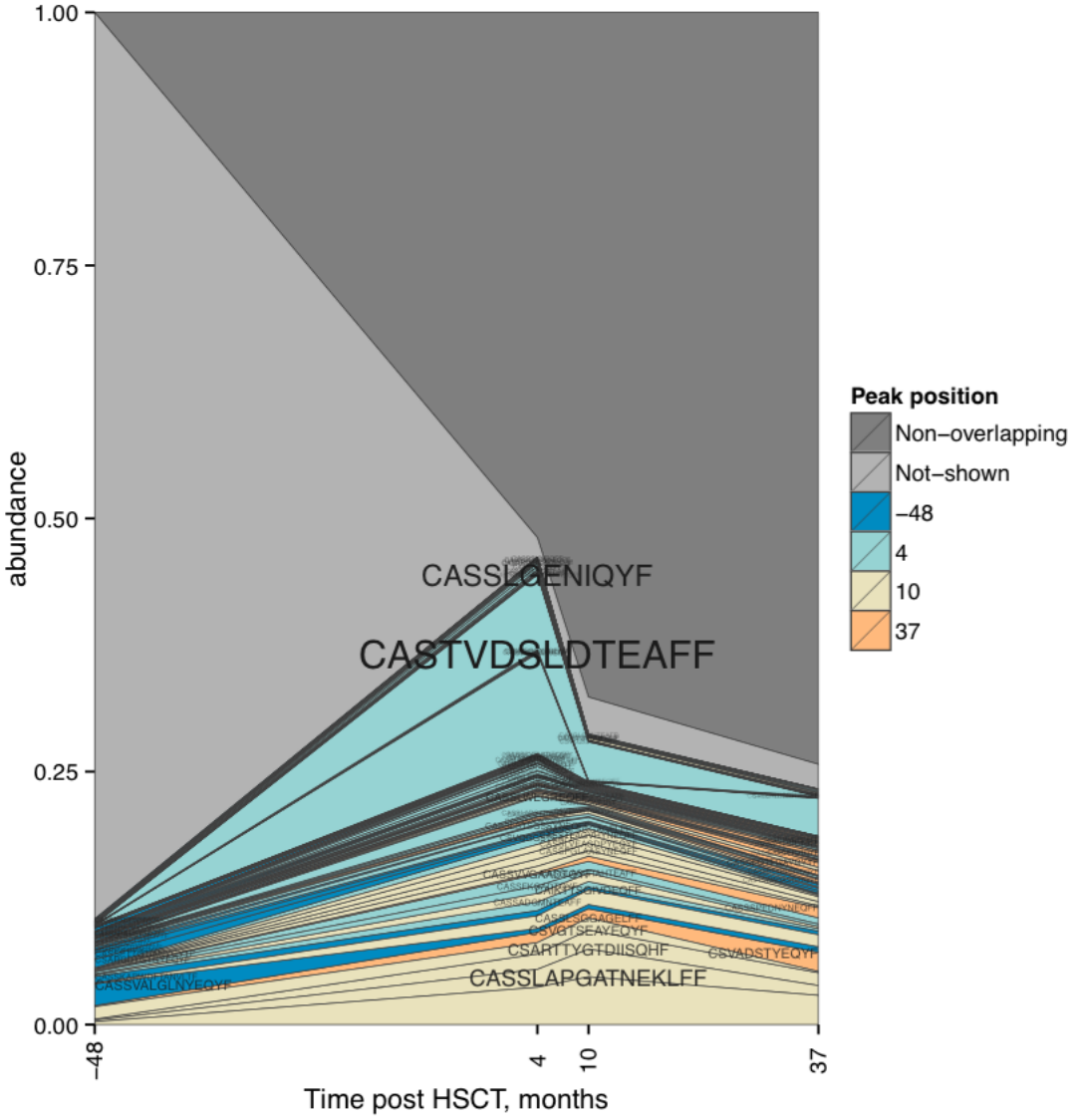
Clonotype tracking heatmap. Shows a heatmap for top `-t` joint clonotype abundances.

