Bulk BCR/TCR Analysis Pipeline

USER GUIDE

Rachael Bashford-Rogers
WELLCOME TRUST CENTRE FOR HUMAN GENETICS | UNIVERSITY OF OXFORD

Program developed by Rachael Bashford-Rogers (2020)
At the Wellcome Centre for Human Genetics, University of Oxford
(rbr1@well.ox.ac.uk)

Table of Contents

Part 1

1. Introduction

This manual provides an outline and user guide for the B cell receptor (BCR) and T cell receptor (TCR) repertoire processing pipeline for bulk RNA sequencing data based on the IsoTyper and TCR protocols developed in the Bashford-Rogers Lab.
Download Code from: https://github.com/Bashford-Rogers-lab/Bulk_Repertoire_Analysis

Table 1: A summary of the pipeline is described below:

Stage 1:

1. QC sequences (Phred >30)

Stage 2:

1. Join forward and reverse reads (merging).
2. Split sequences according to internal PCR barcode.
3. Collapse/Error correct and generate consensus sequences using unique molecular identifiers (UMIs).
4. Assign isotype/constant region.
5. Filter for putative VDJ sequences.
6. Assign functionality and filter (Open reading frame). Note: Can turn filter off.

Stage 3:

1. Network generation: Here, each vertex represents a different sequence, and the number of identical BCR sequences defines the vertex size. Edges are created between vertices that differ by one nucleotide. Clusters are groups of interconnected vertices (*1, 2*). The program described here calculates edges between unique sequences and determines vertex sizes, creating output files in formats that can directly be used in network analysis programs such as networkx (python) or igraph (R or python).

Stage 4:

1. Visualise summary metrics and generate summary files from the results of Stages 1-3.
2. Calculate Jaccard indexes to check for contamination and library hopping and if more than one sample per patient can be used to check for sample mismatch.
    a. Basic mode (all sequences) and filtered mode (sequences occurring 2 or more times)
        i. No correction (Raw Jaccard Matrix).
        ii. Correct for index hopping
        iii. Correct for library contamination

Stage 5:

1. Sequence annotation using IMGT (gold standard).
2. Network Analysis.
3. Generation of broad repertoire statistics.

Stage 6:

1. Concatenate filtered fastq files into a smaller number of multi-individual large files.
2. Upload large files to **IMGT** for annotation.

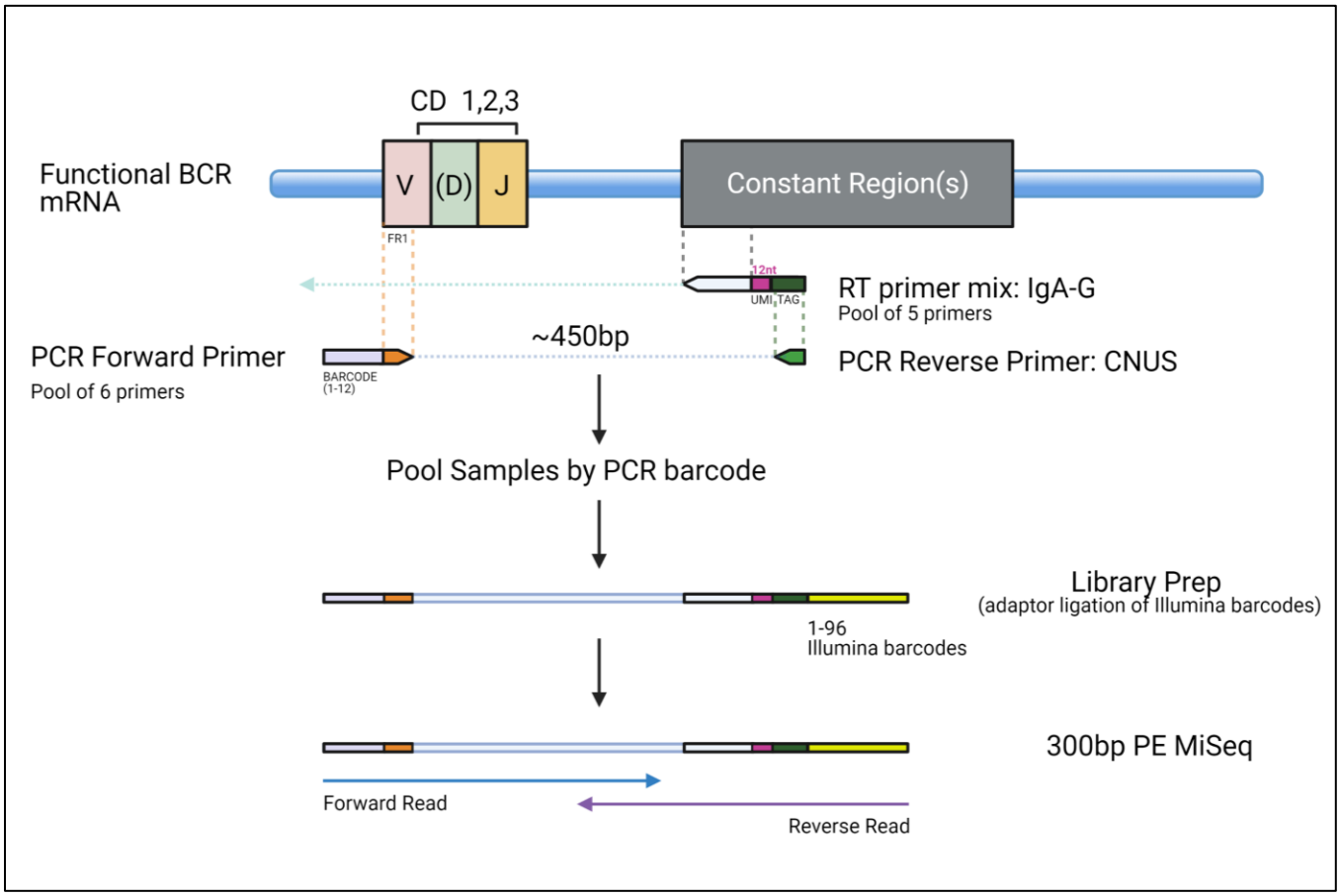
Stage 7:

