ERVsearch

Release 1.0.0

Katy Brown

Dec 11, 2020

CONTENTS

CHAPTER

ONE

INTRODUCTION

ERVsearch is a pipeline for identification of endogenous retrovirus like regions in a host genome, based on sequence similarity to known retroviruses.

ERVsearch screens for endogenous retrovirus (ERV) like regions in any FASTA file using the Exonerate algorithm (Slater and Birney, 2005, doi:10.1186/1471-2105-6-31).

- In the **Screen** section, open reading frames (ORFs) resembling retroviral *gag*, *pol* and *env* genes are identified based on their level of similarity to a database of known complete or partial retroviral ORFs.
- In the **Classify** section, these ORFs are classified into groups based on a database of currently classified retroviruses and phylogenetic trees are built.
- In the **ERVRegions** section, regions with ORFs resembling more than one retroviral gene are identified.

This is a updated and expanded version of the pipeline used to identify ERVs in Brown and Tarlinton 2017 (doi: 10.1111/mam.12079), Brown et al. 2014 (doi: 10.1128/JVI.00966-14), Brown et al. 2012 (doi: j.virol.2012.07.010) and Tarlinton et al. 2012 (doi: 10.1016/j.tvjl.2012.08.011). The original version is available here as a Perl pipeline and was written by Dr Richard Emes.

CHAPTER

TWO

PREREQUISITES

The pipeline is currently available for Unix-based systems only.

The ERVsearch pipeline requires the following freely available software. All packages are available via pip and easy_install

Python 3.5+ with the following packages:

- ruffus
- numpy
- pandas
- ete3
- matplotlib

The following commonly used software needs to be installed and in your $PATH

- Samtools
- Bedtools
- Emboss
- Mafft
- FastTree

The following software also needs to be installed

- Exonerate
- Usearch

CHAPTER

THREE

INSTALLATION

The latest release of ERVsearch is available via pip3.

```
pip3 install ERVsearch
```

Alternatively, if you prefer to install directly, the latest release can be downloaded from Github.

The latest (beta) version can also be cloned from github

```
git clone https://github.com/KatyBrown/ERVsearch.git
```

No compliation is required, just add the ERVsearch directory to your path or use the full path to `ERVsearch/ERVsearch`. If installed using pip you should be able to call `ERVsearch` directly.

CHAPTER

FOUR

QUICK START

After cloning the repository, the program can be used as is (with the above prerequisites installed).

1. Make a copy of the pipeline.ini file (ERVsearch/templates/pipeline.ini or here) in your working directory (the directory in which you would like to store the output).
2. Download a local copy of your genome (or other sequence) of interest as a single FASTA file.
3. Edit your copy of pipeline.ini to configure the pipeline for your computer:

- Add the path to the genome you want to screen (the fasta file in step 2) to the genome section e.g. hg38.fa saved in /home/myname/genome/hg38.fa would require the following options:

```
[genome]
file_=/home/myname/genome/hg38.fa
```

- Add the paths to usearch and exonerate to the paths section e.g.

```
[paths]
path_to_usearch=/home/myname/usearch/usearch11.0.667_i86linux32
path_to_exonerate=/home/myname/exonerate/bin/exonerate
```

- Run the pipeline in your working directory as:

```
ERVsearch --target_tasks full -v5
```

CHAPTER

FIVE

PIPELINE DESCRIPTION

The pipeline is designed to identify regions resembling retroviral *gag*, *pol* and *env* genes in a genome (or other set of sequences) and to perform various analyses on these regions.

It is divided into three sections:

5.1 Screen

function documentation

- Screens the genome for ERV like regions by comparing the genome to a set of known retroviral ORFs using Exonerate.
- Confirms the Exonerate regions using UBLAST
- Finds and confirms ORFs within these regions
- Finds the most similar known retroviral ORF in the database to each of the newly identified ORFs

5.2 Classify

function documentation

- Classifies the newly identified ORFs into groups based on the most similar known ORF
- Aligns the newly identified ORFs with reference sequences within these groups and builds a phylogenetic tree for each group.
- Finds clusters of newly identified ORFs within these trees
- Incorporates representative sequences from these clusters into a summary tree for each retroviral gene and genus (based on classification into *gamma*, *beta*, *spuma*, *alpha*, *lenti*, *epsilon* and *delta* retroviruses as definied by the ICTV (https://talk.ictvonline.org/taxonomy).

5.3 ERVRegions

function documentation

- Identifies regions of the genome containing ORFs resembling more than one different retroviral gene within a certain distance

All functions in all sections are described in detail in the functions section.

CHAPTER

SIX

INPUT FILES

6.1 Required Input Files

1: *FASTA file to screen for ERVs*

The main input file is sequence file in FASTA format containing DNA sequences from the genome which you wish to screen for ERV-like regions. This would usually be a reference or de novo assembled genome but can be any set of DNA sequences.

Reference genome sequences are available from Ensembl and UCSC (amongst others).

To be used as an input file, the reference genome needs to be contained in a single FASTA file.

For Ensembl genomes, this would usually be the `GENOMEID.dna.toplevel.fa.gz`file from the "download DNA sequence" page for the appropriate organism, substituting `GENOMEID` for the genome ID (e.g. GRCh38)

For UCSC genomes this would be `GENOMEID.fa.gz` from the bigZips directory for this organism on the FTP server, substituting `GENOMEID` for the genome ID (e.g. hg38)

It is possible to use a gzipped or zipped file, in which case the filename needs to end with .gz or .zip respectively.

2: `pipeline.ini` file

This file is a configuration file in ini format containing the parmeters you wish to use. This file needs to be in your working directory - the folder in which you wish to run ERVsearch.

A template `pipeline.ini` file should be used and edited - this file is available as `ERVsearch/templates/pipeline.ini` or here

Options specified as `!?` are required, all others have a default value.

Required parameters

Optional parameters

6.2 Optional Input Files

6.2.1 Custom Databases

By default, ERVsearch will use the provided database of 774 ERV nucleotide sequences and corresponding amino acid sequences as a query against the provided genome. This database is designed to be representative of known retroviruses and to identify the majority of ERVs. However, a more specific custom database can also be provided and used for the initial screen.

To do this, the database_use_custom_db parameter in the pipeline.ini can be set to `True`. The query sequences should be stored as FASTA files of amino acid sequences, with one file per retroviral gene. Only gag, pol and env genes are

currently supported. Very short sequences (less than ~100 amino acids) should be avoided where possible. The paths to these files are then specified in the database section of the pipeline.ini

e.g.

```
[database]
use_custom_db=True
gag=/home/katy/my_databases/gag_ervs.fasta
pol=/home/katy/my_databases/pol_ervs.fasta
env=/home/katy/my_databases/env_ervs.fasta
```

Currently, custom databases are only used for the initial Exonerate screen and UBLAST check, after this all classification based steps will use the default databases, as these sequences have been classified into subgroups for phylogenetic analysis.

