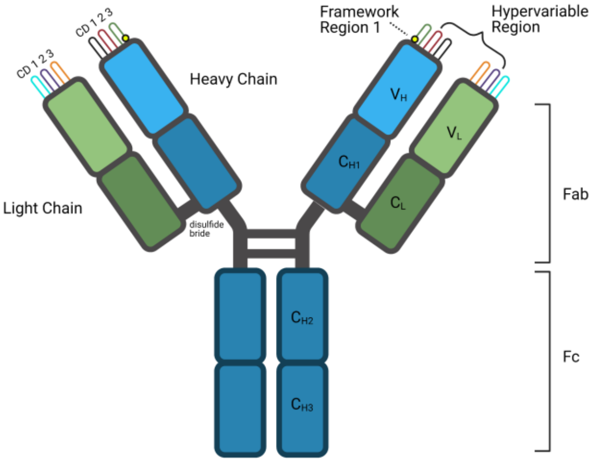


BCR/TCR ANALYSIS PIPELINE

Developed by Rachael J. M. Bashford-Rogers

Wellcome Trust Centre for Human Genetics, Oxford

Rachael Bashford-Rogers
rbr1@well.ox.ac.uk

BCR/TCR PROCESSING PIPELINE DOCUMENTATION

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

1. Introduction

This manual provides an outline of the **BCR/TCR processing pipeline for NGS data** based on the IsoTyper and TCR protocols developed in the Bashford-Rogers Lab. For handling multiple samples we provide two solutions: 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 could be easily adapted for another cluster architecture.

A summary of the pipeline is described below:

Stage 1:

1. QC sequences

Stage 2:

1. Join forward and reverse reads (merging)
2. Split sequences according to sample barcode
3. Identify RNA barcode and collapse/error correct based on groups of sequences sharing same barcode
4. Check isotype against reference
5. Check matches to IGHV/J reference sequences
6. Check open reading frame present

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. Sequence annotation
2. Network Analysis
3. Generation of broad repertoire statistics

Stage 5 (optional):

1. Run the optional R script to analyse and visualise summary metrics from the results of Stages 1-4. This will also concatenate files for all samples.
2. 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. To prevent reads being removed.

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 lab protocol for BCR repertoire amplification.

- 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 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 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 so sequences incorporate an Illumina barcode (1 of 96 unique barcodes). Libraries with different Illumina barcodes can then be pooled.
- Libraries are sequenced on a MiSeq using 300bp Paired End technology.

The protocol for TCR repertoire amplification differs only in primer design.