1. Results of IMGT analysis are used in Isotyper specific Analysis.

Figure 1. Schematic of <u>BCR</u> repertoire amplification experimental protocol.

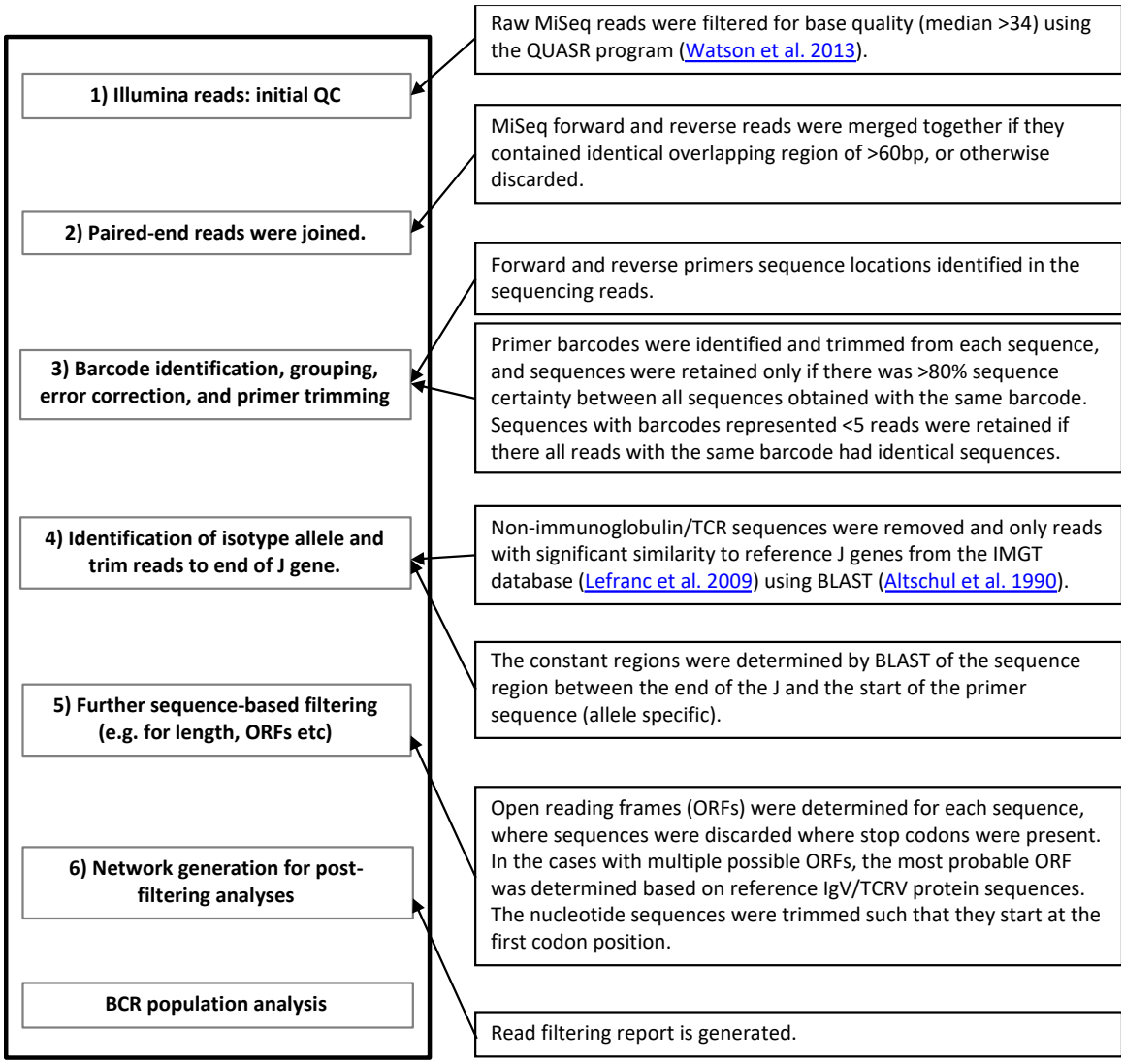
- A reverse transcription (RT) primer pool (5 primers binding to the constant region of each immunoglobulin, incorporating a unique molecular identifier (UMI)) is used to reverse transcribe RNA.
- cDNA is then amplified by multiplex-PCR using a reverse primer which binds to a tag on the RT primers and a pool of 6 barcoded forward primers which bind to the Framework Region 1 of all known BCR V genes. Forward primers are internally barcoded with 1 of 12 barcodes to enable sample multiplexing.
- Samples are then pooled according to PCR barcode e.g. 12 samples with barcodes 1-12 could be pooled together but not two samples both barcoded with barcode 12.
- Pooled samples are then library prepped using a unique Illumina barcode.
- Libraries with different Illumina barcodes are pooled 1:1 to a maximum of ~80 total samples per sequencing lane.
- Libraries are sequenced on a MiSeq using 300bp Paired End technology with 15% PhiX to improve phasing calculations.