6.2.2 Sequence List

`keep_chroms.txt`

A list of chromosome names to include. The names should the names in the fasta file cropped at the first space, e.g. "NW_006711271.1 Panthera tigris altaica isolate TaeGuk unplaced genomic scaffold" should just be listed as NW_006711271.1. The names should be listed with one name per line, are case sensitive and need to be identical to those in the fasta file. This file needs to be named keep_chroms.txt and in the working directory.

CHAPTER

SEVEN

DATABASES

Several sets of reference sequences are provided as part of this package.

For each retroviral gene (gag, pol and env), a representative set of retroviral open reading frames has been selected from NCBI Genbank and various publications.

These files are provided as:

`ERVsearch/ERV_db/gag.fasta` - *gag* gene amino acid sequences `ERVsearch/ERV_db/pol.fasta` - *pol* gene amino acid sequences `ERVsearch/ERV_db/env.fasta` - *env* gene amino acid sequences `ERVsearch/ERV_db/all_ERVs_nt.fasta` - all ORFs as nucleotide sequences `ERVsearch/ERV_db/all_ERVs_aa.fasta` - all ORFs as amino acid sequences

A number of subsets of sequences are also provided to use in phylogenetic analysis.

These are:

`ERVsearch/phylogenies/group_phylogenies/*fasta`

Small groups of nucleotide sequences from the ORF database which are closely related, selected manually as representatives of these groups based on prior knowledge, sequence similarity and phylogenetic analysis. Newly identified sequences are assigned to these groups where possible.

`ERVsearch/phylogenies/summary_phylogenies/*fasta`

Broader groups of nucleotide sequences from the ORF database for each gene (gag, pol and env) and each genus (gamma, beta, delta, alpha, epsilon, lenti and spuma). Newly identified sequences are incorporated into phylogenetic trees based on these sequences, plus more closely related sequences from the `group_phylogenies` fasta files.

Two addtional files are also provided:

`ERVsearch/ERV_db/convert.tsv`

Table showing the group each reference sequence belongs to.

`ERVsearch/phylogenies/outgroups.tsv`

Table providing the name of an appropriate outgroup for each phylogeny.

If you want to see these files (ERVsearch will locate them automatically for internal use), then if you are using a clone of the git repository, database files can be found in `ERVsearch/ERV_db` and `ERVsearch/phylogenies`. If you installed using pip, they will be in the same location in the ERVsearch directory in your python site-packages directory.

CHAPTER

EIGHT

USAGE

8.1 Running the Pipeline

The pipeline is implemented using the pipeline development package ruffus, (Goodstadt 2010, doi:10.1093/bioinformatics/btq524).

To run the full pipeline, the following command is used:

```
ERVsearch --target_tasks full
```

If the pipeline stops or fails at any point, it will restart from after the previous fully completed step (based on the presence of the output files from this step). If the output of an earlier step in the pipeline has a more recent timestamp than the output of a later step, the later step will be rerun.

Sections of the pipeline can be run as follows:

```
ERVsearch --target_tasks Screen
```

Screen with Exonerate and check the results with UBLAST, find ORFs and find the most similar known retroviral ORF.

```
ERVsearch --target_tasks Classify
```

Run the *Screen* steps and then sort the sequences into subgroups, build phylogenetic trees for these groups and a summary tree for each gene and genus.

```
ERVsearch --target_tasks ERVRegions
```

Run the *Screen* steps and then find regions with ORFs resembling more than one retroviral gene in close proximity.

```
ERVsearch --target_tasks full
```

Run all the above sections.

You can also run the pipeline up until any specific function - any function name can be provided for the `target_tasks` parameter.

For example to run up until the end of the function `classifyWithExonerate`, use the following command.

```
ERVsearch --target_tasks classifyWithExonerate
```

All functions prior to this function will run if needed.

8.2 Parallelisation

The pipeline is paralellised to run jobs simultaneously where possible. To do this, set the parameter`--jobs N` in the command line, where N is the number of CPUs available on your machine. e.g.

```
ERVsearch --target_tasks full --jobs 8
```

This would run on 8 CPUs

If running on a high performance cluster, it is recommended to use a single node and set `--jobs` to the number of cores available on that node.

8.3 Verbosity

Ruffus verbosity is set using the `-v` parameter from 1 to 10. The recommended setting for ERvsearch is -v 5, however this needs to be specified to override the ruffus default of 1.

e.g.

```
ERVsearch --target_tasks full -v 5
```

CHAPTER

NINE

MAIN OUTPUT FILES

The main output files produced are as follows. Many additional output files are generated, these are described here.

9.1 Screen

`screen_results.dir/results.tsv` Table showing the ORFs identified with Exonerate and verified with UBLAST which meet the requirements specified in the `pipeline.ini` file. Columns are as follows:

- **name** Name assigned to the region consisting of the ID, chromosome, start and end positions
- **match** The most similar reference ORF to this ORF identified in the `ERVsearch/ERV_db` database using the ungapped Exonerate algorithm.
- **perc_identity** The percentage identity between this sequence and the most similar reference ORF, based on the UBLAST output.
- **alignment_length* The length of the alignment of this sequence and the most similar reference ORF, based on the UBLAST output.
- **evalue** The UBLAST e-value of the alignment of this sequence and the most similar reference ORF.
- **bit_score** The UBLAST bit score of the alignment of this sequence and the most similar reference ORF.
- **ID** ID assigned to this ORF
- **chrom** Chromosome (or scaffold, contig or sequence) on which this ORF was identified.
- **start** Start position of this ORF on the chromosome.
- **end** End position of this ORF on the chromosome.
- **strand** Positive sense (+) or negative sense (+)
- **group** Local group to which the most similar reference ORF belongs. If the reference ORF is not in a group, this is genus_gene.
- **genus** Genus to which the most similar reference ORF belongs.
- **length** ORF length in nucleotides.
- **gene** Retroviral gene - gag, pol or env.

`screen_results.dir/by_length.FMT` Histograms of ORF lengths (in nucleotides) based on the results.tsv table, for the gag, pol and env genes.

`screen_results.dir/by_genus.FMT` Bar charts showing the number of ORFs identified for each genus and gene based on the results.tsv table.

`screen_results.dir/by_group.FMT` Bar charts showing the number of ORFs identified in each small subgroup of reference sequences for each retroviral gene. ORFs assigned as genus_gene were related to a reference sequence which is not in a smaller subgroup, based on the results.tsv table.