Created with BioRender.com

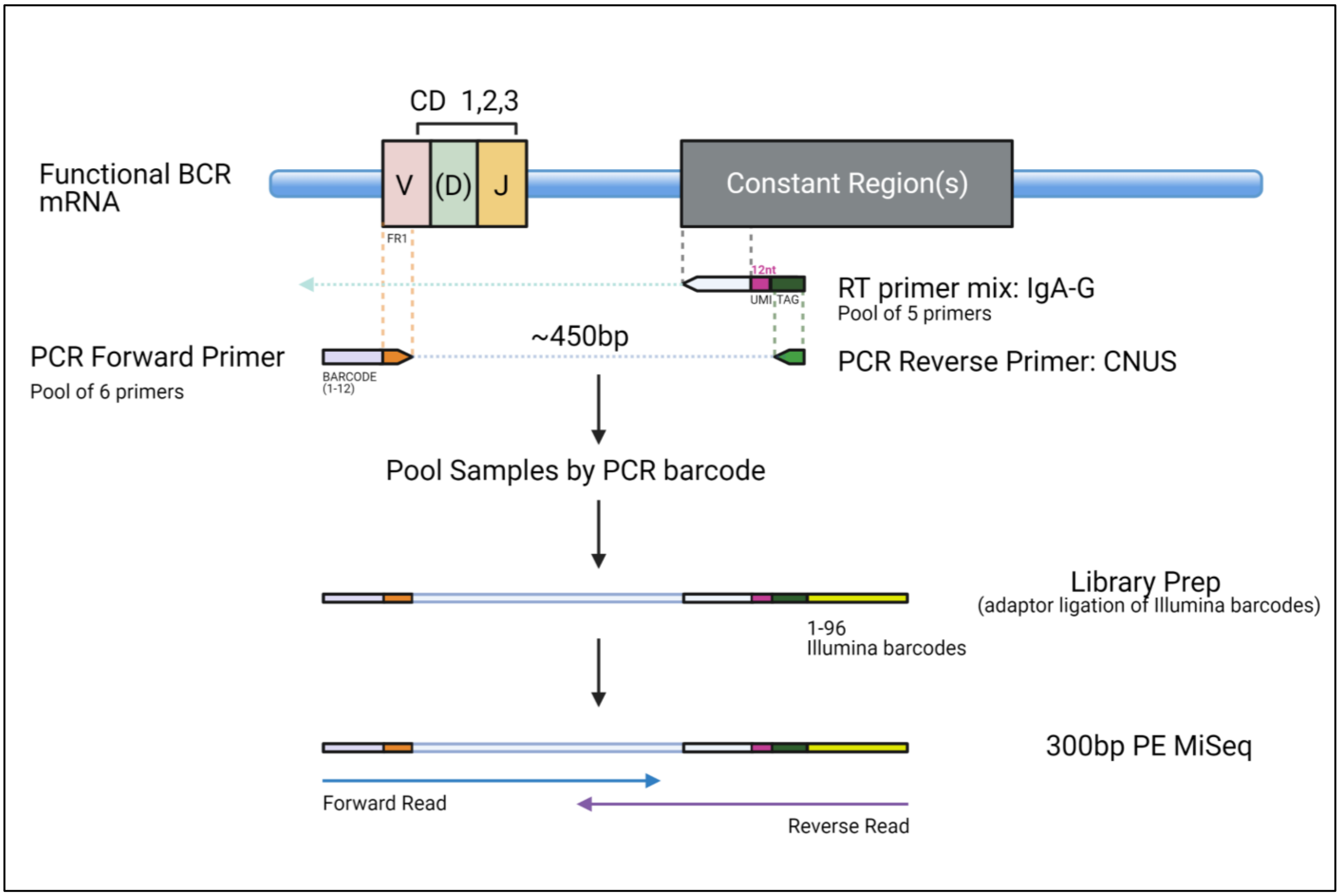
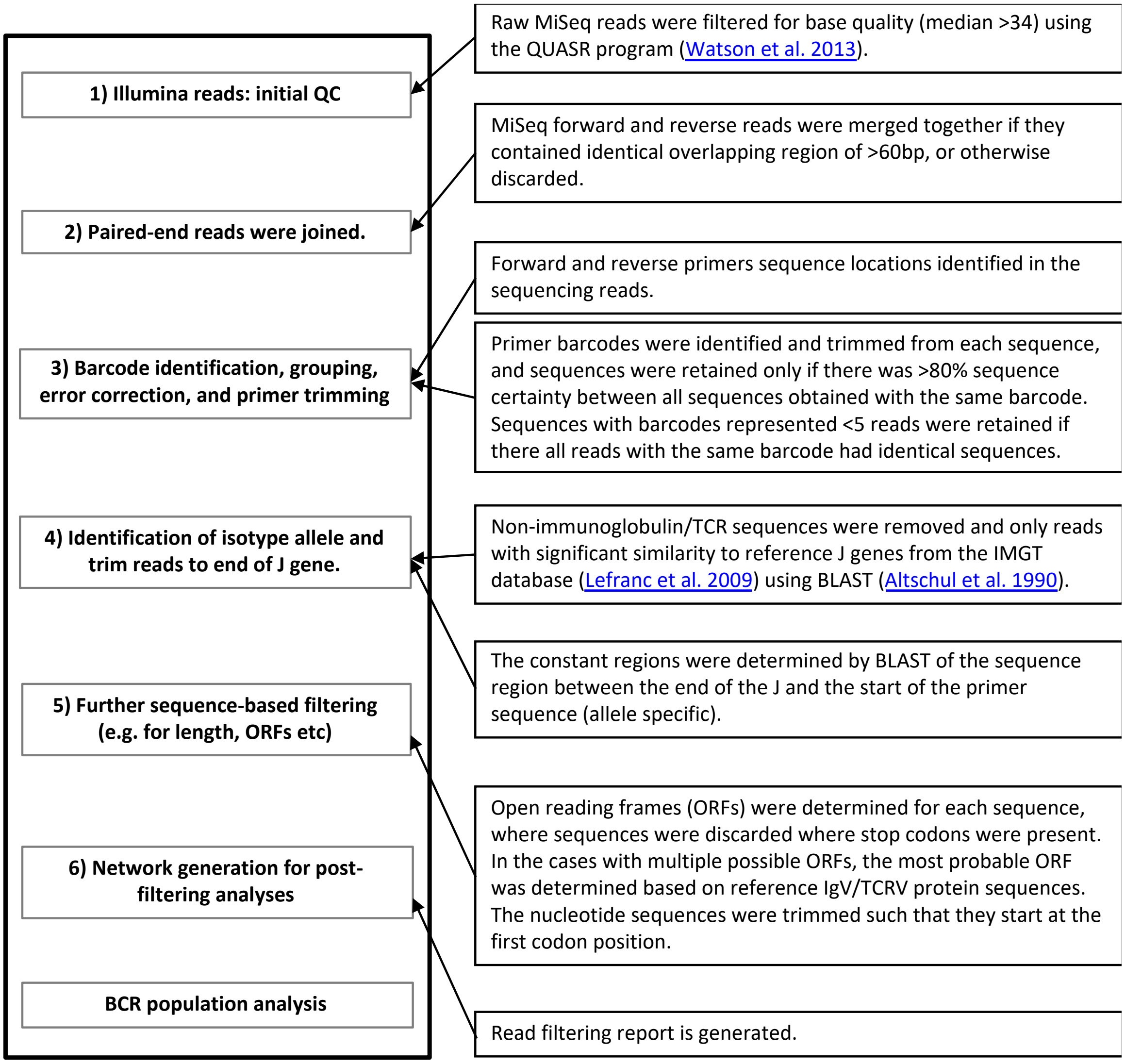


Figure 3. Schematic of QC and filtering pipeline.