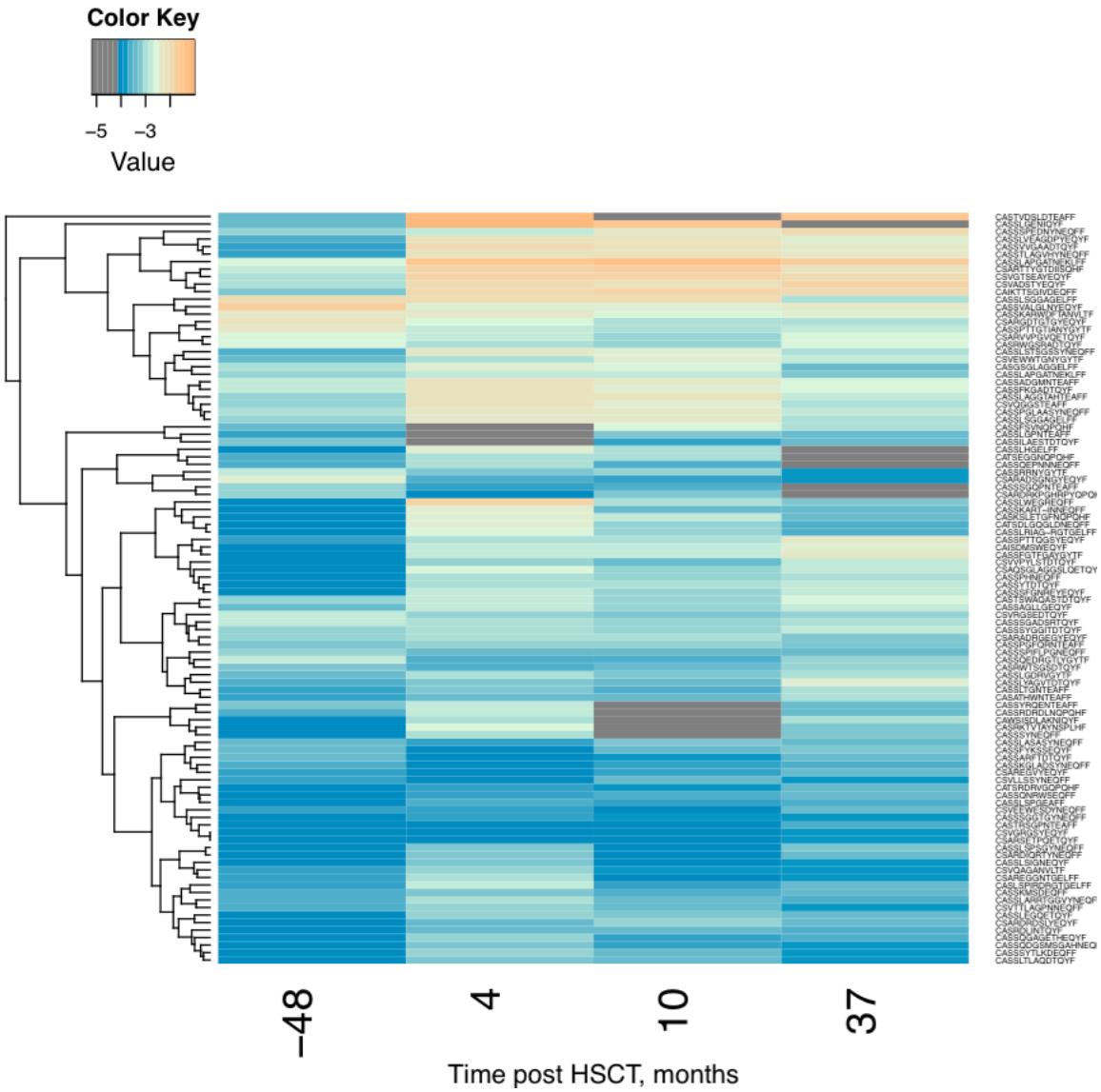
1.10 Pre-processing

Note: Most of routines specified in this section will output processed clonotype tables and re-normalize individual clonotype frequencies by dividing their read count by the total read count in resulting (filtered/processed) sample. For some of the routines this behavior can be disabled with `--save-freqs` option. In this case original clonotype frequencies will be carried over from input samples and they will likely not sum to `1.0` in the resulting clonotype table.

1.10.1 Correct

Performs frequency-based correction to eliminate erroneous clonotypes. Searches the sample for clonotype pairs that differ by one, two ... (up to specified depth) mismatches. In case the ratio of smallest to largest clonotype sizes is lower than the threshold specified as `ratio ^ number_of_mismatches` correction is performed. Largest clonotype in pair increases its size by the read count of the smaller one and the smaller one is discarded. Note that the original sample is not changed during correction, so all comparisons are performed with original count values and erroneous clonotypes are only removed after search procedure is finished. It is also possible to restrict correction to clonotypes with identical V/J segments using `-a` option.


1.00
0.75
0.50
0.25
0.00
abundance
CASSLGENIQYF
CASTVDSLDTEAFF
CSVGTSEAYEQYF
CSARTTYGTDIISQHF
CSVADSTYEQYF
CASSLAPGATNEKLFF
CASSVALGLNYEQYF
−48
4
10
37
Time post HSCT, months
Peak position
Non−overlapping
Not−shown
−48
4
10
37


Color Key
−5
−3
Value
−48
4
10
37
Time post HSCT, months

Command line usage

```
$VDJTOOLS Correct \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-d`	`--depth`	1+	Maximum number of mismatches allowed between clonotypes being compared. Default is 2
`-r`	`--ratio`	[0, 1)	Child-to-parent clonotype size ratio threshold under which child clonotype is considered erroneous. Default is 0.05
`-a`	`--match-segment`		Check for erroneous clonotypes only among those that have identical V and J assignments
`-c`	`--compress`		Compress output sample files
`-h`	`--help`		Display help message

Tabular output

Outputs corrected samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `corr:[-d option value]:[-r option value]:['vjmatch' or 'all' based on -a option]` to `..filter..` metadata column.

Graphical output

none

1.10.2 Decontaminate

Cross-sample contamination can occur at library prep stage, for example sample barcode swithing resulting from PCR chimeras. Those could lead to a high number of artificial shared clonotypes for samples sequenced in the same batch. If no sophisticated library prep method (e.g. paired-end barcoding) is applied, it is highly recommended to filter those before performing any kind of cross-sample analysis.

This routine filters out all clonotypes that have a matching clonotype in a different sample which is `-r` times more abundant. Clonotype fractions within samples are considered, which is good for dealing with FACS-related contaminations. In case of dealing with cross-sample contaminations in samples coming from the same sequencing lane use `--read-based` option that tells the routine to compare read counts.

Command line usage

```
$VDJTOOLS Decontaminate \
[options] [sample1.txt sample2.txt ... if -m is not specified] filter_sample output_
↪prefix
```

Parameters

Short-hand	Long name	Argu-ment	Description
`-S`	`--software`	string	Input format. See *Common parameters*
	`--read-basis`	string	If set will compare clonotype read counts. Clonotype fractions in corresponding samples are compared by default.
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-r`	`--ratio`	nu-meric	Parent-to-child clonotype frequency ratio for contamination filtering. Defaults to `20`
`-c`	`--compress`		Compress output sample files
`-h`	`--help`		Display help message

Tabular output

Outputs filtered samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `dec:[-r value]` to `..filter..` metadata column.

Graphical output

none

1.10.3 DownSample

Down-samples a list of clonotype abundance tables by randomly selecting a pre-defined number of reads or clonotypes. This routine could be useful for

- normalizing samples to remove certain biases for depth-dependent statistics
- speeding up computation / decreasing file size and memory footprint.

Command line usage

```
$VDJTOOLS DownSample \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-x`	`--size`	integer	Number of reads/clonotypes to take. **Required**
`-u`	`--unweighted`		Will not weight clonotypes by frequency
`-c`	`--compress`		Compress output sample files
`-h`	`--help`		Display help message

Tabular output

Outputs sub-samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `ds:[-x value]` to `..filter..` metadata column.