The protocol for TCR repertoire amplification differs slightly in primer design but is conceptually equivalent to the BCR protocol.



Created with BioRender.com

Figure 2. Schematic of QC and filtering pipeline.



1. Register for IMGT

During the analysis pipeline filtered fastq files are annotated using IMGT/HighV-Quest – the gold standard for TCR and BCR repertoire analysis. To use this online tool you must register for a user account. All new users require approval prior to account activation by an administrator. Therefore we recommend registering as soon as possible to avoid potential hold-ups.

Register here (new user): http://www.imgt.org/HighV-QUEST/login.action

2. Installation

1. FLASH:

- Download current version from https://ccb.jhu.edu/software/FLASH/
- Unpack.

2. CD-HIT(*3*):

- Download current CD-HIT from: http://bioinformatics.org/cd-hit/
- Unpack the file with "tar xvf cd-hit-XXX.tar.gz --gunzip"
- Change dir by "cd cd-hit-2006"
- Compile the programs by "make"

3. Quasr(4):

- Download current version from: https://sourceforge.net/projects/quasr/

4. Blast:

- Download current version from: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download
- **Rescomp (BMRC, Oxford) users: Blast is already installed so you do not need to perform this stage.**

5. Python Modules:

Ensure that the following python modules are installed.

Rescomp (BMRC, Oxford) users*: these modules are already available using the Rescomp 'module load' system and need not be installed locally. However to enable access to these modules (load) you* ***MUST run analysis using the BCR_TCR_Wrapper_Cluster.sh*** *job submission script****.*** *If you prefer to use the python wrapper please install the following modules.*

- Sys
- Collections
- Os
- Operator
- networkx

6. R Modules:

Ensure that the following R modules are installed.

Note that the R analysis is only available using the ***BCR_TCR_Wrapper_Cluster.sh*** *job submission script.* **Rescomp (BMRC, Oxford) users***: these modules are already available using the Rescomp 'module load' system and need not be installed locally.*

- Tidyverse
- Data.table
- ggplot2
- ggforce
- Gviz
- foreach
- doParallel
- gridExtra
- cowplot
- gtools

7. Locations of Dependencies:

- To ensure the pipeline can call the dependencies edit: "BCR_TCR_PROCESSING_PIPELINE/ Locations_of_called_programmes.txt" file, providing the full path to correct locations of your versions of:
    - Reference library of genes and primers (already compiled for you)
    - CD-HIT (from above)
    - FLASH (from above)
    - Quasr (*4*) (from above)
    - Blast (from above)

8. Create Log Files Directory

- When using the *BCR_TCR_Wrapper_Cluster.sh wrapper all log files will be output to a directory called COMMANDLOGS within the pipeline directory. However this **may need** to be created prior to running the job submission wrapper using the following bash script e.g.:*

```
cd path_to/BCR_TCR_PROCESSING_PIPELINE
mkdir COMMANDLOGS
```

9. Update Primers

- In the event that novel PCR/RT primers have been used you will need to update the relevant files within the provided LIBRARY with the new primer sequences.
- Files are tab separated. Column 1 = name, 2 = full sequence. For reverse primers this is followed 3 columns, 3= tag, 4=UMI, and 5=complementary region.
- For **reverse transcription** primers, please **prefix** the primer name with "REVERSE_".
    - **For BCR:**
        - Primers_HOMO_SAPIENS_IG_RBR_Constant_region_MPLX.txt
        - Barcode_groups_FR1_groups1.txt
    - **For TCR:**
        - Barcode_groups_TCR_groups1.txt
        - Primers_HOMO_SAPIENS_TCR_RBR_Constant_region_MPLX_table.txt

3. Run

This pipeline performs the steps described in the introduction split into several separate job submission stages. **Each stage is run independently, however they must be run sequentially upon completion of the previous job**.

