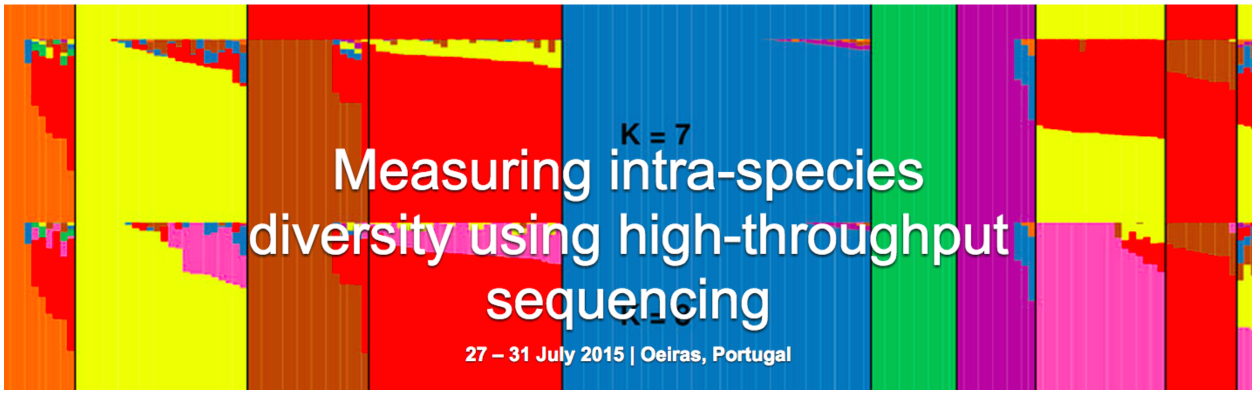
RepSeq tutorial @ EMBO practical course

Release 1.0.0

Mikhail Shugay

Dec 06, 2017

Contents



This tutorial covers processing and post-analysis of T-cell receptor (TCR) repertoire sequencing (RepSeq) data. General strategy for such data would be to de-multiplex samples and then map Variable (V), Diversity (D) and Joining (J) segments that are rearranged to form a mature TCR sequence. The complementarity determining region 3 (CDR3), a hypervariable TCR part that defines antigen specificity containing V-(D)-J junction, is then extracted. V, J and CDR3 regions are further assembled into clonotypes.

Datasets that are discussed here are amplicon libraries of TCR beta chain performed using 5'RACE that introduces unique molecular identifiers (UMI), short 12-bp tags of random nucleotides. Those are introduced at cDNA synthesis step and allow to trace reads back to their original cDNA molecules thus correcting for sequencing errors and allowing to count the number of starting molecules.

The following software tools are used for data processing and analysis:

- MIGEC for de-multiplexing data UMI-based error correction, mapping V(D)J segments
- MITCR for V(D)J mapping and frequency-based error correction
- VDJtools for computing diversity estimates and rarefaction analysis

Those tools are developed specifically for analysis of RepSeq, although some modules of MIGEC can be applied to broad range of UMI-tagged data.

The ‘repseq-tutorial <https://github.com/mikessh/repseq-tutorial>‘__repository contains software binaries, datasets and shell scripts listing all commands used in this tutorial. Some plotting and statistics is done in R.

Note: Some additional introduction slides can be accessed here.

CHAPTER 1

Table of Contents

1.1 Prerequisites

1.1.1 Datasets

The tutorial contains the following datasets:

- Raw sequencing reads (paired-end FASTQ) coming from two samples taken from the same individual, 1000 T-cells each. Together with them comes `barcodes.txt` storing information for de-multiplexing. To be used in Part 1.
- Diversity estimates generated by VDJtools for data from 39 donors of various age. To be used in Part 2 and 3

Datasets are stored in the tutorial repository, to download them run:

```
git clone https://github.com/mikessh/repseq-tutorial
```

they can be found in `part1/` and `part2/` folders respectively.

1.1.2 Software

All software tools necessary for this tutorial are stored in the `repseq-tutorial` repository as well.

Warning: Java v1.8+ is required to run these tools.

