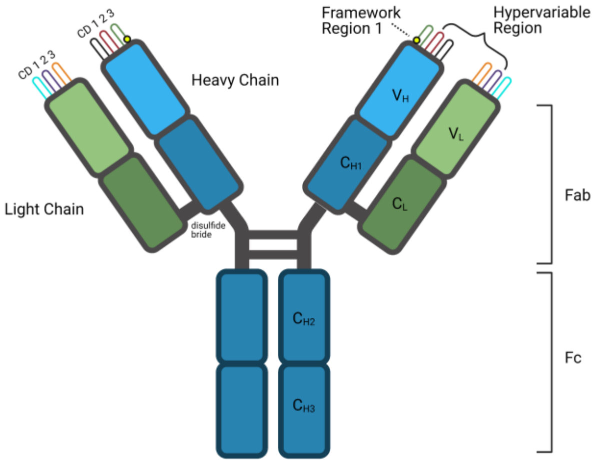


BCR/TCR ANALYSIS PIPELINE

Developed by Rachael J. M. Bashford-Rogers

Wellcome Trust Centre for Human Genetics, Oxford

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

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 4

1. INTRODUCTION 4
- *1. Register for IMGT* 6

2. INSTALLATION 7
- *1. FLASH:* 7
- *2. CD-HIT(3):* 7
- *3. Quasr(4):* 7
- *4. Blast:* 7
- *5. Python Modules:* 7
- *6. R Modules (optional analysis):* 7
- *7. Locations of Dependencies:* 8
- *8. Create Log Files Directory* 8

3. RUN 8
- *3.1 Sample Spreadsheets:* 8
  - 3.1.1 Spreadsheet A - 'name_pre.txt' 8
  - 3.1.2 Spreadsheet B - 'name_post.txt' 9
  - 3.2.3 Batch Files 9
- *3.2 Python Wrapper* 9
  - 3.2.1 Command Line arguments 9
  - 3.2.2 Submission 9
- *3.3 BASH Wrapper* 10
  - 3.3.1 Command Line arguments 10
  - 3.3.2 Submission 10
- *3.4 Pipeline Manager* 10
- *3.5 Run Pipeline* 11
  - 3.5.1 Stage 1: QC Sequences 11
  - 3.5.2 Stage 2: Read Preparation 11
  - 3.5.3 Stage 3: Network Generation 11
  - 3.5.4 Stage 4: Annotation 12
  - 3.5.5 Stage 5: Optional R analysis 12
  - 3.5.6 Stage 6: File Reduction 12
  - 3.5.7 Stage 7: IMGT Analysis 12
  - 3.5.8 Stage 8: Extracting IMGT Analysis 13
  - 3.5.9 Stage 9: Setting Subsample Depth 13
  - 3.5.10 Stage 10: Running Isotyper Analysis 13

1. OUTPUT 14
- *1.1 Table 1. Description of network output files.* 14
- *1.2 Table 2: Description of annotation output files.* 14
- *1.3 Output Fastq File Format:* 15

2. AUTHOR CONTRIBUTION 15

3. REFERENCES 15

PART 2: DETAILED METHODS ... 16

1. FLASH – JOINING OVERLAPPING PE READS ... 16
2. ESTABLISHING CONSENSUS SEQUENCES. ... 16
3. FILTERING RAW READS ... 17
4. IMGT OUTPUT ... 18