Graphical output

none

1.10.4 FilterNonFunctional

Filters non-functional (non-coding) clonotypes, i.e. the ones that contain a stop codon or frameshift in their receptor sequence. Those clonotypes do not have any functional role, but they are useful for dissecting and studying the V-(D)-J recombination machinery as they do not pass thymic selection.

Command line usage

```
$VDJTOOLS FilterNonFunctional \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-e`	`--negative`		Negative filtering, i.e. only non-functional clonotypes are retained
`-e`	`--negative`		Negative filtering, i.e. only non-functional clonotypes are retained
	`--save-freqs`		Don't re-calculate clonotype frequencies and use those from original sample (no re-normalization)
`-h`	`--help`		Display help message

Tabular output

Outputs filtered samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `ncfilter:[retain or remove based on -e option]` to `..filter..` metadata column.

Creates a filter summary file with a `ncfilter.summary.txt` suffix containing info on the number of unique clonotypes that passed the filtering process, their total frequency and count.

Graphical output

none

1.10.5 SelectTop

Selects top N clonotypes from the sample. Useful for studying exapanded clonotypes and clonotypes with strong convergent recombination bias, as well as robust computing of unweighted statistics.

Command line usage

```
$VDJTOOLS SelectTop \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-x`	`--top`	integer	Number of top clonotypes to take. **Required**
	`--save-freqs`		Don't re-calculate clonotype frequencies and use those from original sample (no re-normalization)
`-c`	`--compress`		Compress output sample files
`-h`	`--help`		Display help message

Tabular output

Outputs sub-samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `top:[-x value]` to `..filter..` metadata column.

Graphical output

none

1.10.6 FilterByFrequency

Selects clonotypes that either have a frequency above the specified threshold and/or constitute more than a specified percent of reads (e.g. quantile threshold of 0.25 will top N clonotypes that in total contain 25% of reads in the sample). Those two filters can be used together or separately by setting either frequency threshold to 0 or quantile threshold to 1.

Command line usage

```
$VDJTOOLS FilterByFrequency \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-f`	`--freq-threshold`	0.0-1.0	Clonotype frequency threshold. Default is `0.01`
`-q`	`--quantile-threshold`	0.0-1.0	Quantile threshold. Will retain a set of top N clonotypes so that their total frequency is equal or less to the specified threshold. Default is `0.25`
	`--save-freqs`		Don't re-calculate clonotype frequencies and use those from original sample (no re-normalization)
`-c`	`--compress`		Compress output sample files
`-h`	`--help`		Display help message

Tabular output

Outputs filtered samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `freqfilter:[-f value]:[-q value]` to `..filter..` metadata column.

Graphical output

none

1.10.7 ApplySampleAsFilter

Retains/filters out all clonotypes found in a given sample **S** from other samples. Useful when **S** contains some specific cells of interest e.g. tumor-infiltrating T-cells or sorted tetramer+ T-cells.

Command line usage

```
$VDJTOOLS ApplySampleAsFilter \
[options] [sample1.txt sample2.txt ... if -m is not specified] filter_sample output_
↪prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-i`	`--intersect-type`	string	Sample intersection rule. Defaults to `strict`. See *Common parameters*
`-e`	`--negative`		Negative filtering, i.e. only clonotypes absent in sample *S* are retained
	`--save-freqs`		Don't re-calculate clonotype frequencies and use those from original sample (no re-normalization)
`-c`	`--compress`		Compress output sample files
`-h`	`--help`		Display help message

Tabular output

Outputs filtered samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `asaf:[- if -e, + otherwise]:[-i value]` to `..filter..` metadata column.

Graphical output

none

1.10.8 FilterBySegment

Filters clonotypes that have V/D/J segments that match a specified segment set.

Command line usage

```
$VDJTOOLS FilterBySegment \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-n`	`--negative`		Retain only clonotypes that lack specified V/D/J segments.
`-v`	`--v-segments`	v1,v2,...	A comma-separated list of Variable segment names. Non-matching incomplete names will be partially matched.
`-d`	`--d-segments`	d1,d2,...	A comma-separated list of Diversity segment names. Non-matching incomplete names will be partially matched.
`-j`	`--j-segments`	j1,j2,...	A comma-separated list of Joining segment names. Non-matching incomplete names will be partially matched.
	`--save-freqs`		Don't re-calculate clonotype frequencies and use those from original sample (no re-normalization)
`-c`	`--compress`		Compress output sample files
`-h`	`--help`		Display help message

Tabular output

Outputs filtered samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `segfilter:[retain or remove based on -e option]:[-v value]:[-d value]:[-j value]` to `..filter..` metadata column.

Creates a filter summary file with a `segfilter.summary.txt` suffix containing info on the number of unique clonotypes that passed the filtering process, their total frequency and count.

Graphical output

none

1.11 Operate on clonotype tables

1.11.1 JoinSamples

Joins several clonotype tables together to form a joint clonotype abundance table. Joint clonotype holds information on all clonotypes that match under a certain comparison criteria (e.g. identical CDR3nt and V segment), their samples of origin and corresponding abundances. At least two samples should be specified for this routine. For two sample case also consider using *OverlapPair* routine.

Attention: This is the most memory-demanding routine, especially for a large number of samples.

Command line usage