To run the pipeline on multiple samples we provide two wrappers a) a python based wrapper for job submission using bsub (Processing_sequences_large_scale.py) and b) a standard job submission bash script using qsub (BCR_TCR_Wrapper_Cluster.sh). **The latter is preferable for running on the BMRC cluster (Recomp) as it utilises the module system and can be easily adapted for another cluster architecture.** Both wrappers require the same input files - details of which can be found below.

Pre-Submission:

1. Sample Spreadsheets:

We provide two example sample files (for BCR and TCR examples) within the pipeline directory as a guideline. As this pipeline allows for the potential of double barcoding (Illumina and sample barcodes) you must create two input spreadsheets using the below specifications. *Note: spreadsheet 1 and spreadsheet 2 will be the same if there is no internal barcoding. In most cases you will need to change the sample names, internal barcodes, libraries, location of fastq and location of output files along with the path to the provided primers.*

1.1 Spreadsheet A - 'name_pre.txt':

- A spreadsheet detailing the library pools (e.g. for each Illumina barcode) to be **used with stage 1.** There must be no header-line or empty lines at the end of the txt file. It is often helpful to create this file in an excel spreadsheet and then save in the relevant format (see below).
    - **If running the bash wrapper (.sh) the sample sheet must be tab separated.**
    - **If running the python wrapper (.py) the sample sheet must be space (" ") separated.**

1.2 Spreadsheet B - 'name_post.txt':

- A spreadsheet detailing the individual samples, linked to the library pools from which they originate. This **will be used from stage 2 onwards**. There must be no header-line or empty lines at the end of the txt file. It is often helpful to create this file in an excel spreadsheet and then save in the relevant format (see below).
    - **If running the bash wrapper (.sh) the sample sheet must be tab separated.**
    - **If running the python wrapper (.py) the sample sheet must be space (" ") separated.**

2. Layouts File.

In order to run the JACCARD index correction R analysis you must provide a tab separated layouts file equivalent to Batching_LayoutsBCR.txt. This details the experimental design including: SampleID and Barcode (if multiple samples from the same individual), sequencing

lane, plate (for example if PCR is run in batches), Library, Position on plate and the internal PCR barcode used. Jaccard indexes will then be calculated with and without correction for index hopping / library contamination taking into consideration whether samples originate from the same individual and experimental design e.g. there can be no index hopping across sequencing lanes.

3. Python Wrapper

We provide a python wrapper, Processing_sequences_large_scale.py, which will manage the job submission for multiple samples using **bsub**. *Note: not recommended for Rescomp users and* ***incompatible with the R analysis****.*

3.1 Command Line arguments

In order to run the wrapper we must provide the following command line arguments in the order shown:

- Argument 1: Sample input file: name_pre.txt or name_post.txt (**space separated format**)
- Argument 2: Stage of pipeline: 1/2/3/ etc.
- Argument 3: Run script in job format - Y/N
- Argument 4: Print commands to screen - Y/N
- Argument 5: Run pipeline – Y/N

3.2 Submission

The python wrapper can be run in the following way (from within the pipeline directory):

```
python Processing_sequences_large_scale.py <sample file list> <commands (comma separated list)> <run as a job: Y/N> <print commands: Y/N> <run commands: Y/N
```

Note: ***Unless argument 5=Y*** *no analysis will be performed.*

4. BASH Wrapper

We provide a standard job submission bash script (BCR_TCR_Wrapper_Cluster.sh) which manages sample submission using qsub for any stage of the pipeline. ***Note: recommended for Rescomp users.*** *Users of different clusters may choose to use this wrapper but will need to adjust the module load commands for equivalent.*

4.1 Command Line arguments

In order to run the wrapper we must provide the following command line arguments in the order shown:

- Argument 1: Sample input file – name_pre.txt or name_post.txt (**tab separated format**)
- Argument 2: Command e.g. stage of pipeline – 1/2/3/ etc.
- Argument 3: Optional Run Name (R analysis only)
- Argument 4: Optional Layouts File (R analysis only)

4.2 Submission

The bash wrapper can be run in the following way (from within the pipeline directory).

```
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list> <commands><opt argument 3> <opt argument 4>
```

Submission

1. *Stage 1: QC Sequences*

Here, you run the wrapper over the Illumina barcode-split files (from **spreadsheet A – pre.txt**) using the stage command-line argument [1]. This stage will split the Illumina-barcode split files into PCR-barcode split files.

```
python Processing_sequences_large_scale.py <sample file list pre>  1 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list pre >  1
```