Part 1

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

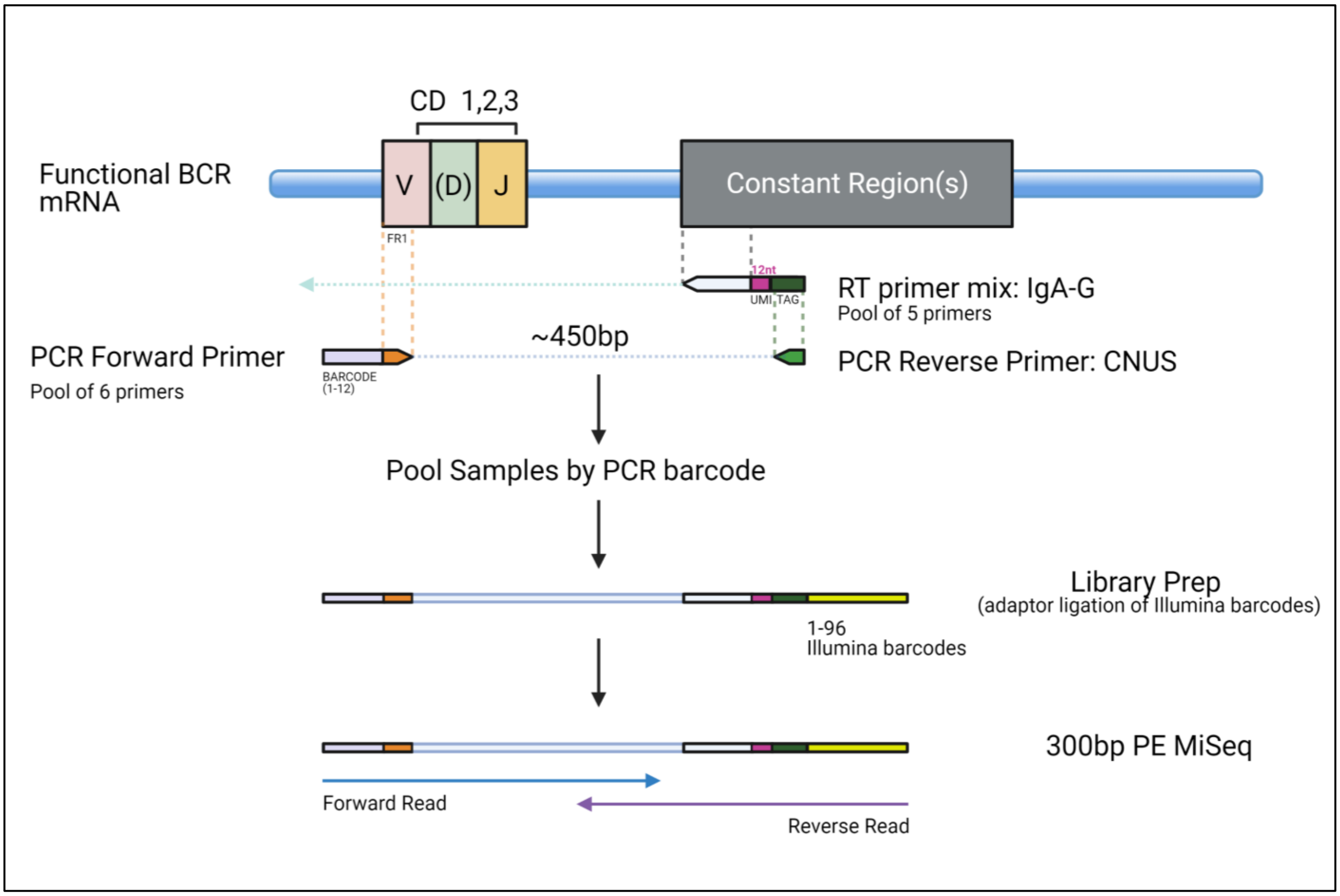
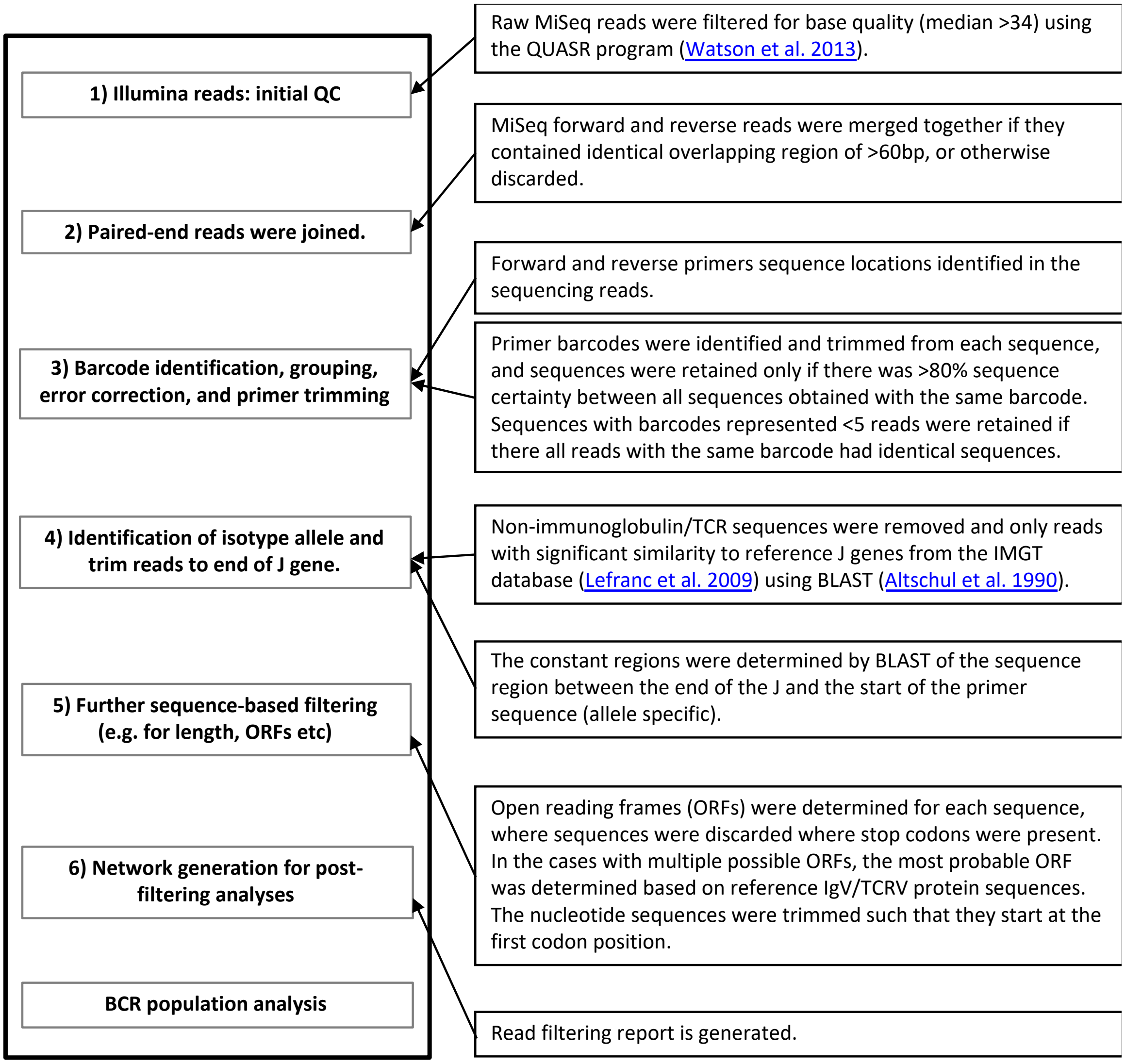