```
$VDJTOOLS JoinSamples \
[options] [sample1.txt sample2.txt sample3.txt ... if -m is not specified] output_
↪prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See See *Common parameters*
`-i`	`--intersect-type`	string	Sample intersection rule. Defaults to `aa`. See *Common parameters*
`-x`	`--times-detected`	integer	Minimal number of samples in which a clonotype should be detected to get to the final output. Default = 2
`-p`	`--plot`		Turns on plotting. See *Common parameters*
`-c`	`--compress`		Compressed output for clonotype table. See *Common parameters*
`-h`	`--help`		Display help message

Tabular output

Summary table suffixed `join.[value of -i argument].summary.txt` is created with the following columns.

Column	Description
<first sample id >	Indicator for the first sample, either `0` or `1`
<second sample id >	Indicator for the second sample
...	
clonotypes	Number of clonotypes detected in all samples that have `1` indicator in a given row.

Joint clonotype abundance table file having `join.[value of -i argument].table.txt` suffix that contains joint clonotypes detected in at least `-x` samples. Table structure is described in the section below.

Joint clonotype abundance table structure

First columns have the same meaning as in *VDJtools format* clonotype abundance table, they are computed as follows:

- Normalized frequency is computed as geometric mean of clonotype frequencies that comprise a given joint clonotype in intersected samples. If clonotype is missing, its frequency is set to `1e-9`.

Note: Joint clonotype is formed as a union of all clonotype variants in all samples that match under the specified `-i` rule.

- Normalized count is calculated by scaling normalized frequencies so that the joint clonotypes with smallest frequency has a count of `1`.
- Clonotype signature (CDR3nt, CDR3aa, V, D and J) is taken from a representative clonotype.

Note: When several clonotype variants are present in samples that correspond to the same clonotype under `-i` rule (e.g. several Variable segment variants when `-i nt` is set), only the most abundant form is selected as a **representative** clonotype to final output.

Column	Description
count	Normalized clonotype count
freq	Normalized clonotype frequency
cdr3nt	Representative CDR3 nucleotide sequence
cdr3aa	Representative CDR3 amino acid sequence
v	Representative Variable segment
d	Representative Diversity segment
j	Representative Joining segment
peak	Index of a time point at which given clonotype reaches its maximum frequency
occurrences	Number of samples the joint clonotype was detected in
<*sample name*>	Frequency of a joint clonotype at corresponding sample
...	

Graphical output

A Venn diagram can be found in a file having `join.[value of -i argument].venn.pdf` suffix. Note that if there are more than 5 samples, it will be constructed for the first 5 samples. Plotting is performed using VennDiagram R package.

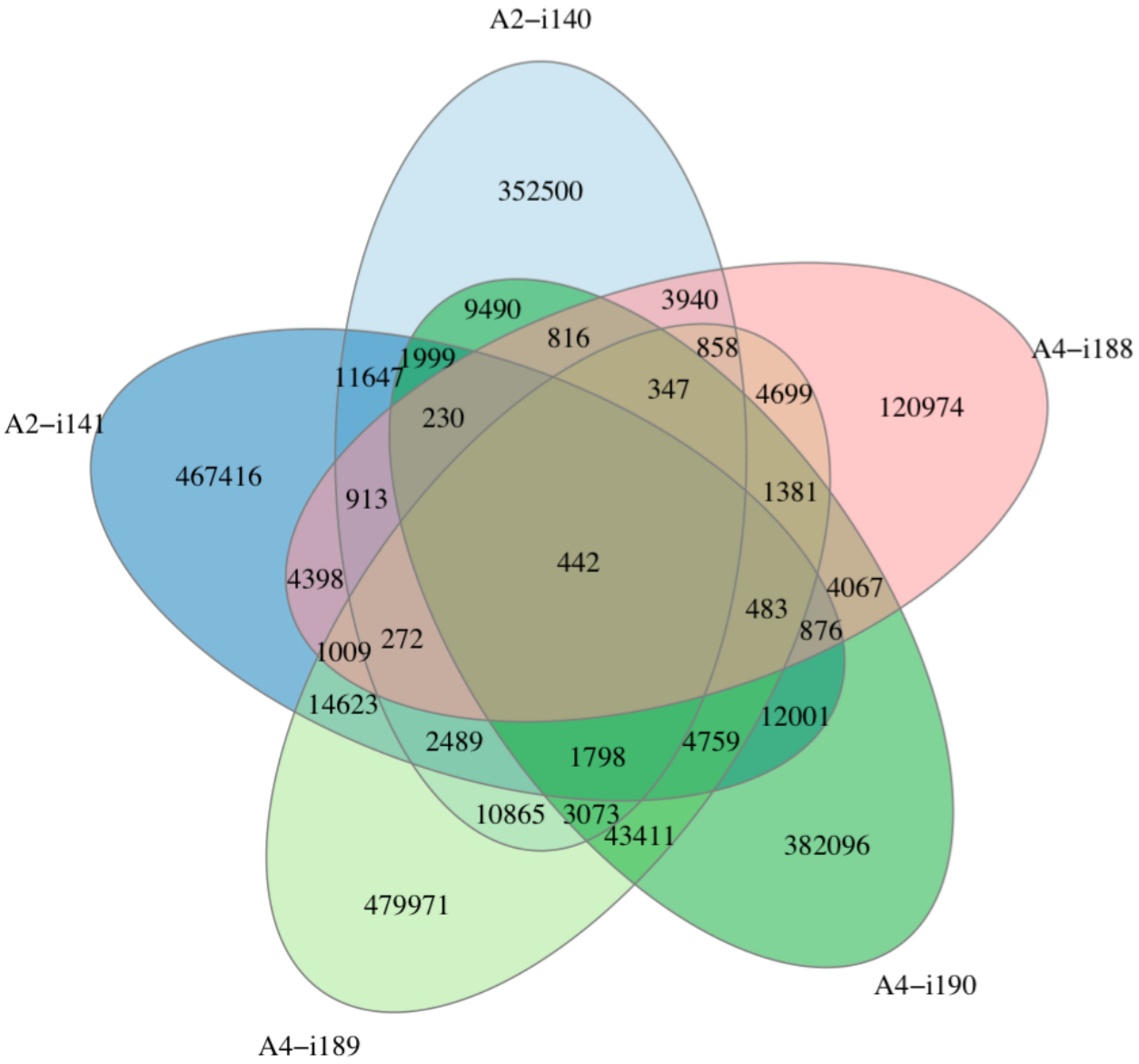
Overlap of clonotype sets. See Venn diagram wiki article for the description.