2. *Stage 2: Read Preparation*

Here, you run the wrapper over the PCR-barcode split files (from spreadsheet B – post.txt) using the stage command-line argument [2]. The following steps will be performed:

a) Join forward and reverse reads (merging)
b) Split sequences according to UMI.
c) Identify RNA barcode and collapse/error correct based on groups of sequences sharing same barcode
d) Check isotype against reference
e) Check matches to IGHV/J reference sequences
f) Check open reading frame present

```
python Processing_sequences_large_scale.py <sample file list post >  2 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list post>  2
```

3. *Stage 3: Network Generation*

Here, you run the wrapper over the PCR-barcode split files (from spreadsheet B – post.txt) using the stage command-line argument [3].

- In summary: each vertex represents a different sequence, and the number of identical BCR sequences defines the vertex size. Edges are created between vertices that differ by one nucleotide. Clusters are groups of interconnected vertices (*1*). The program described here calculates edges between unique sequences and determines vertex sizes, creating output files in formats that can directly be used in network analysis programs such as networkx (python) or igraph (R or python).

```
python Processing_sequences_large_scale.py <sample file list post >  3 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list post >  3
```

4. *Stage 4: Annotation*

Here, you run the wrapper over the PCR-barcode split files (from spreadsheet B – post.txt) using the stage command-line argument [4]. The following steps will be performed:

a. Sequence annotation
b. Generation of broad repertoire statistics

```
python Processing_sequences_large_scale.py <sample file list post >  4 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list post >  4
```

5. Stage 5: Optional R analysis

We have written several R auxiliary functions that can be run on the output of stages 1-4 using the stage command-line argument [RS] or [RSB] if Jaccard library correction is required. You must provide a runname (argument 3) and if command = RSB the location of a layouts.txt file.
This is stage is only available for use with the BASH wrapper.
In Summary:

1. Concatenate files for all samples into summary files within the output directory.
2. Generate visual data summaries and save these in the output directory.
3. **Note:** Check the percentage of reads which pass Open-Read-Frame filtering.
    a. If this is abnormally low (potentially due to large clonal expansion) consider running the analysis with the ORF column in the sample sheet set to anything other than TRUE. This will prevent reads being removed by this filter.
4. Calculate and plot the Jaccard Index for all pairwise sample comparisons, to a minimum depth of 0.9*(minsamplereadcount >200). Indexes will be calculated on a) the full dataset and b) a reduced data set including only sequences >1 read identified. Note this can take several hours.
    a. If RSB a corrected Jaccard will also be calculated:
        i. Removing shared sequences from non-related samples which share an internal barcode, different illumina barcode and were sequenced on the **same lane** -> Index Hopping.
        ii. Removing shared sequences from non-related samples which share an internal barcode, different illumina barcode and **irrespective of sequencing lane** -> Index Hopping + library contamination.
5. You may need to adjust the width and height of the pdf files within the auxiliary functions depending on how many samples you have. The current settings work well with 80 samples (~ 1 lane of sequencing).

```
qsub -t 1 BCR_TCR_Wrapper_Cluster.sh <sample file list post >  RS <Runname > <layouts.txt>
```

6. Stage 6: File Reduction

Here, you run the wrapper over the PCR-barcode split files (from spreadsheet B – post.txt) using the stage command-line argument [5]. At this stage the finalised fastq files from each sample are concatenated into a fewer number of larger files named in the format:
Fully_reduced_<name of sample file list>_post<no.X>.txt.
You can find these batched files in OUTPUTDIR/ORIENTATED_SEQUENCES/NETWORKS.

```
python Processing_sequences_large_scale.py <sample file list post >  5 N Y Y
qsub -t 1  BCR_TCR_Wrapper_Cluster.sh <sample file list post >  5
```

7. Stage 7: IMGT Analysis

It is recommended at this point download the 'fully_reduced' fastqs from 6. and then to run the output files from **3.4.5** through IMGT/HighV-QUEST (http://www.imgt.org/HighV-QUEST/). This provides the gold standard annotation of BCR/TCR repertoire analysis. Use **the default submission parameters** and select csv and AIRR as the provided output format. You will receive an email when analysis has finished.
Note: File upload on IMGT can take ~30 minutes for large files, however there is no loading screen. The page will refresh to a summary page when upload is complete.

Download the completed files and upload to the cluster in the IMGT_RAW directory created below. Do not de-compress – this will be performed in the next stage.

To make the IMGT_RAW directory:

```
mkdir <outputdir>/ ORIENTATED_SEQUENCES/ANNOTATIONS/IMGT_RAW
```

8. Stage 8: Extracting IMGT Analysis

In order to extract per sample analysis from IMGT you must run the wrapper over the PCR-barcode split files (from spreadsheet B – post.txt) using the stage command-line argument [6]. The output of this stage will be saved in the IMGT_SPLIT directory.

```
python Processing_sequences_large_scale.py <sample file list>  6 N Y Y
qsub -t <1 >  BCR_TCR_Wrapper_Cluster.sh <sample file list>  6
```

Note: If you wish to repeat this stage without having to extract from IMGT again, edit Combine_extract_IMGT_information.py lines 55 and 56. Hash extract=TRUE and un-hash extract=FALSE.