`screen_results.dir/by_gene.FMT` Bar chart showing the number of ORFs identified for each gene, based on the results.tsv table.

9.2 Classify

`summary_trees.dir/*FMT` (can be png, jpg, svg or pdf depending on the plots_format parameter). Image files of the phylogenetic trees for each retroviral gene and genus. Different sized circles are used to show the relative size of collapsed monophyletic groups. Newly identified ERVs are highlighted.

`summary_trees.dir/*tre` Tree files in Newick format for each retroviral gene and genus combining reference and newly identified sequences. Monophyletic groups of newly identified ORF sequences are represented by a single sequence.

9.3 ERVRegions

`erv_regions_results.dir/results.tsv` Table summarising regions identified containing retrovirus-like ORFs from more than one gene. Columns:

- **name** - the final ID of the ERV region - the genes found plus an integer e.g. gag_pol_12
- **chrom** - chromosome
- **start** - start position of the ERV region
- **end** - end position of the ERV region
- **strand** - strand of the ERv region
- **genus** - genus of the ERV region, can be multiple genera delimted by "|" if different genes had different genera
- for each gene screened for (usually gag, pol and env)
  - **GENE_name** - the names of the ORFs for this gene in this region
  - **GENE_ID** - the original IDs of the ORFs for this gene in this region
  - **GENE_start** - the start position of this gene in this region (genome co-ordinates)
  - **GENE_relative_start** - the start position of this gene in this region (relative to the start of the region)
  - **GENE_end** - the end position of this gene in this region (genome co-ordinates)
  - **GENE_relative_end** - the end position of this gene in this region (relative to the start of the region)
  - **GENE_strand** - the strand for this gene in this region
  - **GENE_match** - the closest reference retrovirus to this gene in this region
  - **GENE_group** - the group of the closest reference retrovirus to this gene in this region
  - **GENE_genus** - the genus of the closest reference retrovirus to this gene in this region
- **orig_name** - the name of the region in the input table

CHAPTER

TEN

PARAMETERS

All parameters should be specified in the `pipeline.ini` configuration file

The template file can be found in `ERvsearch/templates/pipeline.ini`.

Make a copy of this file in your working directory (the directory where you want to run the program and store the output) and change the values of the parameters according to your system and your needs.

10.1 Required parameters

These parameters should always be set (there are no default options).

1. *genome* 1.1. *file*
2. *paths* 2.1 *path_to_usearch* 2.2 *path_to_exonerate*

10.1.1 genome

Input file parameters

file

`string`

Path to a single fasta file containing the genome or other sequence you would like to screen for ORFs

e.g. `/home/katy/genomes/hg38.fasta`

10.1.2 paths

Paths to software

path_to_usearch

`string`

Path to usearch executable

e.g. `/usr/bin/usearch11.0.667_u86linux32`

path_to_exonerate

'string'

Path to exonerate executable

e.g. `/usr/bin/exonerate`

10.2 Optional Parameters

The following parameters are optional (as they have default values)

1. *genomesplits* 1.1 *split* 1.2 *split_n* 1.3 *force*
2. *output* 2.1 *outfile_stem*
3. *database* 3.1 *use_custom_db* 3.2 *gag* 3.3 *pol* 3.4 *env*
4. *exonerate* 4.1 *overlap* 4.2 *min_score*
5. *usearch* 5.1 *min_id* 5.2 *min_hit_length* 5.3 *min_coverage*
6. *plots* 6.1 *dpi* 6.2 *format* 6.3 *gag_colour* 6.4 *pol_colour* 6.5 *env_colour*
7. *ORFs* 7.1 *min_orf_len* 7.2 *translation_table*
8. *trees* 8.1 *use_gene_colour* 8.2 *maincolour* 8.3 *highlightcolour* 8.4 *outgroupcolour* 8.5 *dpi* 8.6 *format*
9. *regions* 9.1 *maxdist*

10.2.1 genomesplits

Parameters for dividing the input genome into batches

If the genome has more than ~100 contigs or scaffolds, it is recommended to batch these rather than running Exonerate on each contig individually, to avoid creating an excessive number of output files

The default is to split the input into 50 batches, this is a manageable number for most systems.

split

`string` `True` or `False` Default: `True`

If split is True the contigs will be batched, if it is False Exonerate will run once for every gene-contig combination (usually 3x number of contigs). If there are less contigs than batches this will be ignored

split_n

`integer` Default: `50`

Number of batches to split the contigs into

force

`string` `True` or `False` Default: `False`

The pipeline will error if running using these genome split settings will run Exonerate more than 500 times. If you want to force the pipeline to run despite this, change force to True.

10.2.2 output

Output file parameters

outfile_stem

`string` Default: `ERVsearch`

Log files will have this as a prefix (e.g. ERVsearch_log.txt)

10.2.3 database

Parameters concerning the database of reference ERV sequences `string` `True` or `False`

use_custom_db

Default: `False`

When screening using Exonerate, query sequences are used to identify ERV like regions of the genome. It is possible to use the default ERV database provided with the pipeline (recommended) or to use a custom database.

False - use the default database

True - use a custom database

gag

`string` Default: `None`

Path to a custom fasta file of gag amino acid sequences. To skip this gene use None as this value (only if use_custom_db is True)

pol

`string` Default: `None`

Path to a custom fasta file of pol amino acid sequences. To skip this gene use None as this value (only if use_custom_db is True).

env

`string` Default: `None`

Path to a custom fasta file of env amino acid sequences. To skip this gene use None as this value (only if use_custom_db is True).

10.2.4 exonerate

Exonerate parameters

min_hit_length

`integer` Default: `100`

Minimum Exonerate hit length on the chromosome. Shorter hits are filtered out.

overlap

`integer` Default: `30`

Maximum distance between chromosome regions identified with Exonerate which are merged into single regions.

min_score

`integer` Default: `100`

Minimum score in the second exonerate pass (with ungapped algorithm).

10.2.5 usearch

USEARCH parameters

min_id

`float` Default: `0.5`

Percentage identity used by the UBLAST algorithm. Used to set the `-id` UBLAST parameter

min_hit_length

`integer` Default: `100`

Minimum hit length for UBLAST. Used to set the `-mincols` UBLAST parameter

min_coverage

`float` Default: `0.5`

Minimum proportion of the query sequence which should be covered using UBLAST. Used to set the `-query_cov` UBLAST parameter

10.2.6 plots

Plotting parameters

dpi

`integer` Default: `300`

Dots per inch for output plots (from the summarise functions).

format

`string` `png`, `svg`, `pdf` or `jpg`

File format for summary plot files, can be svg, png, pdf or jpg

gag_colour

`string` (hex colour code with #) Default: `#f38918`

Colour for *gag* gene in summary plots. Default is orange.

pol_colour

`string` (hex colour code with #) Default: `#4876f9`

Colour for *pol* gene in summary plots. Default is blue.

env_colour

`string` (hex colour code with #) Default: `#d61f54`

Colour for *env* gene in summary plots. Default is pink

other_colour

`string` (hex colour code with #) Default: `#33b54d`

Colour for anything which doesn't relate to a specific gene in summary plots. Default is green.

match_axes

`string` `True` or `False` Default: `False`

If True, when gag, pol and env are shown as subplots on the same figure they should all have the same axis limits. If they do some can be very small but they are more comparable.