MIGEC

A pipeline for processing of RepSeq data, focused on UMI-tagged reads. There are 5 steps of MIGEC pipeline:

- **Checkout** De-multiplexing, UMI sequence extraction, read re-orientation, adapter trimming
- **Histogram** UMI coverage statistics

- **Assemble** Assemble consensus sequences from reads tagged with the same UMI. Filters out low-coverage and erroneous UMIs.
- **CdrBlast** V-(D)-J mapping and clonotype assembly
- **FilterCdrBlastResults** Additional correction of hot-spot errors.

Latest binaries: https://github.com/mikessh/migec/releases/latest

Docs: http://migec.readthedocs.org/en/latest/

MITCR

A software for fast and robust processing of TCR sequencing data. The software performs V-(D)-J mapping, clonotype assembly with frequency-based error correction and low-quality read rescue via re-mapping to assembled clonotypes.

Latest binaries: http://mitcr.milaboratory.com/downloads/

Docs: http://mitcr.milaboratory.com/documentation/

VDJtools

A framework for post-analysis of immune repertoire sequencing data. Includes 20+ analysis modules: from simple routines such as spectratyping to diversity estimation and repertoire clustering modules.

Latest binaries: https://github.com/mikessh/vdjtools/releases/latest

Docs: http://vdjtools-doc.readthedocs.org/en/latest/modules.html

Important: As VDJtools performs some plotting via R, so it can throw an error that a specific R package is lacking. Those packages should be installed with `install.packages` command in R shell.

Alternatively, one can run `java -jar vdjtools-1.0.2.jar RInstall` to allow VDJtools intall all R packages it uses (will take a substantial amount of time).

The list of required packages:

```
reshape, ggplot2, gridExtra, FField,
ape, reshape2, MASS, plotrix, RColorBrewer,
scales
```

1.2 Part I: Processing, error correction and diversity estimation

Clone the repository with datasets, MIGEC, MITCR and VDJtools executable JAR files and introduce aliases for running those JARs

```
export JAVA_OPTS="-Xmx8G" # set memory limit
MIGEC="java -jar ../migec-1.2.1b.jar"
MITCR="java -jar ../mitcr.jar"
VDJTOOLS="java -jar ../vdjtools-1.0.2.jar"
```

change dir to the folder with datasets

```
cd part1/
```

1.2.1 De-multiplex

The barcodes file is a tab-delimited file containing columns that correspond to sample ID, left and right barcode sequnece. Barcode sequence should contain a seed sub-sequence marked with upper-case bases, lower-case bases mark fuzzy match region, ambiguous nucleotides are allowed. Given multiplexed FASTQ files and file with sample barcodes one can run de-multiplexing as follows:

```
$MIGEC Checkout -ou barcodes.txt trb_R1.fastq.gz trb_R2.fastq.gz checkout/
```

Here `-u` option tells to extract UMI sequences marked with capital `N` in the barcodes file.

The `checkout/` directory will contain two pairs of FASTQ files for corresponding samples and a log file with info on de-multiplexing efficiency.

1.2.2 Check UMI statistics

Generate UMI coverage and error rate summary table

```
$MIGEC Histogram checkout/ histogram/
```

For plotting run the following auxiliary R script:

```
cd histogram/
wget https://raw.githubusercontent.com/mikessh/migec/master/util/histogram.R
Rscript histogram.R
cd ..
```

Note: Manual inspection reveals that there is good coverage peak separation (more than 5 reads per UMI) from low-coverage erroneous UMIs (peak at 1 read per UMI). Still there are some highly-amplified erroneous UMIs resulting from highly- covered true UMIs.

1.2.3 Assemble

Next, assemble all reads groups having the same UMI if the group count is 5+ reads `-m 5` and filter erroneous UMIs with `--filter-collisions`. The `--default-mask 0:1` option tells to only assemble second read which is the one containing CDR3 in current library prep setting (note that reads are oriented with Checkout routine).

```
$MIGEC AssembleBatch --force-overseq 5 --force-collision-filter --default-mask 0:1␣
↪checkout/ histogram/ assemble/
```

The `assemble/` folder now contains FASTQ files with assembled consensuses.