Isotype Analysis scripts pending.

4. Output

1. **Table 1.** Description of network output files.

File	Description	Format
NETWORKS/Att_SAMPLE.txt	List of unique sequences.	Column 1 = List of unique sequence ids; column 2 = number of reads; column 3 = sequence;
NETWORKS/Cluster_identities_SAMPLE.txt	List of sequences within clusters.	Column 1 =sequence number; column 2 = cluster number; column 3 = sequence ID; column 4 = number of reads;
NETWORKS/Fully_reduced_SAMPLE.fasta	Fasta file of unique sequences.	Sequence ID multiplicity format;
NETWORKS/Plot_ids_SAMPLE.txt	List of sequences for plotting (only sequences that are connected or representing >1 read).	Column 1 =sequence number; column 2 = sequence ID; column 3 = number of reads;

2. **Table 2:** Description of annotation output files.

File	Description
ANNOTATIONS/Cluster_statistics_SAMPLE.txt	Cluster statistics
ANNOTATIONS/Constant_region_counts_SAMPLE.txt	Constant region counts
ANNOTATIONS/Distribution_cluster_sizes_SAMPLE.txt	Counts of cluster sizes
ANNOTATIONS/Distribution_vertex_sizes_SAMPLE.txt	Counts of vertiex sizes
ANNOTATIONS/Gene_frequencies_SAMPLE.txt	V/J gene frequencies
ANNOTATIONS/IsoTyper_chain_repertoire_statistics_file_SAMPLE.txt	Network parameters per isotype/constant region
ANNOTATIONS/Network_statistics_SAMPLE.txt	Network parameters (total)
ANNOTATIONS/TMP/Annotation_SAMPLE.txt	Annotation file
ANNOTATIONS/TMP/CDR3_frequencies_SAMPLE.txt	Distribution of CDR3 frequencies
ANNOTATIONS/TMP/CDR3_lengths_SAMPLE.txt	Distribution of CDR3 lengths

3. Output Fastq File Format:

To save memory and speed up analysis, output fastqs are stored in the dense multiplicity format described below:

- >IDOFSEQUENCE__X_Y_Z|IGX_IGY_IGZ
- GACGCATGATGCGTAGCAGACGGATATAGC…..

- Where the sequence header provides information of:
  - A unique sequence identifier
  - X, Y and Z correspond to the number of reads mapped to each constant region respectively.
- **Note:** the constant region has been trimmed from the sequence and this information is encoded in the header.

5. Author Contribution

Rachael J. M. Bashford-Rogers developed the python based TCR/BCR repertoire analysis pipeline and User Guide.
Lauren E. Overend developed the bash wrapper, R functions and helped write documentation.

6. References

If you find Immune-Network-Generation useful, please cite reference #2 (*R. J. Bashford-Rogers et al., 2019*).

- 1. R. J. Bashford-Rogers *et al.*, Network properties derived from deep sequencing of human B-cell receptor repertoires delineate B-cell populations. *Genome Res* **23**, 1874-1884 (2013).
- 2. R. J. M. Bashford-Rogers *et al.*, Analysis of the B cell receptor repertoire in six immune-mediated diseases. *Nature*, (2019).
- 3. W. Li, A. Godzik, Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22**, 1658-1659 (2006).
- 4. S. J. Watson *et al.*, Viral population analysis and minority-variant detection using short read next-generation sequencing. *Philos Trans R Soc Lond B Biol Sci* **368**, 20120205 (2013).
- 5. T. Magoc, S. L. Salzberg, FLASH: fast length adjustment of short reads to improve genome assemblies. *Bioinformatics* **27**, 2957-2963 (2011).

Part 2: Detailed Methods

1. Flash – Joining Overlapping PE Reads

Parameters:

- [FLASH] Min overlap: 15
- [FLASH] Max overlap: 280
- [FLASH] Max mismatch density: 0.250000
- [FLASH] Allow "outie" pairs: true
- [FLASH] Cap mismatch quals: false
- [FLASH] Combiner threads: 24
- [FLASH] Input format: FASTQ, phred_offset=33
- [FLASH] Output format: FASTQ, phred_offset=33

- **Min overlap set to 15** – based on Figure 6 (*5*). Provides a good balance between detecting correctly merged and incorrectly merged pairs.
- **Max overlap set to 280** – based on Paired End sequencing of 300bp (Total min fragment length would be about 320bp). May need to be adjusted if an alternative sequencing method is used.
- **Function:**
  - Processes each read separately and searches for the correct overlap.
  - If correct overlap is found, reads are merged into ONE extended read which is then used in downstream analysis.
  - Allows only un-gapped alignments as it targets Illumina sequencing platforms.