1.11.2 PoolSamples

Pools clonotypes from several samples together and merges clonotypes that that match under a certain comparison criteria (e.g. identical CDR3nt and V segment). Note that this routine can be used with a single sample to aggregate the sameple, e.g. by CDR3 amino acid sequence, in this case CDR3 nucleotide sequence, V and J segments will be taken from a representative clonotype variant with the highest frequency.

Command line usage

```
$VDJTOOLS PoolSamples \
[options] [sample1.txt sample2.txt sample3.txt ... if -m is not specified] output_
↪prefix
```


A2−i140
352500
9490
3940
816
858
1999
11647
347
4699
A4−i188
120974
A2−i141
230
467416
913
1381
442
4398
4067
483
876
272
1009
14623
12001
2489
4759
1798
10865
3073
43411
382096
479971
A4−i190
A4−i189

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-i`	`--intersect-type`	string	Sample intersection rule. Defaults to `strict`. See *Common parameters*
`-p`	`--plot`		Turns on plotting. See *Common parameters*
`-c`	`--compress`		Compressed output for clonotype table. See *Common parameters*
`-h`	`--help`		Display help message

Tabular output

Summary table suffixed `pool.[value of -i argument].summary.txt` is created with the following columns.

Column	Description
incidence.count	Indicator for the first sample, either `0` or `1`
read.count	Total number of reads associated with a given pooled clonotype
convergence	Total number of clonotype variants that match the pooled clonotype under `-i` rule.

Pooled clonotype abundance table file having `pool.[value of -i argument].summary.txt`. Table structure is described in the section below.

Pooled clonotype abundance table structure

First columns have the same meaning as in *VDJtools format* clonotype abundance table, they are computed as follows:

- Pooled count is computed as the total number of reads associated with clonotype variants that match under the specified `-i` rule.
- Frequency is computed as pooled count divided by total number of reads in all samples.
- Clonotype signature (CDR3nt, CDR3aa, V, D and J) is taken from a representative clonotype in the same way as described for *Joint clonotype abundance table structure*.

Column	Description
count	Pooled clonotype count
freq	Pooled clonotype frequency
cdr3nt	Representative CDR3 nucleotide sequence
cdr3aa	Representative CDR3 amino acid sequence
v	Representative Variable segment
d	Representative Diversity segment
j	Representative Joining segment
incidence	Number of samples containing clonotype variants that comprise a given pooled clonotype
convergence	Total number of clonotype variants that match the pooled clonotype under `-i` rule

Graphical output

planned

1.12 Annotation

1.12.1 CalcDegreeStats

Performs a TCR neighborhood enrichment test (TCRNET), testing each sample for clonotypes that have more neighbours (higher **degree** in a graph), i.e. clonotypes with similar CDR3 amino acid sequences, than would be expected by chance according to some control dataset. User can specify the actual **search scope** (i.e. number of allowed CDR3 mismatches), whether to only compare clonotypes with same V/J, and the control sample. If control sample is not provided, a pooling (see *PoolSamples*) of all provided samples is used. Note that this test, if supplied with real samples and a control pooled using `-i strict` option will account for the number of neighbours with the same CDR3 amino acid sequence, but distinct nucleotide sequences. If this is not desired, all input samples and control should be pre-pooled with `-i aa` or `-i aaVJ` to collapse variants coding for the amino acid CDR3 sequence.

Note: Running this routine will not return the actual clonotype graph for you, just annotate input samples. To build the graph, one should refer to VDJmatch software and its `Cluster` routine. Make sure the search scope option is the same as `-o` used for `CalcDegreeStats` and that all scoring/filtering is turned off. Next, one should retain only the edges that connect pairs of enriched clonotypes and enriched clonotypes with their neighbours.