1) Illumina reads: initial QC
2) Paired-end reads were joined.
3) Barcode identification, grouping, error correction, and primer trimming
4) Identification of isotype allele and trim reads to end of J gene.
5) Further sequence-based filtering (e.g. for length, ORFs etc)
6) Network generation for post-filtering analyses
BCR population analysis

Raw MiSeq reads were filtered for base quality (median >34) using the QUASR program (Watson et al. 2013).

MiSeq forward and reverse reads were merged together if they contained identical overlapping region of >60bp, or otherwise discarded.

Forward and reverse primers sequence locations identified in the sequencing reads.

Primer barcodes were identified and trimmed from each sequence, and sequences were retained only if there was >80% sequence certainty between all sequences obtained with the same barcode. Sequences with barcodes represented <5 reads were retained if there all reads with the same barcode had identical sequences.

Non-immunoglobulin/TCR sequences were removed and only reads with significant similarity to reference J genes from the IMGT database (Lefranc et al. 2009) using BLAST (Altschul et al. 1990).

The constant regions were determined by BLAST of the sequence region between the end of the J and the start of the primer sequence (allele specific).

Open reading frames (ORFs) were determined for each sequence, where sequences were discarded where stop codons were present. In the cases with multiple possible ORFs, the most probable ORF was determined based on reference IgV/TCRV protein sequences. The nucleotide sequences were trimmed such that they start at the first codon position.

Read filtering report is generated.

2. Installation

Note: Jobs should be run in the directory containing the pipeline so that relative paths are used.

- Ensure access to Local_immune_repertoire_annotator_1.0.py from:
  - ANNOTATION_OF_TCRs_CDR3_REGIONS/Local_immune_repertoire_annotator_1.0.py

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
- **For users of Rescomp (BMRC, Oxford) please not this is already installed so you do not need to perform this stage.**

5. Python Modules:

Ensure that the following python modules are installed.

Note for 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****.***

- Sys
- Collections
- Os
- Operator
- networkx

6. R Modules (optional analysis):

Ensure that the following R modules are installed:

- tidyverse
- ggplot2
- foreach
- doParallel
- gridExtra

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 current working directory*** *(directory containing pipeline). However this must 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
```

3. Run

This pipeline performs the steps described in the introduction, split into several separate stages. Each stage can be run independently. 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.

3.1 Sample Spreadsheets:

We provide two example files within the pipeline directory to get started. As this pipeline allows for the potential of double barcoding (Illumina and sample barcodes) you must create two input spreadsheets according to details below.
Note: spreadsheet 1 and spreadsheet 2 will be the same if there is no internal barcoding, see example input file "Sample_example_no_internl_barcodes.txt".

3.1.1 Spreadsheet A - 'name_pre.txt':

- A spreadsheet detailing the library pools (i.e. 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). Example sheet: Samples_WHG_Trial1_Iso_pre1.txt
    - **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.**

3.1.2 Spreadsheet B - 'name_post.txt':

- A spreadsheet detailing the individual samples, linked to the library pools from which they come. 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). Example sheet: Samples_WHG_Trial1_Iso_post1.txt

- ○ **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.**

3.2 Python Wrapper

We provide a python wrapper, Processing_sequences_large_scale.py, which will manage the job submission for multiple samples using bsub for any stage of the pipeline. *Note this is not recommended for Rescomp users.*

3.2.1 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

3.2.2 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 is Y, then no analysis will be performed.*

3.3 BASH Wrapper

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

3.3.1 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.

3.3.2 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>
```

Note: *Remember to create the COMMANDLOGS directory within the working directory prior to submission.*

3.4 Run Pipeline

3.4.1 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>  1 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list>  1
```

3.4.2 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>  2 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list>  2
```

3.4.3 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>  3 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list>  3
```

3.4.4 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>  4 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list>  4
```

3.4.5 Stage 5: Optional R analysis

At this stage it is strongly recommended that you visualise the results of the pipeline. To do this we have written several R auxiliary functions that can be called within an interactive R session. These functions require, as arguments, the output directory from the samples file and a run name e.g. "BCR_Lane1". You may also chose to specify the number of worker nodes, default is 5.

1. This will concatenate files for all samples into summary files within the working directory.
2. This will plot some summary plots for visualising data.
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.

3.4.6 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 fastaq files from each sample are concatenated into a fewer number of larger files named reduced named in the format:
Sampling_depth_per_isotype_<name of sample file list>.txt

```
python Processing_sequences_large_scale.py <sample file list>  5 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list>  5
```

3.4.7 Stage 7: IMGT Analysis

It is recommended at this point 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.

3.4.8 Stage 8: Isotyper Analysis

NEED TO DESCRIBE.

```
python Processing_sequences_large_scale.py <sample file list>  6 N Y Y
qsub -t <1-no. samples>  BCR_TCR_Wrapper_Cluster.sh <sample file list>  6
```

4. Output

4.1 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;

4.2 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

4.3 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 IgX, IgY and IgZ 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).