1.2.4 V-(D)-J mapping

The following code runs MIGEC/CdrBlast and MITCR software to map V, D and J segments, extract CDR3 sequences, assemble clonotypes and correct erroneous ones using various techniques.

```
# different quality thresholds
for q in 20 25 30 35; do
   $MIGEC CdrBlast -R TRB -q $q checkout/S2-1-beta_R2.fastq cdrblast/S2-1-beta.raw$q.
↪txt
done
```

```
# second sample, Q35, for replica-based filtering
$MIGEC CdrBlast -R TRB -q 35 checkout/S2-2-beta_R2.fastq cdrblast/S2-2-beta.raw35.txt
# frequency-based error correction (mitcr)
$MITCR -pset flex checkout/S2-1-beta_R2.fastq cdrblast/S2-1-beta.mitcr.txt
# assembled data
$MIGEC CdrBlast -a -R TRB assemble/S2-1-beta_R2.t5.cf.fastq cdrblast/S2-1-beta.asm.txt
$MIGEC CdrBlast -a -R TRB assemble/S2-2-beta_R2.t5.cf.fastq cdrblast/S2-2-beta.asm.txt
```

The results are provided as tab-delimited clonotype abundance tables.

1.2.5 Take a glance at the data

Upload clonotype table(s) from `cdrblast/` folder to vdjviz.milaboratory.com, browse the clonotype tables, check for erroneous clonotypes by performing a search for CDR3 amino acid sequence matching one of the top clonotypes.

1.2.6 Final steps and report generation

MIGEC pipeline aims at running all steps from de-multiplexing to clonotype assembly and error correction for a batch of samples.

To continue with *batch* MIGEC analysis, run

```
# Process both raw and assembled data
$MIGEC CdrBlastBatch -R TRB checkout/ assemble/ cdrblast/
# Filter results from hot-spot PCR errors
$MIGEC FilterCdrBlastResultsBatch cdrblast/ cdrfinal/
```

The `cdrfinal/` folder will contain filtered clonotype abundance tables. Once all stages of *batch* MIGEC are complete, one can generate analysis report with the following command:

```
$MIGEC Report .
```

Generated report will contain comprehensive statistics for all five stages of MIGEC analysis.

Note: Report generation uses R markdown and parsing it to HTML requires installation of additional libraries. One can either follow instructions on R markdown web page or install RStudio that will in turn install all necessary packages. If the report generation is unsuccessful, one can still use RStudio to compile the report template (`*.Rmd` file that will be generated anyway) by opening it and clicking “knit html” button.

1.2.7 Repertoire diversity analysis

First, convert samples into VDJtools input format

```
$VDJTOOLS Convert -S migec `ls cdrblast/S2-*-beta.raw*.txt` `ls cdrblast/S2-*-beta.
↪asm.txt` convert/
$VDJTOOLS Convert -S mitcr cdrblast/S2-1-beta.mitcr.txt convert/
```

Next, compare rarefaction curves for quality-based filtering, frequency-based filtering and UMI-based assembly

```
$VDJTOOLS RarefactionPlot -f sample_id `ls convert/S2-1-beta.raw*.txt` convert/S2-1-
↪beta.mitcr.txt rarefaction/qual-and-freq
# plot curve for assembled data separately, as it uses #UMIs as count, not reads
$VDJTOOLS RarefactionPlot -f sample_id convert/S2-1-beta.asm.txt convert/S2-2-beta.
↪asm.txt rarefaction/umi
```

Inspect pdf files in `rarefaction/` folder.

Note: Note that as only 1000 cells were sequenced, and the protocol efficiency is ~0.5 reactions per cell, therefore at most 500 clonotypes can be expected. The number of observed clonotypes in raw data is substantially higer,UMI-assembled data, on the other hand is in good agreement with our expectations.

1.2.8 Using replicates

Overlap clonotype tables for two Q35-filtered raw datasets coming from different samples and compare rarefaction curves

```
$VDJTOOLS OverlapPair convert/S2-1-beta.raw35.txt convert/S2-2-beta.raw35.txt convert/
$VDJTOOLS RarefactionPlot -f sample_id convert/S2-1-beta.raw35.txt convert/S2-2-beta.
↪raw35.txt convert/paired.strict.table.txt rarefaction/overlap
```

Note: There is still a substantial level of artificial diversity, suggesting that the errors in CDR3 sequence are recurrent. So replicates are not very useful to correct diversity estimates.

1.2.9 Expected results

1.3 Part II: Comparing diversity estimation methods

Datasets to be analyzed in this section can be found in `task2/` folder. There are 39 preprocessed clonotype tables for TRB repertoires of healthy donors of various ages. This folder also contains `metadata.txt` file with the age and percent of naive T-cells for each donor.