Command line usage

```
$VDJTOOLS CalcDegreeStats \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-b`	`--background`	path	Path to the background (control) sample, used to compute expected statistics/P-values. If not provided, will pool input samples and uses them as control.
`-o`	`--search-scope`	s,id,t	Search scope: number of substitutions (s), indels (id) and total number of mismatches (t) allowed. Default is `1,0,1`
`-g`	`--grouping`	string	Primary grouping type, limits set of clonotype comparisons: 'dummy' (no grouping, default), 'vj' (same V and J) or 'vjl' (same V, J and CDR3 length).
`-g2`	`--grouping2`	string	Secondary grouping, used for computing statistics, accepts same values as `-g`. By default will select 'vjl' if no indels allowed and 'vj' otherwise.
`-h`	`--help`		Display help message

Note: There are two possible schemes for running the algorithm. Firstly, one can select, say a search scope of `1,0,1` allowing no indels, and `-g vjl` to only allow comparisons between clonotypes that match in V, J and CDR3 length. Then, one should only consider `p.value.g` in the output and disregard all columns with `g2/group2`. On the other hand, if one wants to allow comparison of clonotypes with different V/J, and/or comparisons with indels, the option `-g dummy` should be used. If one thinks there might be certain biases in V/J frequencies between control/background sample and input samples, and one wants to control for them, he should select `-g2 vj`, then observed degree values will be provided as is (i.e. not limiting clonotype comparisons to a fixed V/J), but the expected degree will be corrected to account for V/J usage difference between input sample and control. One should only consider `p.value.g2` in this case. See below for more explaination on output columns.

Tabular output

Processed samples will have additional annotation columns appended to VDJtools clonotype table columns. These columns are the following:

Column	Description
degree.s	Degree (number of neighbours) of a given clonotype in sample. The degree is the number of unique clonotypes (incl. nucleotide variants) that match a given clonotype under specified search scope.
group.count.s	Number of unique clonotypes that match the group, defined by primary grouping (`-g`), of a given clonotype in sample, say have the same V and J.
group2.count.s	Same as above, but the group is defined by secondary grouping `-g2`.
degree.c	Degree (number of neighbours) of a given clonotype in the control sample.
group.count.c	Number of unique clonotypes in the control sample that match the group of given clonotype as defined by primary grouping (`-g`).
group2.count.c	Same as above, but the group is defined by secondary grouping `-g2`.
p.value.g	P-value for the neighbour (degree) enrichment of a given clonotype according to primary grouping. The P-value is computed as `Pbinom(n=degree.s|p=degree.c/group.count.c, N=group.count.s)`.
p.value.g2	P-value for the neighbour (degree) enrichment of a given clonotype according to secondary grouping. The P-value is computed as `Ppoisson(n=degree.s|lambda=group.count.s*degree.c/group.count.c)`.

A metadata file will be created for resulting samples with `degstat` appended to the `..filter..` metadata column.

Graphical output

none

1.12.2 CalcCdrAAProfile

Generates amino acid physical properties profile of CDR3. Amino acids are first grouped to corresponding CDR3 sub-regions and then binned by position within the sub-region. Amino acids in a given bin is scored according to its physical properties, sums of those scores and total number of amino acids is reported for each sample/sub-region/bin/property combination.

For example under the **polarity** property amino acids are marked as polar (`1`) and non-polar (`0`) and the sum of these values is returned. When divided by the total number of amino acids one will get the fraction of polar amino acids in a given sample/sub-region. For **volume** the same operation will return the average volume of amino acids.

Command line usage

```
$VDJTOOLS CalcCdrAAProfile \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Shorthand	Long name	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-w`	`--weighted`		If set, will weight amino acid property values by clonotype frequency.
`-n`	`--normalize`		If set, will normalize amino acid property values by dividing them by corresponding CDR3 sub-region size.
`-r`	`--region-list`	region1,...	List of CDR3 sub-regions to count statistics for, default is `"CDR3-full,VJ-junc,V-germ,J-germ`
`-o`	`--property-list`	property1,...	List of amino acid physicochemical properties to use, see below for allowed value. Uses all amino acid properties from list below by default.
`-h`	`--help`		Display help message

Supported CDR3 sub-regions:

Name	Description
`CDR3-full`	Complete CDR3 region
`CDR3-center`	Central 5 amino acids of CDR3
`V-germ`	Germline part of CDR3 region corresponding to Variable segment
`D-germ`	Germline part of CDR3 region corresponding to Diversity segment
`J-germ`	Germline part of CDR3 region corresponding to Joining segment
`VD-junc`	Variable-Diversity segment junction, applicable when D segment is mapped
`DJ-junc`	Diversity-Joining segment junction, applicable when D segment is mapped
`VJ-junc`	Variable-Joining segment junction, including D segment if it is mapped

Supported amino acid physical properties (see full table for raw values):

Name	Description	Reference
`alpha`	Preference to appear in alpha helices	Stryer L et al. Biochemistry, 5th edition. ISBN 978-0716746843
`beta`	Preference to appear in beta sheets	Stryer L et al. Biochemistry, 5th edition. ISBN 978-0716746843
`turn`	Preference to appear in turns	Stryer L et al. Biochemistry, 5th edition. ISBN 978-0716746843
`surface`	Residues that have unchanged accessibility area when PPI partner is present	PMID:22559010
`rim`	Residues that have changed accessibility area, but no atoms with zero accessibility in PPI interfaces	PMID:22559010
`core`	Residues that have changed accessibility area and at least one atom with zero accessibility in PPI interfaces	PMID:22559010
`disorder`	Intrinsic structural disorder-promoting, order-promoting and neutral amino acids	PMID:11381529
`charge`	Charged/non-charged amino acids	Wikipedia
`pH`	Amino acid pH level	Wikipedia
`polarity`	Polar/non-polar amino acids	Wikipedia
`hydropathy`	Amino acid hydropathy	Wikipedia
`volume`	Amino acid volume	Wikipedia
`strength`	Strongly-interacting amino acids / amino acids depleted by purifying selection in thymus	PMID:18946038
`mjenergy`	Mean value of MJ statistical potential for each amino acid, used to derive 'strength'	PMID:8604144
`kf1`..`kf10`	Values of 10 Kidera factors summarizing physicochemical properties of amino acids	unpublished

Tabular output

A summary table with averaged amino acid property values is generated, suffixed `cdr3aa.profile.[wt or unwt based on -u].txt`. The table contains the following columns:

Column	Description
sample_id	Sample unique identifier
...	Sample metadata columns. See Metadata section
region	Current CDR3 sub-region, see above
property	Amino acid physical property name, see above
mean	Mean property value

Graphical output

none

1.12.3 Annotate

This routine will compute a set of properties for each clonotype's CDR3 sequence and append them to resulting clonotype table. For example, number of added N-nucleotides and the sum of polar amino acids in CDR3. The main difference from *CalcCdrAAProfile* is that the former computes sample-level average while this routine performs calculation on clonotype level.