10.2.7 ORFs

Parameters for ORF identification

min_orf_len

`integer` Default: `100`

Minimum length of ORFs to characterise.

translation_table

`integer` Default `1`

Translation table to use when identifying ORFs - listed here https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi. Usually table 1 is fine - default plus alternative initiation codons

10.2.8 trees

Phylogenetic tree parameters

use_gene_colour

`string` `True` or `False` Default: `True`

Use the colours specified in the plots section for each gene to highlight newly identified ERVs. If this is `True`, `highlightcolour` will be ignored and the gene colours will be used instead.If it is `False`, `highlightcolour` is used to highlight newly identified ERVs.

maincolour

`string` (hex colour code with #) Default: `#000000`

Main colour for text in tree images. Default is black.

highlightcolour

`string` (hex colour code with #) Default : `#382bfe`

Text colour for leaves highlighted in tree images - newly identified ERV-like regions. This is ignored if `use_gene_colour` above is `True`. Default is blue.

outgroupcolour

`string` (hex colour code with #) Default: `#0e954b`

Text colour for outgroup in tree images. Default is green.

dpi

`integer` Default: `300` Dots per inch for phylogenetic tree images.

format

`string` `png`, `svg`, `pdf` or `jpg` Default: `png`

File format for phylogenetic tree images, can be svg, png, pdf or jpg

10.2.9 regions

Parameters for defining regions with ORFs from more than one retroviral gene

maxdist

`integer` Default: 3000

Maximum distance (in nucleotides) between ORFs to be defined as part of the same ERV region.

maxoverlap

`float` Default: 0.5

Maximum proportion of an ORF which can overlap with an ORF from another gene before they are filtered out.

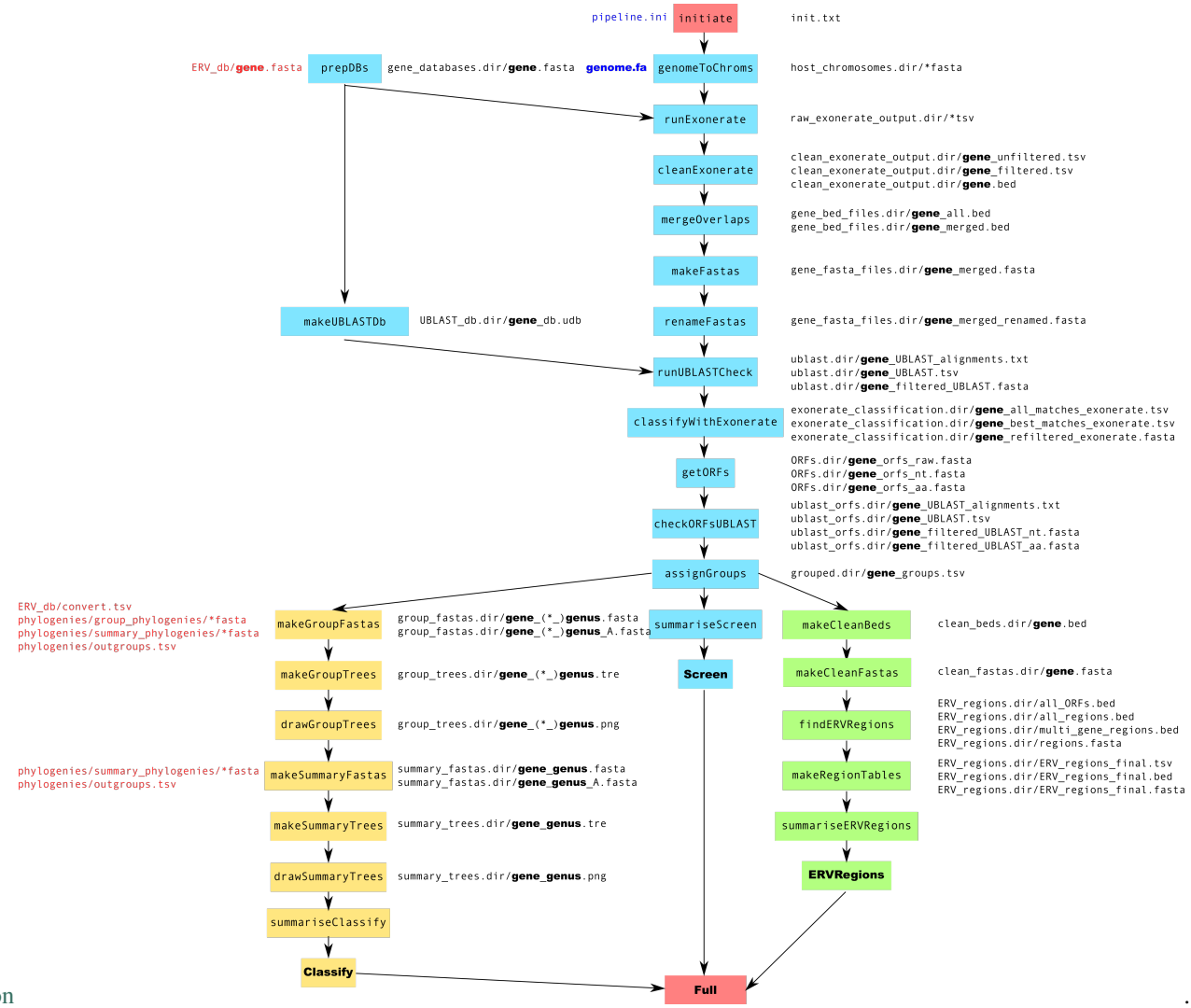
CHAPTER

ELEVEN

FUNCTIONS

1.*Pipeline Schematic* 2.*Screen* 3.*Classify* 4.*ERVRegions*

11.1 Pipeline Schematic



large version ../ima

NB: Where GENE is specified in a file path one file will be created for each gene - *gag*, *pol* and *env* (unless otherwise

specified by the user).