2. Establishing consensus sequences.

1 read detected with one UMI.	• Taken as consensus
2-15 reads detected with one UMI:	• Sequences aligned using MAFFT. • 80% similarity required in sequence or sequences discarded. • Select most common sequence as consensus.
15+ reads detected with one UMI:	• Cluster the sequences (CD-HIT) and pull out largest cluster. All other sequences discarded. • Sequences aligned using MAFFT. • If cluster bigger than 80% of reads take 80% similarity. • 80% confidence required in each base pair to pass filtering. • Select most common sequence

- **Note:** cDNA molecules that originate from different RNAs but with the same UMI will most likely result in the loss of reads from one (or more) RNA molecules. This can happen as despite

using a pool of UMIs of 12nt there can be preferential UMI usage resulting in the incorporation of the same UMI in multiple different RNA molecules.

3. Filtering Raw Reads

Summary of filtering stages from the output of FLASH (joined reads). Note that V and J gene matching is relatively relaxed as annotation will be performed later by IMGT, however this helps identify and filter out garbage sequences.

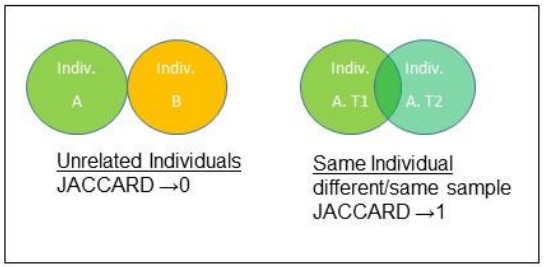
Filtering Stage	Description
N joined reads	After FLASH – number of reads which were joined.
N reads with UMIs	Total number of joined reads that have a UMI.
N uniq UMIs	Number of unique UMIs (i.e. RNA molecules captured). ***Deduplication to account for amplification biases.***
N reads post-V matching	Number of unique UMIs after filtering out sequences without any similarity to a V gene.
N reads post-J matching	Number of unique UMIs after filtering out sequences without any similarity to a J gene.
N BCR filtered (post ORF filtering)	Number of unique UMIs after filtering out sequences without functional ORF. ***This filter can be turned off.***
N unique BCRs	Number of unique sequences from the pool of unique UMIs. ***Account for multiple mRNA BCRs per cell (e.g. identical BCR but different constant region???***

4. IMGT Output

Summary of files produced by IMGT. Table table from: (you will need to log in to view).
http://www.imgt.org/IMGT_vquest/user_guide#Esummary