Figure 3. Schematic of QC and filtering pipeline.



1) Illumina reads: initial QC

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

2) Paired-end reads were joined.

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

3) Barcode identification, grouping, error correction, and primer trimming

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.

4) Identification of isotype allele and trim reads to end of J gene.

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).

5) Further sequence-based filtering (e.g. for length, ORFs etc)

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.

6) Network generation for post-filtering analyses

Read filtering report is generated.

BCR population analysis

1. Register for IMGT

As part of the analysis pipeline fatsq files will be annotated using IMGT/HighV-Quest – the gold standard for TCR and BCR repertoire analysis. However to use the high-throughput (HighV) version you must register a user account. All new users must be approved and accounts are then activated by an administrator. Therefore we recommend registering as soon as possible to avoid hold-ups later on.

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

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	data.table
ggplot2	ggforce
Gviz	foreach
gridExtra	doParrallel
cowplot	gtools
optparse	

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.**

3.2.3 Batch Files

A tab separated txt file detailing the project experimental design:

- SampleID
- Barcode (the same as SampleID unless there are multiple samples from the same individual e.g blood and biopsy. Naming convention: "Individual"_"Sample")
- Lane – Sequencing Lane
- Plate – PCR plate
- Library – Illumina library e.g. A1
- Position – Location on PCR Plate
- PCRBarcode – Internal PCR barcode A-L

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 (**tab 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 Pipeline Manager

We provide a **bash pipeline manager** to automate job submission on Rescomp. This could be adapted for a similar linux based cluster architecture.