Command line usage

```
$VDJTOOLS Annotate \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long-hand	Argument	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-b`	`--basic`	param1,param2,...	Comma-separated list of basic clonotype features to calculate and append to resulting clonotype tables. See below for allowed values. Default: `cdr3Length,ndnSize,insertSize`
`-a`	`--aap`	prop-erty1,...	Comma-separated list of amino acid properties. Amino acid property value sum will be calculated for CDR3 sequence (blank annotations will be generated for non-coding clonotypes). See below for allowed values. Default: `hydropathy,charge,polarity,strength,contact`
`-h`	`--help`		Display help message

List of basic annotation properties:

Name	Description
`cdr3Length`	Length of CDR3 region
`NDNSize`	Number of nucleotides between last base of V germline and first base of J germline parts of CDR3
`insertSize`	Number of added N-nucleotides
`VDIns`	Number of added N-nucleotides in V-D junction or `-1` if D segment is undefined
`DJIns`	Number of added N-nucleotides in D-J junction or `-1` if D segment is undefined

See *CalcCdrAAProfile* for the list of amino acid properties available for annotation. Sum of specified amino acid property values across all amino acids of CDR3 will be computed. It can be divided by `cdr3Length / 3` basic property value to get the average.

Tabular output

Processed samples will have additional annotation columns appended to VDJtools clonotype table columns. Those columns will be prefixed with `base.` for basic CDR3 properties and `aaprop.` for CDR3 amino acid composition properties.

A metadata file will be created for resulting samples with `annot:[-b value]:[-a value]` appended to the `..filter..` metadata column.

Graphical output

none

1.12.4 ScanDatabase (DEPRECATED since v1.0.5, use VDJmatch)

Annotates a set of samples using immune receptor database based on V-(D)-J junction matching. By default uses VDJdb, which contains CDR3 sequences, Variable and Joining segments of known specificity obtained using literature mining. This routine supports user-provided databases and allows flexible filtering of results based on database fields. The output of ScanDatabase includes both detailed (clonotype-wise) annotation of samples and summary statistics. Only amino-acid CDR3 sequences are used in database querying.

Command line usage

```
$VDJTOOLS ScanDatabase \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-D`	`--database`	path	Path to an external database file. Will use built-in VDJdb if not specified.
`-d`	`--details`		Will provide a detailed output for each sample with annotated clonotype matches
`-f`	`--fuzzy`		Will query database allowing at most 2 substitutions, 1 deletion and 1 insertion but no more than 2 mismatches simultaneously. If not set, only exact matches will be reported
	`--filter`	expression	Logical pre-filter on database columns. See below
	`--v-match`		V segment must to match
	`--j-match`		J segment must to match
`-h`	`--help`		Display help message

Note: Database filter is a logical expression that contains reference to input table columns. Database column name references should be surrounded with double underscores (__). Syntax supports Regex and standard Java/Groovy functions such as `.contains()`, `.startsWith()`, etc. Here are some examples:

```
__origin__=~/EBV/
!(__origin__=~/CMV/)
```

Note that the expression should be quoted: `--filter "__origin__=~/HSV/"`

Tabular output

A summary table suffixed `annot.[database name].summary.txt` is generated. First header line marked with `##FILTER` contains filtering expression that was used. The table contains the following columns:

Column	Description
sample_id	Sample unique identifier
...	Sample metadata columns. See Metadata section
diversity	Number of clonotypes in sample
match_size	Number of matches between sample and database. In case `--fuzzy` mode is on, all matches will be counted. E.g. if clonotype `a` in the sample matches clonotypes `A` and `B` in the database and clonotype `b` in the sample matches clonotype B the value in this column will be 3.
sample_diversity_in_matches	Number of unique clonotypes in the sample that matched clonotypes from the database
db_diversity_in_matches	Number of unique clonotypes in the database that matched clonotypes from the sample
sample_freq_in_matches	Overall frequency of unique clonotypes in the sample that matched clonotypes from the database
mean_matched_clone_size	Geometric mean of frequency of unique clonotypes in the sample that matched clonotypes from the database

Detailed database query results will be also reported for each sample if `-d` is specified. Those tables are suffixed `annot.[database name].[sample id].txt` and contain the following columns.

Column	Description
score	CDR3 sequence alignment score
query_cdr3aa	Query CDR3 amino acid sequence
query_v	Query Variable segment
query_j	Query Joining segment
subject_cdr3aa	Subject CDR3 amino acid sequence
subject_v	Subject Variable segment
subject_j	Subject Joining segment
v_match	`true` if Variable segments of query and subject clonotypes match
j_match	`true` if Joining segments of query and subject clonotypes match
mismatches	Comma-separated list of query->subject mismatches
...	Database fields corresponding to subject clonotype

Graphical output

none

1.13 Utilities

1.13.1 SplitMetadata

Splits metadata file into separate metadata files according to the set of values in specified column(s). Can be handly for implementing pipelines using VDJtools.

Command line usage

```
$VDJTOOLS SplitMetadata [options] metadata.txt output_dir
```

Parameters:

Short-hand	Long name	Argument	Description
`-c`	`--columns`	string1,string2,...	A comma separated list of column name(s) to split metadata by.