1.3.1 Estimating diversity

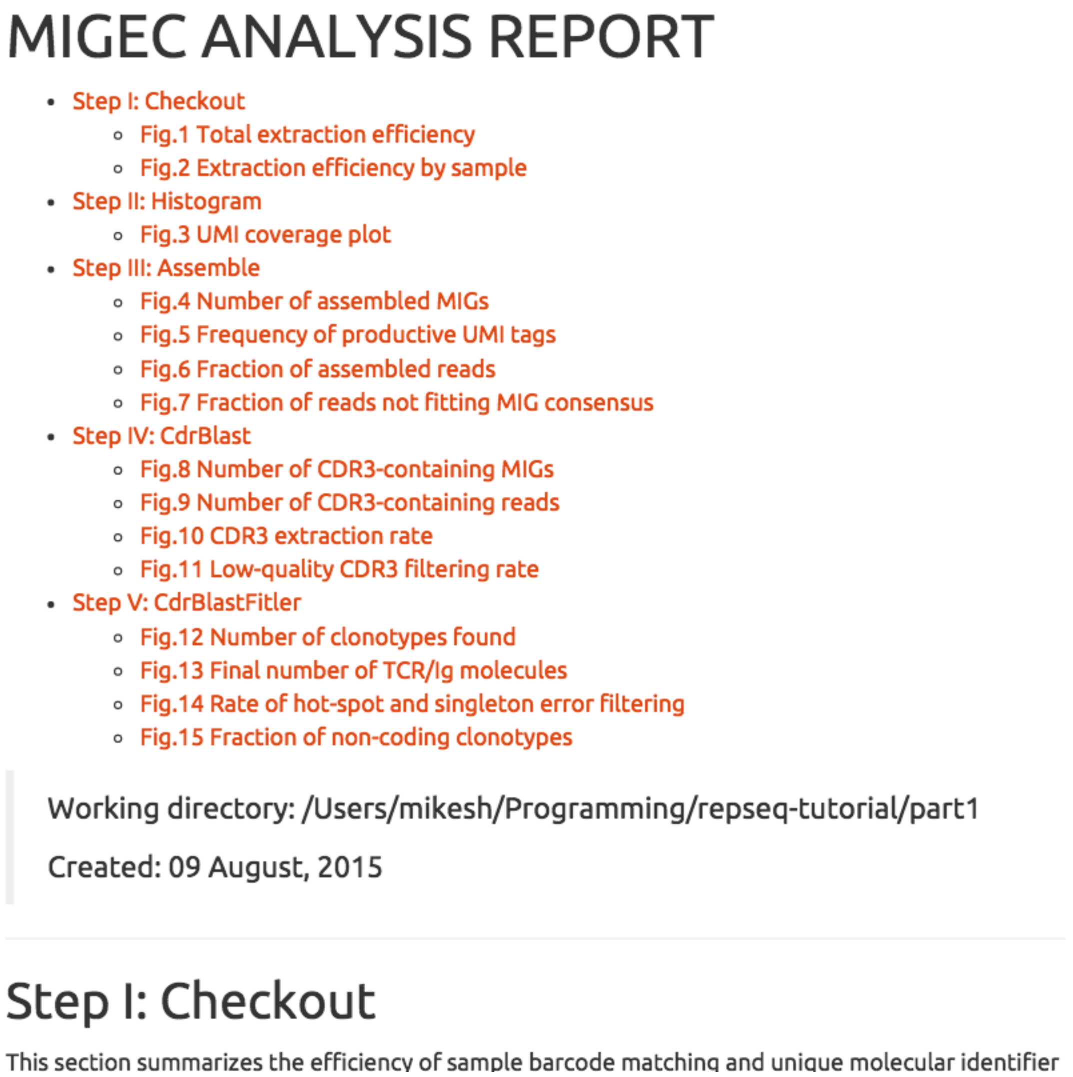
Corresponding routine of VDJtools framework is called CalcDiversityStats. The list of diversity estimates that can be computed using this routine is given here

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

This will generate two files with diversity estimates, suffixed as `.exact.txt` and `.resampled.txt`. The latter contains estimates computed on datasets down-sampled to the size of the smallest one.