11.2 Screen

- Screens the genome for ERV like regions by comparing the genome to a set of known retroviral ORFs using Exonerate.
- Confirms the Exonerate regions using UBLAST
- Finds and confirms ORFs within these regions
- Finds the most similar known retroviral ORF in the database to each of the newly identified ORFs

1. *initiate*
2. *genomeToChroms*
3. *prepDBs*
4. *runExonerate*
5. *cleanExonerate*
6. *mergeOverlaps*
7. *makeFastas*
8. *renameFastas*
9. *makeUBLASTDb*
10. *runUBLASTCheck*
11. *classifyWithExonerate*
12. *getORFs*
13. *checkORFsUBLAST*
14. *assignGroups*
15. *summariseScreen*
16. *Screen*

11.2.1 initiate

Input Files `pipeline.ini`

Output Files `init.txt`

Parameters `[genome] file [paths] path_to_usearch [paths] path_to_exonerate`

Initialises the pipeline and checks that the required parameters in the pipeline.ini are set and valid and that the required software is in your $PATH.

Checks that:

- The input genome file exists.
- The correct path to ERVsearch is provided.
- samtools, bedtools, FastTree and mafft are in the $PATH
- The correct paths to usearch and exonerate are provided.

`init.txt` is a placeholder to show that this step has been completed.

11.2.2 genomeToChroms

Input Files `[genome] file` `keep_chroms.txt`

Output Files `host_chromosomes.dir/*fasta`

Parameters `[genome] file` `[genomesplits] split` `[genomesplits] split_n` `[genomesplits] force`

Splits the host genome provided by the user into FASTA files of a suitable size to run Exonerate efficiently.

If `genomesplits_split` in the pipeline.ini is False, the genome is split into one fasta file for each sequence - each chromosome, scaffold or contig.

If `genomesplits_split` in the pipeline.ini is True, the genome is split into the number of batches specified by the `genomesplits_splitn` parameter, unless the total number of sequences in the input file is less than this number.

The pipeline will fail if the number of sequences which would result from the genomesplits settings would result in >500 Exonerate runs, however it is possible to force the pipeline to run despite this by setting `genomesplits_force` to True.

If the file keep_chroms.txt exists in the working directory only chromosomes listed in this file will be kept.

An unzipped copy of zipped and gzipped fasta files will be created or a link to the file if it is already unzipped. this will be named `genome.fa` and be in the working directory.

This function generates a series of fasta files which are stored in the host_chromosomes.dir directory.

11.2.3 prepDBs

Input Files None

Output Files `gene_databases.dir/GENE.fasta`

Parameters `[database] use_custom_db` `[database] gag` `[database] pol` `[database] env`

Retrieves the gag, pol and env amino acid sequence database fasta files and puts a copy of each gene_databases.dir directory.

If custom databases are used they are retrieved and named as gag.fasta pol.fasta, env.fasta so the path doesn't need to be changed every time.

11.2.4 runExonerate

Input Files `gene_databases.dir/GENE.fasta` `host_chromosomes.dir/*fasta`

Output Files `raw_exonerate_output.dir/GENE_*.tsv`

Parameters `[paths] path_to_exonerate`

Runs the `protein2dna` algorithm in the Exonerate software package with the host chromosomes (or other regions) in `host_chromosomes.dir` as target sequences and the FASTA files from prepDBs as the query sequences.

The raw output of Exonerate is stored in the raw_exonerate_output directory, one file is created for each combination of query and target sequences.

This step is carried out with low stringency as results are later filtered using UBLAST and Exonerate.

11.2.5 cleanExonerate

Input Files `raw_exonerate_output.dir/GENE_*.tsv`

Output_Files `clean_exonerate_output.dir/GENE_*_unfiltered.tsv` `clean_exonerate_output.dir/GENE_*_filtered.tsv` `clean_exonerate_output.dir/GENE_*.bed`

Parameters `[exonerate] min_hit_length`

Filters and cleans up the Exonerate output.

- Converts the raw Exonerate output files into dataframes - GENE_unfiltered.tsv
- Filters out any regions containing introns (as defined by Exonerate)
- Filters out regions less than `exonerate_min_hit_length` on the host sequence (in nucleotides).
- Outputs the filtered regions to GENE_filtered.tsv
- Converts this to bed format and outputs this to GENE.bed

11.2.6 mergeOverlaps

Input Files `clean_exonerate_output.dir/GENE_*.bed`

Output_Files `gene_bed_files.dir/GENE_all.bed` `gene_bed_files.dir/GENE_merged.bed`

Parameters `[exonerate] overlap`

Merges the output bed files for individual sections of the input genome into a single bed file.

Overlapping regions or very close together regions of the genome detected by Exonerate with similarity to the same retroviral gene are then merged into single regions. This is performed using bedtools merge on the bed files output by cleanExonerate.

If there is a gap of less than `exonerate_overlap` between the regions they will be merged.

11.2.7 makeFastas

Input Files `gene_bed_files.dir/GENE_merged.bed` `genome.fa`

Output Files `gene_fasta_files.dir/GENE_merged.fasta`

Parameters None

Fasta files are generated containing the sequences of the merged regions of the genome identified using mergeOverlaps. These are extracted from the host chromosomes using bedtools getfasta.

11.2.8 renameFastas

Input Files `gene_fasta_files.dir/GENE_merged.fasta`

Output Files `gene_fasta_files.dir/GENE_merged_renamed.fasta`

Parameters None

Renames the sequences in the fasta files of ERV-like regions identified with Exonerate so each record has a numbered unique ID (gag1, gag2 etc). Also removes ":" from sequence names as this causes problems later.

11.2.9 makeUBLASTDb

Input Files `gene_databases.dir/GENE.fasta`

Output Files `UBLAST_db.dir/GENE_db.udb`

Parameters `[paths] path_to_usearch`

USEARCH requires an indexed database of query sequences to run. This function generates this database for the three gene amino acid fasta files used to screen the genome.

11.2.10 runUBLASTCheck

Input Files `UBLAST_db.dir/GENE_db.udb gene_fasta_files.dir/GENE_merged_renamed.fasta`

Output Files `ublast.dir/GENE_UBLAST_alignments.txt ublast.dir/GENE_UBLAST.tsv ublast.dir/GENE_filtered_UBLAST.fasta`

Parameters `[paths] path_to_usearch [usearch] min_id [usearch] min_hit_length [usearch] min_coverage`

ERV regions in the fasta files generated by makeFasta are compared to the ERV amino acid database files for a second time, this time using USEARCH (https://www.drive5.com/usearch/). Using both of these tools reduces the number of false positives.

This allows sequences with low similarity to known ERVs to be filtered out. Similarity thresholds can be set in the pipeline.ini file (`usearch_min_id`, - minimum identity between query and target - `usearch_min_hit_length` - minimum length of hit on target sequence - and `usearch_min_coverage` - minimum proportion of the query sequence the hit should cover).