Tabular output

Output resulting metadata files to specified folder. Unique combinations of metadata entries in specified columns will be appended to names of corresponding metadata files, relative sample paths will be handled appropriately.

1.13.2 FilterMetadata

Filters metadata by evaluating expression over values in specified metadata columns, e.g.:

```
"__chain__=~/TR[AB]/"
"__chain__=='TRA'||__chain__=='TRB'"
"__chain__.contains('TRA')"
"!__condition__.startsWith('control')"
```

Both Java and Groovy syntax are supported, column names should be marked by double underscores before and after the name.

Command line usage

```
$VDJTOOLS FilterMetadata [options] metadata.txt output_dir output_suffix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-f`	`--filter`	expres-sion	Filter expression, should be surrounded with quotation marks, metadata column names should be marked with __.

Tabular output

Filtered metadata table with corresponding suffix will be created in the specified folder, relative sample paths will be handled appropriately.

1.13.3 Convert

Converts datasets from an arbitrary supported format to *VDJtools format*. You can also re-normalize your data - collapse clonotypes by V, D, J and CDR3 nucleotide sequence and re-compute clonotype frequencies - by using `-S VDJtoolsRenorm` option. This is useful if you want to groom manually converted data, or somewhy your clonotype frequencies do not sum to 1.

Command line usage

```
$VDJTOOLS Convert \
[options] [sample1.txt sample2.txt ... if -m is not specified] output_prefix
```

Parameters:

Short-hand	Long name	Argu-ment	Description
`-S`	`--software`	path	Format to convert from, see the *Formats supported for conversion* section
`-m`	`--metadata`	path	Path to metadata file. See *Common parameters*
`-c`	`--compress`		Compressed output for clonotype table. See *Common parameters*

Tabular output

Outputs converted samples to the path specified by output prefix and creates a corresponding metadata file. Will also append `conv:[-S value]` to `..filter..` metadata column.

1.13.4 RInstall

Prints the list of required R packages and installs dependencies into a local library (*RPackages* folder) which is placed in the parent folder of VDJtools jar. If this routine does not return with "PASSED" message, manual installation of packages that failed to deploy is required.

Command line usage

```
$VDJTOOLS RInstall
```

1.14 Clonotype browser

In order to demonstrate VDJtools API features, a lightweight immune repertoire browser **VDJviz** was implemented by @bvdmitri. VDJviz is a Play framework application that uses D3js for interactive visualization of VDJtools output. It allows visualizing and comparing various immune repertoire features such as spectratypes and rarefaction curves.

To try it out register at vdjviz.milaboratory.com and upload some RepSeq files in any supported format.

Important: Currently there is an upload limit of 25 files with at most 10,000 clonotypes, so the *DownSample* routine could come in handy

---

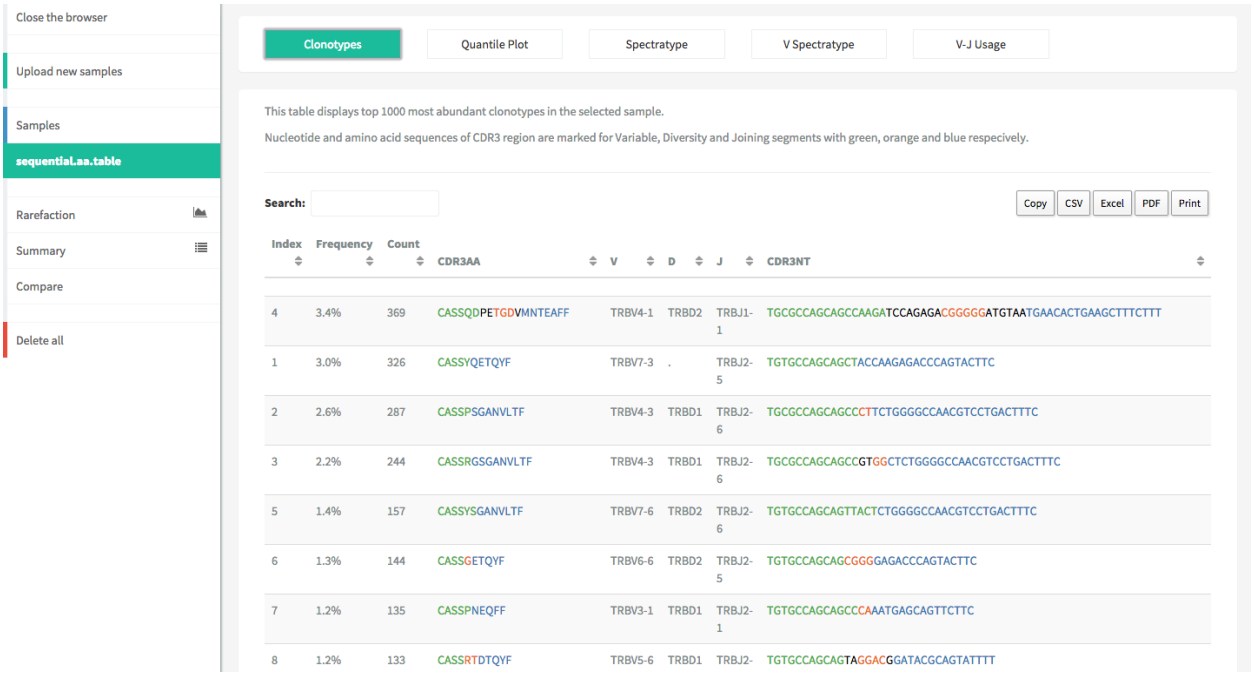


Fig. 2: Clonotype browser panel

Fig. 3: Interactive graphs