1.3.2 Plotting and comparing

The following R script will generate two plots comparing the performance of various diversity estimates computed for original and resampled data. The comparison is performed based on Spearman correlation



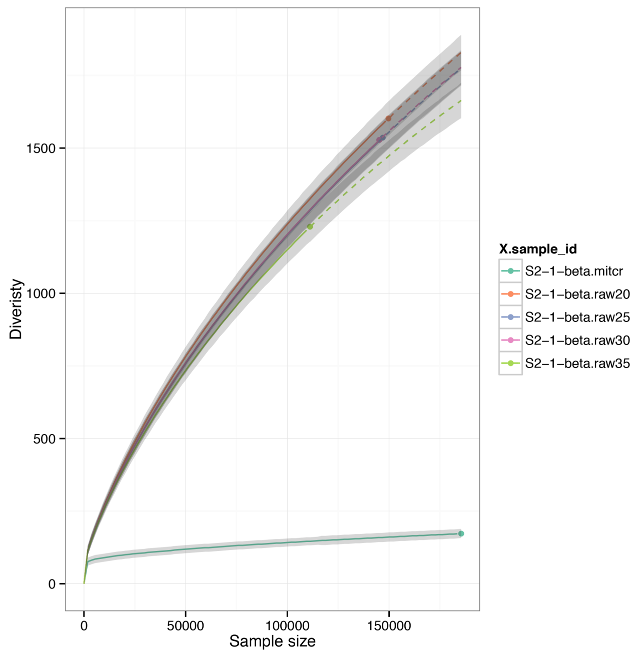
MIGEC ANALYSIS REPORT

Working directory: /Users/mikesh/Programming/repseq-tutorial/part1

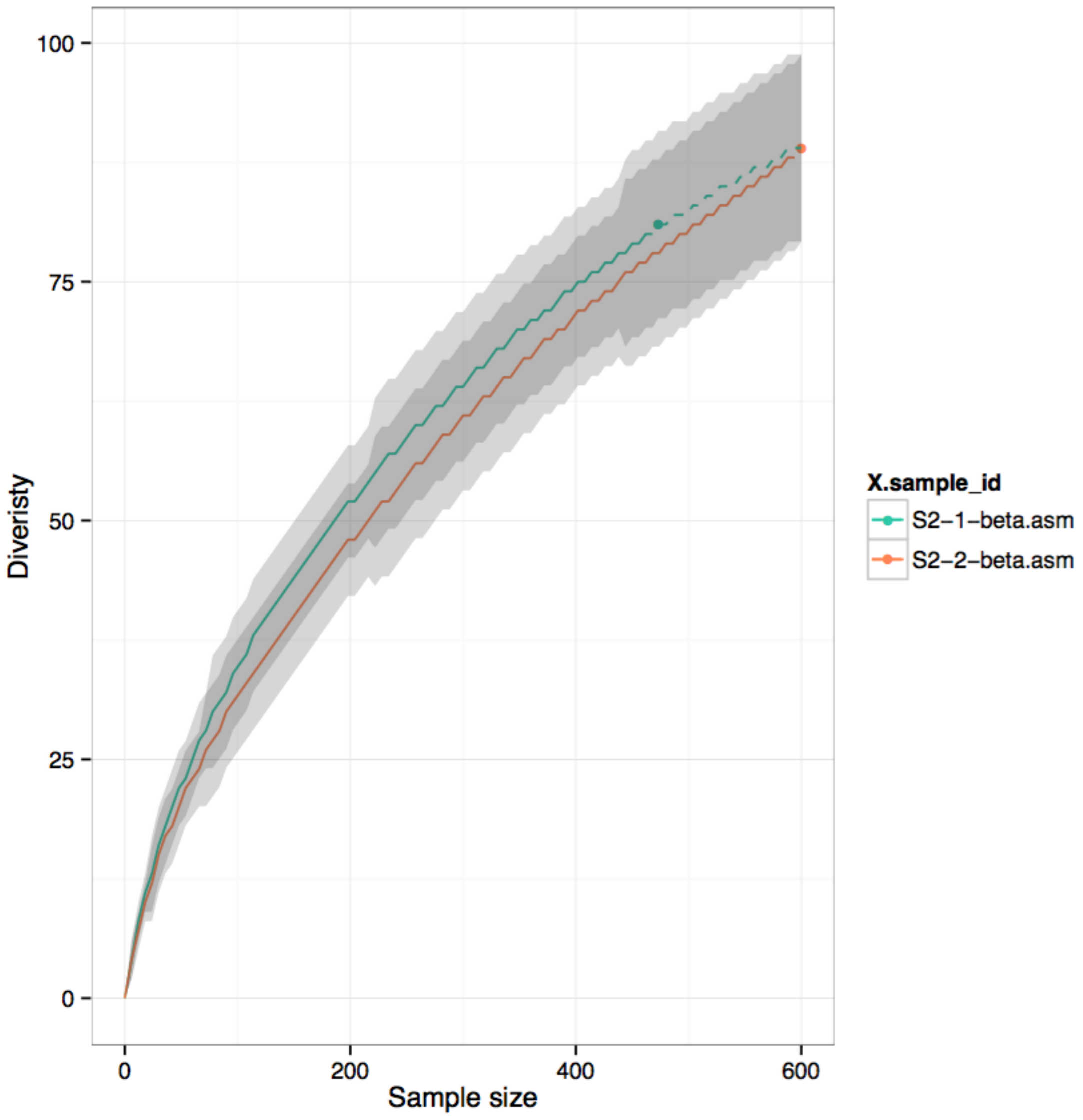
Created: 09 August, 2015

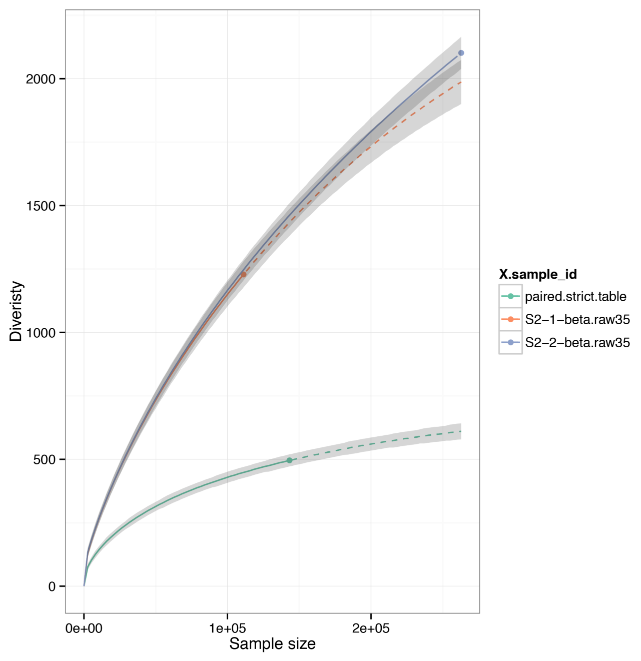
Step I: Checkout

This section summarizes the efficiency of sample barcode matching and unique molecular identifier



```
require(ggplot2)

# load data tables
df.e <- read.table("diversity.strict.exact.txt", header=T, comment="", sep="\t")
df.r <- read.table("diversity.strict.resampled.txt", header=T, comment="", sep="\t")

# prepare a table to store correlation coefficients
measures <- c("observedDiversity", "chaoE", "efronThisted", "chao1", "d50Index",
↪"shannonWienerIndex", "inverseSimpsonIndex")
methods <- c("exact", "resampled")

g <- as.data.frame(expand.grid(measures, methods))
colnames(g) <- c("measure", "method")
g[, "age"] <- numeric(nrow(g))
g[, "naive"] <- numeric(nrow(g))

# fill table
for (i in 1:nrow(g)) {
  measure <- as.character(g[i,1])
  method <- as.character(g[i,2])

  if (method == "exact") {
    df <- df.e
  } else {
    df <- df.r
  }

  y <- as.numeric(as.character(df[,paste(measure, "mean", sep="_")]))

  x <- as.numeric(as.character(df[,"age"]))
  g[i, 3] <- cor(x, y, method="spearman") ^ 2
  x <- as.numeric(as.character(df[,"naive"]))
  g[i, 4] <- cor(x, y, method="spearman") ^ 2
```