The raw output of running UBLAST against the target sequences is saved in GENE_UBLAST_alignments.txt (equivalent to the BLAST default output) and GENE_UBLAST.tsv (equivalent to the BLAST -outfmt 6 tabular output) this is already filtered by passing the appropriate parameters to UBLAST. The regions which passed the filtering and are therefore in these output files are then output to a FASTA file GENE_filtered_UBLAST.fasta.

11.2.11 classifyWithExonerate

Input Files `ublast.dir/GENE_filtered_UBLAST.fasta ERVsearch/ERV_db/all_ERVs_nt.fasta`

Output Files `exonerate_classification.dir/GENE_all_matches_exonerate.tsv exonerate_classification.dir/GENE_best_matches_exonerate.tsv exonerate_classification.dir/GENE_refiltered_matches_exonerate.fasta`

Parameters `[paths] path_to_exonerate [exonerate] min_score`

Runs the Exonerate ungapped algorithm with each ERV region in the fasta files generated by makeFasta as queries and the all_ERVs_nt.fasta fasta file as a target, to detect which known retrovirus is most similar to each newly identified ERV region. Regions which don't meet a minimum score threshold (`exonerate_min_score`) are filtered out.

all_ERVs_nt.fasta contains nucleic acid sequences for many known endogenous and exogenous retroviruses with known classifications.

First all seqeunces are compared to the database and the raw output is saved as exonerate_classification.dirGENE_all_matches_exonerate.tsv. Results need a score greater than `exonerate_min_score`

against one of the genes of the same type (*gag*, *pol* or *env*) in the database. The highest scoring result which meets these critera for each sequence is then identified and output to exonerate_classification.dir/GENE_best_matches_exonerate.tsv. The sequences which meet these critera are also output to a FASTA file exonerate_classification.dir/GENE_refiltered_exonerate.fasta.

11.2.12 getORFs

Input Files `exonerate_classification.dir/GENE_refiltered_matches_exonerate.fasta`

Output Files `ORFs.dir/GENE_orfs_raw.fasta` `ORFs.dir/GENE_orfs_nt.fasta` `ORFs.dir/GENE_orfs_aa.fasta`

Parameters `[orfs] translation_table` `[orfs] min_orf_len`

Finds the longest open reading frame in each of the ERV regions in the filtered output table.

This analysis is performed using EMBOSS revseq and EMBOSS transeq.

The sequence is translated in all six frames using the user specified translation table. The longest ORF is then identified. ORFs shorter than orfs_min_orf_length are filtered out.

The positions of the ORFs are also convered so that they can be extracted directly from the input sequence file, rather than using the co-ordinates relative to the original Exonerate regions.

The raw transeq output, the nucleotide sequences of the ORFs and the amino acid sequences of the ORFs are written to the output FASTA files.

11.2.13 checkORFsUBLAST

Input Files `ORFs.dir/GENE_orfs_nt.fasta` `UBLAST_dbs.dir/GENE_db.udb`

Output Files `ublast_orfs.dir/GENE_UBLAST_alignments.txt` `ublast_orfs.dir/GENE_UBLAST.tsv` `ublast_orfs.dir/GENE_filtered_UBLAST.fasta`

Parameters `[paths] path_to_usearch` `[usearch] min_id` `[usearch] min_hit_length` `[usearch] min_coverage`

ERV ORFs in the fasta files generated by the ORFs function are compared to the original ERV amino acid files using UBLAST. This allows any remaining sequences with poor similarity to known ERVs to be filtered out.

This allows ORFs with low similarity to known ERVs to be filtered out. Similarity thresholds can be set in the pipeline.ini file (`usearch_min_id`, - minimum identity between query and target - `usearch_min_hit_length` - minimum length of hit on target sequence - and `usearch_min_coverage` - minimum proportion of the query sequence the hit should cover).

The raw output of running UBLAST against the target sequences is saved in GENE_UBLAST_alignments.txt (equivalent to the BLAST default output) and GENE_UBLAST.tsv (equivalent to the BLAST -outfmt 6 tabular output) this is already filtered by passing the appropriate parameters to UBLAST. The regions which passed the filtering and are therefore in these output files are then output to a FASTA file GENE_filtered_UBLAST.fasta.

11.2.14 assignGroups

Input Files `ublast_orfs.dir/GENE_UBLAST.tsv` `ERVsearch/ERV_db/convert.tsv`

Output Files `grouped.dir/GENE_groups.tsv`

Parameters None

Many of the retroviruses in the input database all_ERVs_nt.fasta have been classified into groups based on sequence similarity, prior knowledge and phylogenetic clustering. Some sequences don't fall into any well defined group, in these cases they are just assigned to a genus, usually based on prior knowledge. The information about these groups is stored in the provided file ERVsearch/ERV_db/convert.tsv.

Each sequence in the filtered fasta file of newly identified ORFs is assigned to one of these groups based on the sequence identified as the most similar in the classifyWithExonerate step.

The output table is also tidied up to include the UBLAST output, chromosome, ORF start and end positions, genus and group.

11.2.15 summariseScreen

Input Files `gene_bed_files.dir/GENE_merged.bed` `ublast.dir/GENE_UBLAST.tsv` `ORFs.dir/GENE_orfs_aa.fasta` `ublast_orfs.dir/GENE_UBLAST.tsv`

Output Files FMT can be png, svg, pdf, jpg depending on the `plot_format` parameter

`summary_tables.dir/exonerate_initial_summary.txt` `summary_tables.dir/ublast_hits_initial_summary.txt` `summary_tables.dir/orfs_initial_summary.txt` `summary_tables.dir/ublast_orfs_initial_summary.txt` `summary_plots.dir/exonerate_initial_lengths.FMT` `summary_plots.dir/exonerate_initial_scores.FMT` `summary_plots.dir/exonerate_initial_strands.FMT` `summary_plots.dir/exonerate_initial_by_sequence.FMT` `summary_plots.dir/exonerate_initial_counts_per_gene.FMT` `summary_plots.dir/ublast_hits_alignment_length.FMT` `summary_plots.dir/ublast_hits_perc_similarity.FMT` `summary_plots.dir/ublast_hits_by_match.FMT` `summary_plots.dir/ublast_hits_per_gene.FMT` `summary_plots.dir/orfs_lengths.FMT` `summary_plots.dir/orfs_strands.FMT` `summary_plots.dir/orfs_by_gene.FMT` `summary_plots.dir/ublast_orfs_alignment_length.FMT` `summary_plots.dir/ublast_orfs_perc_similarity.FMT` `summary_plots.dir/ublast_orfs_bit_score.FMT` `summary_plots.dir/ublast_orfs_by_match.FMT` `summary_plots.dir/ublast_orfs_per_gene.FMT` `screen_results.dir/results.tsv` `screen_results.dir/by_length.FMT` `screen_results.dir/by_genus.FMT` `screen_results.dir/by_group.FMT` `screen_results.dir/by_gene.FMT`

Parameters `[plots] dpi` `[plots] format` `[plots] gag_colour` `[plots] pol_colour` `[plots] env_colour` `[plots] other_colour` `[plots] match_axes`

Generates a series of summary plots and tables showing the results of running the screening functions.