File number	File name	Number of columns filled	Results content *
#1	**"Summary"**	**33 (or 29)**	- Alignment score and identity percentage with the closest V and J genes and alleles, - D-REGION reading frame, - FR-IMGT and CDR-IMGT lengths, - Amino acid (AA) JUNCTION, - Description of insertions and deletions if any, - User sequence in the direct orientation, - Sequence orientation at the submission, the number of trimmed "n" before analysis if any, sequence length, sequence category.
#2	**"IMGT-gapped-nt-sequences"**	**18**	- Nucleotide (nt) sequences gapped according to the IMGT unique numbering for the labels V-D-J-REGION, V-J-REGION, V-REGION, FR1-IMGT, CDR1-IMGT, FR2-IMGT, CDR2-IMGT, FR3-IMGT, - nt sequences of CDR3-IMGT, JUNCTION, J-REGION and FR4-IMGT.
#3	**"nt-sequences "**	**118 (57 (V-J), 79 (1D), 91 (2D) 103 (3D))**	- nt sequences of all labels that can be automatically described and delimitated by IMGT/Automat (57 columns for IGL, IGK, TRA and TRG sequences, 79 (if one D), 91 (if two D) or 103 (if 3 D) columns for IGH, TRB and TRD sequences). The 3 last columns evaluate the number of missing nt for partial V-(D)-J-REGION and of uncertain nt in V-REGION
#4	**"IMGT-gapped-AA-sequences"**	**18**	- AA sequences gapped according to the IMGT unique numbering for the labels V-D-J-REGION, V-J-REGION, V-REGION, FR1-IMGT, CDR1-IMGT, FR2-IMGT, CDR2-IMGT, FR3-IMGT, - AA sequences of CDR3-IMGT, JUNCTION, J-REGION and FR4-IMGT.
#5	**"AA-sequences"**	**18**	Same columns as "IMGT-gapped-AA-sequences" (#4), but sequences of labels are without IMGT gaps.
#6	**"Junction"**	**84 (36 (V-J), 50 (1D), 62 (2D), 77 (3D)**	- Results of IMGT/JunctionAnalysis (36 columns for IGL, IGK, TRA and TRG sequences, 50 (if one D), 62 (if two D) or 77 (if 3 D) columns for IGH, TRB and TRD sequences).
#7	**"V-REGION-mutation-and-AA-change-table"**	**11**	- List of mutations (nt mutations, AA changes, codon change, hotspot motifs, AA class identity (+) or change (-)) for V-REGION, FR1-IMGT, CDR1-IMGT, FR2-IMGT, CDR2-IMGT, FR3-IMGT and germline CDR3-IMGT.
#8	**"V-REGION-nt-mutation-statistics "**	**130**	- Number (nb) of nt positions including IMGT gaps, nb of nt, nb of identical nt, total nb of mutations, nb of silent mutations, nb of nonsilent mutations, nb of transitions (a>g, g>a, c>t, t>c) and nb of transversions (a>c, c>a, a>t, t>a, g>c, c>g, g>t, t>g) for V-REGION, FR1-IMGT, CDR1-IMGT, FR2-IMGT, CDR2-IMGT, FR3-IMGT and germline CDR3-IMGT.
#9	**"V-REGION-AA-change-statistics "**	**109**	- nb of AA positions including IMGT gaps, nb of AA, nb of identical AA, total nb of AA changes, nb of AA changes according to AAclassChangeType (+++, ++-, +-+, +--, -+-, --+, ---), and nb of AA class changes according to AAclassSimilarityDegree (nb of Very similar, nb of Similar, nb of Dissimilar, nb of Very dissimilar) for V-REGION, FR1-IMGT, CDR1-IMGT, FR2-IMGT, CDR2-IMGT, FR3-IMGT and germline CDR3-IMGT.
#10	**" V-REGION-mutation-hotspots "**	**8**	- Hot spots motifs ((a/t)a, t(a/t), (a/g)g(c/t)(a/t), (a/t)(a/g)c(c/t)) detected in the closest germline V-REGION with positions in FR-IMGT and CDR-IMGT.
#11	**"Parameters "**		- Date of the analysis, - IMGT/V-QUEST programme version, IMGT/V-QUEST reference directory release, - Parameters used for the analysis: species, receptor type or locus, IMGT reference directory set and Advanced parameters.
#12	**"scFv "**	40	Available only for Advanced functionalities, Analysis of single chain Fragment variable (scFv). - positions and length, CDR_length, JUNCTION for the 2 V-DOMAIN of the scFv

JACCARD INDEX: Sample Mismatch/Contamination

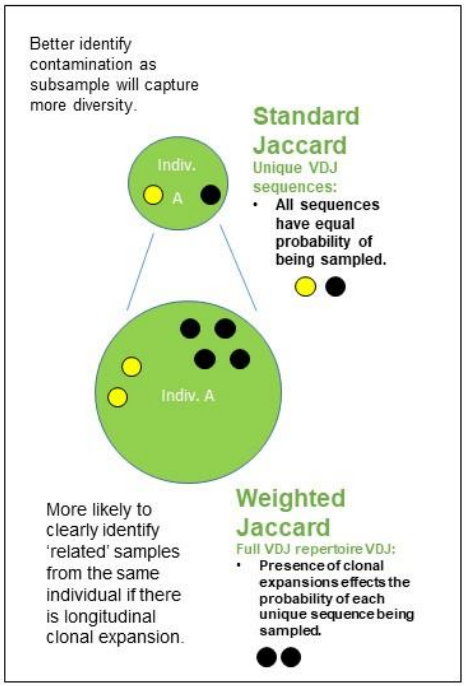


Hypothesis: Repertoires from the same individual (even over successive time-points) should be more similar than from different individuals.

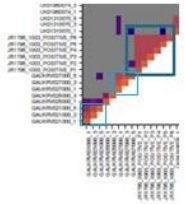
Sample size (sequencing depth) can have a significant effect on repertoire statistics. Therefore we calculate Jaccard indexes on a) the full dataset and b) a **subsample** (without replacement), repeated **x10,000** and take an **average** of the Jaccard index.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Size of the intersection divided by the size of the union of the sample sets.



- **Page 1:** SharedSeq.MeanSubsample + log(SharedSeq.MeanSubsample):
  - Mean number of shared sequences between two sub-samples x10000
- **Page 2:** Jaccard Mean Subsample + log(Jaccard Mean Subsample):
  - Mean jaccard index between two sub-samples x10000
- **Page 3:** SharedSeq.Full + log(SharedSeq.Full):
  - Number of shared sequences between full data set – no subsampling so sensitive to sequencing depth effect.
- **Page 4:** Jaccard.Full + log(JACCARD_full)
  - The jaccard index on the full data set – no subsampling so sensitive to sequencing depth effect.
- **Pages 5-8:** repeats pages 1-4 using statistics calculated on the weighted repertoire.



Identification of samples from same individual vs different individuals using log(Jaccard Mean Subsample). *Visualize sample mismatch/contamination.*