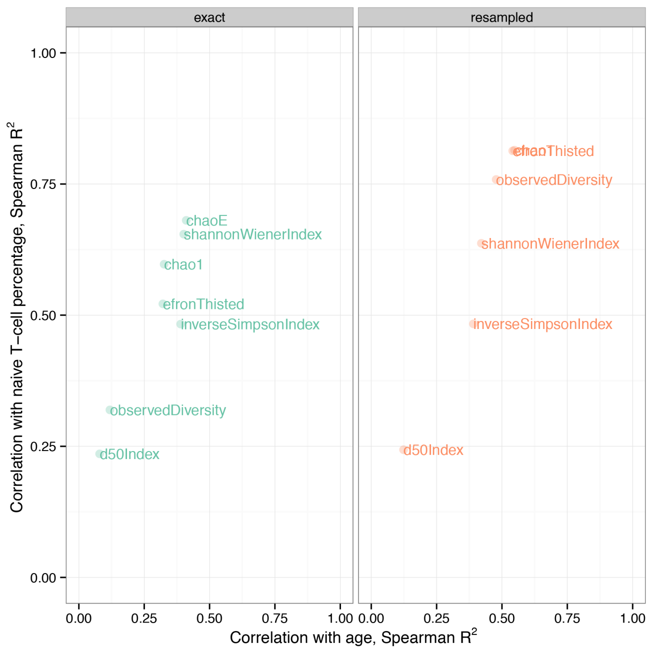

100
75
50
25
0
Diveristy
0
200
400
600
Sample size
X.sample_id
S2–1–beta.asm
S2–2–beta.asm



```
}

# plot results
pdf("measure_comparison.pdf")
ggplot(g, aes(x=age, y=naive, color=method, label=measure)) +
  geom_point(size=3, alpha=0.3) + geom_text(cex=4, hjust=0) +
  scale_x_continuous(name = bquote('Correlation with age, Spearman '*R^2*' '),␣
↪limits=c(0,1)) +
  scale_y_continuous(name = bquote('Correlation with naive T-cell percentage,␣
↪Spearman '*R^2*' '), limits=c(0,1)) +
  scale_color_brewer(palette="Set2") +
  facet_grid(~method) +
  theme_bw()+theme(legend.position="none")
dev.off()
```

Warning: Headers in VDJtools output are marked with #, so we need to specify `comment=""` when loading the data table in R.

Note: We use Spearman correlation coefficient, as the distribution for some measures like lower bound total diversity estimates is highly skewed.



1.3.3 Expected results

1.4 Part III: Diversity and similarity of repertoires

In this part the repertoire similarity in terms of the number of unique overlapping clonotypes and their frequency is calculated, termed overlap-D and overlap-F respectively. Datasets used in this part are the same as ones in Part II.

1.4.1 Computing pairwise similarities

As computing pairwise overlaps is either memory-demanding (in case we load all samples into memory) or slow (loading only one pair at a time), we will first down-sample datasets to 50,000 cells each.

```
$VDJTOOLS Downsample -x 50000 -m ../part2/metadata.txt .
```

Next we will compute pairwise similarities

```
$VDJTOOLS CalcPairwiseDistances -i aa\!nt -m metadata.txt .
```

Important: Here we apply the *aa!nt* clonotype matching rule, which means that clonotypes match when their CDR3 amino acid sequences match, but nucleotide sequences are different. This rule is effective agains cross-sample contamination, which is an issue in analyzed datasets as they were prepared in three separate batches (A2, A3 and A4)

1.4.2 Correlation between overlap and diversity

Run the following script to compute and plot the relation between sample diversity and its mean overlap with other samples

```
require(ggplot2); require(plyr); require(reshape)

# load table
df <- read.table("intersect.batch.aa!nt.txt", header=T, sep="\t", comment="")

df$F2 <- as.numeric(as.character(df$F2))

# need to do this, as only the upper triangular of intersection matrix is stored
df.1 <- rbind(data.frame(sample=df$X.1_sample_id, div=df$div1, overlapF=df$F2,␣
↪overlapD=df$div12),
              data.frame(sample=df$X2_sample_id, div=df$div2, overlapF=df$F2,␣
↪overlapD=df$div12))

# compute mean values for overlap-F and -D
df.2 <- ddply(df.1, .(sample), summarise,
              div=mean(div),
              overlapF=mean(overlapF),
              overlapD=mean(overlapD))

# plot
df.3 <- melt(df.2, id=c("sample","div"))

pdf("overlap.pdf")
ggplot(df.3, aes(x=div,y=value)) + geom_point() + geom_smooth(method=lm) +
  xlab("Sample diversity") + ylab("Metric value") +
  facet_grid(variable~.,scales="free_y") + theme_bw()
dev.off()
```

Note: More diverse samples are actually highly similar, both in terms of unique clonotypes in the overlap and total overlap frequency.