		Stage Command
1-5+RS	pre-IMT	**1**
1-5+RS+basic Jaccard	pre-IMT	**2**
1-5+RS+basic and consensus Jaccard	pre-IMT	**3**
6 + isotyper scripts (productive, unproductive and all) (post IMGT)	Post IMGT	**4**
Isotyper scripts (productive, unproductive and all)	Post IMGT	**5**

Jobs will be held in the queue until the previous job dependencies have been run to completion. To ensure jobs will **only run** if prior jobs dependencies have been run successfully to completion without error, each array task will output the sample ID to a single summary file upon **successful** completion, which can be inspected. Only if all samples are found in this file will the subsequent jobs run. In the case of a task failure a list of failed samples will be output to a corresponding *task*FAILED_SAMPLES.txt file which can be inspected. Usually failures result from a sample having 0 reads post filtering. **Dependency checking is specified using the argument Y (recommended) or turned off using N.**

All log files will be output to a named directory within COMMANDLOGS while the 'check files' containing sample id will be output directly to the COMMANDLOGS directory. To run the pipeline manager enter the below code from within the cloned code directory.

The pipeline manager is recommended when running multiple batches/jobs with large numbers of samples as it can be challenging to manually keep track of job ids, errors and completion status when running each stage independently. Automation decreases run time as there is no user submission delay.

```
./BCR_TCR_PIPELINEMANAGER.sh  <Dependency check Y/N> <sample file pre> <sample file post> <stage> <run name> <optional batch file>
```

3.5 Run Pipeline

3.5.1 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
```

3.5.2 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
```

3.5.3 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
```

3.5.4 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
```

3.5.5 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 run on the files from stages 1-4 using the stage command-line argument [RS]. **This is stage is only available for use with the BASH wrapper.** You have the option of providing a run-name as a 3rd command line argument.

1. This will concatenate files for all samples into summary files within the output directory.
2. This will also plot some summary plots for visualising data 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. You may need to adjust the width and height of the pdf files within the auxillary 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 <OptionalRunname >
```

3.5.6 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:
ORIENTATED_SEQUENCES/NETWORKS/Fully_reduced_<name of sample file list>_post<no.X>.txt

```
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
```

3.5.7 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. Use **the default submission parameters** and select csv and AIRR as the provided output format. You will receive an email when analysis has finished. Download the completed files and upload to the cluster. 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
```

Save the IMGT files output files in:
OUPUTDIR/ORIENTATED_SEQUENCES/ANNOTATIONS/IMGT_RAW

3.5.8 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]. Remember to save the IMGT files (compressed) in the IMGT_RAW directory from 3.4.7. The output of this stage will be saved in IMGT_SPLIT.

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

3.5.9 Stage 9: Setting Subsample Depth

Prior to running the isotyper (productive/non-produtive/combined) anlalysis you must first adjust the subsample depth by editing the **'/ORIENTATED_SEQUENCES/ANNOTATIONS/Sampling_depth_per_isotype_'** files. Adjust the value in the 'min' column for all listed constant regions – this will be used as the subsample depth for each constant region. The 5th percentile often offers a good compromise between sample-loss and read depth however this is very much data-set dependant. If a sample does not have the 'min' number of reads set metrics will be set to -1.

3.5.10 Stage 10: Running Isotyper Analysis

After setting the subsampling depth you may then proceed with running the isotyper analysis script which will calculate a variety of different metrics from the IMGT analysis, using downsampling to correct for the correlation between read depth. This may be run on either the productive and non-productive reads combined using the stage argument ISO1, or on either productive or unproductive reads only using the arguments ISO1_PRODUCTIVE and ISO1_NON_PRODUCTIVE respectively. Recommended to run all 3. You can rerun this stage using different user specified read depths, provided that each time you rerun it you rename the ISOTYPER directory to something unique e.g. ISOTYPER_TRY1 to prevent the directory being overwritten. In this way you can have multiple directories containing metrics calculated at different read depths. This is beneficial as in some cases the user specified read depth is not sufficiently large to negate the effect of read depth.

```
python Processing_sequences_large_scale.py <sample file list>  ISO1 N Y Y
qsub -t <1 >  BCR_TCR_Wrapper_Cluster.sh <sample file list post >  ISO1
```

1. Output

1.1 **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;

1.2 **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

1.3 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.

2. 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.

3. 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 than80% 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 UMI).***

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