The major outputs of this function are stored in the screen_results.dir directory. Further details of these files are provided in the Main Output Files section.

The other files show the output of the intermediate steps.

Exonerate Initial

- `summary_tables.dir/exonerate_initial_summary.txt` Summary of the output of the initial Exonerate screening step. Note that these are unfiltered and many will not be true ERVs.
- `summary_tables.dir/ublast_hits_initial_summary.txt` Summary of the results of running UBLAST on the initial Exonerate output.

- summary_tables.dir/orfs_initial_summary.txt Summary of the results of the initial ORF identification.
- summary_tables.dir/ublast_orfs_initial_summary.txt Summary of the results of running UBLAST on these ORFs.
- summary_plots.dir/exonerate_initial_lengths.FMT Histogram showing the lengths of the initial Exonerate regions for each gene.
- summary_plots.dir/exonerate_initial_scores.FMT Histogram showing the Exonerate score of the initial Exonerate regions for each gene.
- summary_plots.dir/exonerate_initial_strands.FMT Bar chart showing the number of regions identified on each strand in the initial Exonerate screen.
- summary_plots.dir/exonerate_initial_by_sequence.FMT Histogram showing the number of ERV-like regions identified on each sequence in the reference genome being screened.
- summary_plots.dir/exonerate_initial_counts_per_gene.FMT Bar chart showing the number of ERV regions identified per gene in the initial Exonerate screen.

UBLAST

- summary_plots.dir/ublast_hits_alignment_length.FMT Histogram showing the lengths of the alignments of the UBLAST filtered Exonerate regions and the most similar reference ORF, based on the UBLAST output.
- summary_plots.dir/ublast_hits_perc_similarity.FMT Histogram showing the percentage identity between the UBLAST filtered Exonerate regions and the most similar reference ORF, based on the UBLAST output.
- summary_plots.dir/ublast_hits_perc_similarity.FMT Histogram showing the UBLAST bit score between the UBLAST filtered Exonerate regions and the most similar reference ORF, based on the UBLAST output.
- summary_plots.dir/ublast_hits_by_match.FMT Bar chart showing the number of UBLAST filtered Exonerate regions most similar to each reference ORF in the ERVsearch/ERV_db database.
- summary_plots.dir/ublast_hits_per_gene.FMT Bar chart showing the number of UBLAST filtered Exonerate regions identified per gene.

ORFs

- summary_plots.dir/orfs_lengths.FMT Histogram of the lengths of ORFs identified in the ERV regions.
- summary_plots.dir/orfs_strands.FMT Bar chart of the strand (positive (+) or negative (-) sense) of the ORFs identified in the ERV regions.
- summary_plots.dir/orfs_by_gene.FMT Bar chart of the number of ORFs identified for each gene.

UBLAST ORFs

- summary_plots.dir/ublast_orfs_alignment_length.FMT Histogram showing the lengths of the alignments of the ERV-like ORFs and the most similar reference ORF, based on the UBLAST output.
- summary_plots.dir/ublast_orfs_perc_similarity.FMT Histogram showing the percentage identity between the ERV-like ORFs and the most similar reference ORF, based on the UBLAST output.
- summary_plots.dir/ublast_orfs_bit_score.FMT Histogram showing the UBLAST bit score between the ERV-like ORFs and the most similar reference ORF, based on the UBLAST output.
- summary_plots.dir/ublast_orfs_by_match.FMT Bar chart showing the number of ERV-like ORFs most similar to each reference ORF in the ERVsearch/ERV_db database.

- summary_plots.dir/ublast_orfs_per_gene.FMT Bar chart showing the number of ERV-like ORFs identified per gene.

11.2.16 Screen

Input Files None

Output Files None

Parameters None

Helper function to run all screening functions (all functions prior to this point).

11.3 Classify

- Classifies the newly identified ORFs into groups based on the most similar known ORF
- Aligns the newly identified ORFs with reference sequences within these groups and builds a phylogenetic tree for each group.
- Finds clusters of newly identified ORFs within these trees
- Incorporates representative sequences from these clusters into a summary tree for each retroviral gene and genus (based on classification into *gamma*, *beta*, *spuma*, *alpha*, *lenti*, *epsilon* and *delta* retroviruses as defined by the ICTV.

1. *makeGroupFastas*
2. *makeGroupTrees*
3. *drawGroupTrees*
4. *makeSummaryFastas*
5. *makeSummaryTrees*
6. *drawSummaryTrees*
7. *summariseClassify*
8. *Classify*

11.3.1 makeGroupFastas

Input Files grouped.dir/GENE_groups.tsv ERVsearch/phylogenies/group_phylogenies/*fasta ERVsearch/phylogenies/summary_phylogenies/*fasta ERVsearch/phylogenies/outgroups.tsv

Output Files group_fastas.dir/GENE_(.*)_GENUS.fasta group_fastas.dir/GENE_(.*)_GENUS_A.fasta

Parameters None

Two sets of reference fasta files are available (files are stored in ERVsearch/phylogenies/group_phylogenies and ERVsearch/phylogenies/summary_phylogenies)

- group_phylogenies - groups of closely related ERVs for fine classification of sequences
- summary_phylogenies - groups of most distant ERVs for broad classification of sequences

Sequences have been assigned to groups based on the most similar sequence in the provided ERV database, based on the score using the Exonerate ungapped algorithm. Where the most similar sequence is not part of a a well defined group, it has been assigned to a genus.

Fasta files are generated containing all members of the group from the group_phylogenies file (plus an outgroup) where possible and using representative sequences from the same genus, using the summary_phylogenies file, where only a genus has been assigned, plus all the newly identified ERVs in the group. These files are saved as GENE_(group_name_)GENUS.fasta.

A "~" is added to all new sequence names so they can be searched for easily.

The files are aligned using the MAFFT fftns algorithm https://mafft.cbrc.jp/alignment/software/manual/manual.html to generate the GENE_(group_name_)GENUS_A.fasta aligned output files.

11.3.2 makeGroupTrees

Input Files `group_fastas.dir/GENE_(.*_)GENUS_A.fasta`

Output Files `group_trees.dir/GENE_(.*_)GENUS.tre`

Parameters None

Builds a phylogenetic tree, using the FastTree2 algorithm (http://www.microbesonline.org/fasttree) with the default settings plus the GTR model, for the aligned group FASTA files generated by the makeGroupFastas function.