Next, lets cluster samples based on their similarity,

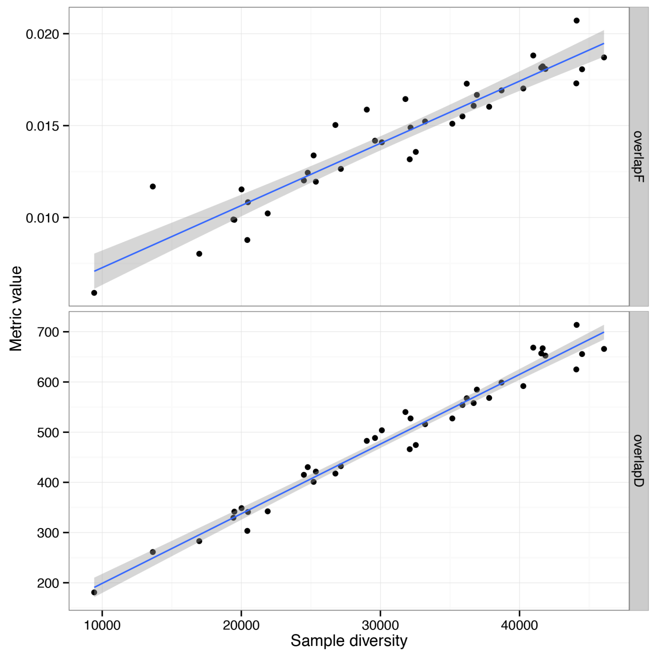
```
$VDJTOOLS ClusterSamples -e F2 -i aa\!nt -f age -n -p .
```

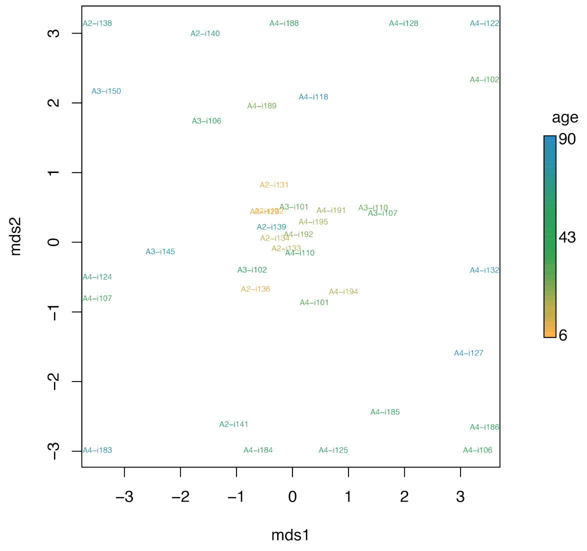
This will generate `hc*.pdf` with hierarchical clustering and `mds*.pdf` with multi-dimensional scaling plots.

Note: The MDS plot show that younger and more diverse repertoires are more similar than elder repertoires. From immunological point of view given additional data on umbilical cord blood repertoires and elder individuals (not shown here), this is interpreted in the following way:

We are born with highly similar, convergent and evolutionary optimized repertoires, but random clonal expansion (partially due to T-cell cross-reactivity) decrease the overlap between our repertoires.

1.4.3 Expected results


0.020
0.015
0.010
overlapF
700
600
500
400
300
200
overlapD
Metric value
10000
20000
30000
40000
Sample diversity


A2–i138
A2–i140
A4–i188
A4–i128
A4–i122
A4–i102
A3–i150
A4–i118
A4–i189
A3–i106
A2–i131
A2–i129
A2–i132
A3–i101
A4–i191
A3–i110
A3–i107
A4–i195
A2–i139
A4–i192
A2–i134
A2–i133
A3–i145
A4–i110
A3–i102
A4–i124
A4–i132
A2–i136
A4–i194
A4–i107
A4–i101
A4–i127
A4–i185
A2–i141
A4–i186
A4–i183
A4–i184
A4–i125
A4–i106
3
2
1
0
−1
−2
−3
mds2
−3
−2
−1
0
1
2
3
mds1
age
90
43
6