11.3.3 drawGroupTrees

Input Files `group_trees.dir/GENE_(.*_)GENUS.tre`

Output Files `group_trees.dir/GENE_(.*_)GENUS.FMT` (png, svg, pdf or jpg)

Parameters `[plots] gag_colour` `[plots] pol_colour` `[plots] env_colour` `[trees] use_gene_colour` `[trees] maincolour` `[trees] highlightcolour` `[trees] outgroupcolour` `[trees] dpi` `[trees] format`

Generates an image file for each file generated in the makeGroupTrees step, using ete3 (http://etetoolkit.org). Newly identified sequences are labelled as "~" and shown in a different colour.

By default, newly identified sequences are shown in the colours specified in `plots_gag_colour`, `plots_pol_colour` and `plots_env_colour` - to do this then `trees_use_gene_colour` should be set to True in the `pipeline.ini`. Alternatively, a fixed colour can be used by setting `trees_use_gene_colour` to False and settings `trees_highlightcolour`. The text colour of the reference sequences (default black) can be set using `trees_maincolour` and the outgroup using `trees_outgroupcolour`.

The output file DPI can be specified using `trees_dpi` and the format (which can be png, svg, pdf or jpg) using `trees_format`.

11.3.4 makeSummaryFastas

Input Files `group_fastas.dir/GENE_(.*_)GENUS.fasta` `group_trees.dir/GENE_(*_)GENUS.tre` `ERVsearch/phylogenies/summary_phylogenies/GENE_GENUS.fasta` `ERVsearch/phylogenies/group_phylogenies/(.*)_GENUS_GENE.fasta`

Output Files `summary_fastas.dir/GENE_GENUS.fasta` `summary_fastas.dir/GENE_GENUS.tre`

Parameters None

Based on the group phylogenetic trees generated in makeGroupTrees, monophyletic groups of newly idenified ERVs are identified. For each of these groups, a single sequence (the longest) is selected as representative. The representative sequences are combined with the FASTA files in `ERVsearch/phylogenies/summary_phylogenies`, which contain representative sequences for each retroviral gene and genus. These are extended to include further reference sequences from the same small group as the newly identified sequences.

For example, if one MLV-like pol and one HERVF-like pol was identified in the gamma genus, the gamma_pol.fasta summary fasta would contain: * The new MLV-like pol sequence * The new HERVF-like pol sequence * The reference sequences from `ERVsearch/phylogenies/group_phylogenies/MLV-like_gamma_pol.fasta` - highly related sequences from the MLV-like group * The reference sequences from `ERVsearch/phylogenies/group_phylogenies/HERVF-like_gamma_pol.fasta` - highly related sequences from the HERVF-like group. * The reference sequences from `ERVsearch/phylogenies/summary_phylogenies/gamma_pol.fasta` - a less detailed but more diverse set of gammaretroviral pol ORFs. * A epsilonretrovirus outgroup

This ensures sufficient detail in the groups of interest while avoiding excessive detail in groups where nothing new has been identified.

These FASTA files are saved as GENE_GENUS.fasta

The files are aligned using the MAFFT fftns algorithm https://mafft.cbrc.jp/alignment/software/manual/manual.html to generate the GENE_GENUS_A.fasta aligned output files.

11.3.5 makeSummaryTrees

Input Files `summary_fastas.dir/GENE_GENUS_A.fasta`

Output Files `summary_trees.dir/GENE_GENUS.tre`

Parameters None

Builds a phylogenetic tree, using the FastTree2 algorithm (http://www.microbesonline.org/fasttree) with the default settings plus the GTR model, for the aligned group FASTA files generated by the makeSummaryFastas function.

11.3.6 drawSummaryTrees

Input Files `summary_trees.dir/GENE_GENUS.tre`

Output Files `summary_trees.dir/GENE_GENUS.FMT`(FMT = png, svg, pdf or jpg)

Parameters `[plots] gag_colour` `[plots] pol_colour` `[plots] env_colour` `[trees] use_gene_colour` `[trees] maincolour` `[trees] highlightcolour` `[trees] outgroupcolour` `[trees] dpi` `[trees] format`

Generates an image file for each file generated in the makeSummaryTrees step, using ete3 (http://etetoolkit.org). Newly identified sequences are labelled as "~" and shown in a different colour. Monophyletic groups of newly identified ERVs have been collapsed (by choosing a single representative sequence) and the number of sequences in the group is added to the label and represented by the size of the node tip.

By default, newly identified sequences are shown in the colours specified in `plots_gag_colour`, `plots_pol_colour` and `plots_env_colour` - to do this then `trees_use_gene_colour` should be set to True in the `pipeline.ini`. Alternatively, a fixed colour can be used by setting `trees_use_gene_colour` to False and settings `trees_highlightcolour`. The text colour of the reference sequences (default black) can be set using `trees_maincolour` and the outgroup using `trees_outgroupcolour`.

The output file DPI can be specified using `trees_dpi` and the format (which can be png, svg, pdf or jpg) using `trees_format`.

11.3.7 summariseClassify

Input Files `summary_fastas.dir/GENE_GENUS.fasta` `summary_trees.dir/GENE_GENUS.tre`

Output Files `classify_results.dir/results.tsv` `classify_results.dir/by_gene_genus.FMT` (png, svg, pdf or jpg)

Parameters None

Combines the results of the classify steps to generate additional summary files. The `results.tsv` output file lists the number of genes which have been collapsed into each group in the trees in the `summary_trees.dir` directory. The by_gene_genus.FMT plot is a bar chart of the same information, organised by gene and genus.

11.3.8 Classify

Input Files None

Output Files None

Parameters None

Helper function to run all screening functions and classification functions (all functions prior to this point).

11.4 ERVRegions

Identifies regions of the genome containing ORFs resembling more than one different retroviral gene within a certain distance

1. *makeCleanBeds*
2. *makeCleanFastas*
3. *findERVRegions*
4. *makeRegionTables*
5. *plotERVRegions*
6. *summariseERVRegions*
7. *ERVRegions*
8. *Full*

11.4.1 makeCleanBeds

Input Files `grouped.dir/GENE_groups.tsv`

Output Files `clean_beds.dir/GENE.bed`

Parameters None

Generates a bed file for each gene which contains the co-ordinates of the ORFs which have passed all filtering criteria in the Screen section.

11.4.2 makeCleanFastas

Input Files `clean_beds.dir/GENE.bed` `genome.fa`

Output Files `clean_fastas.dir/GENE.fasta`

Parameters None

Fasta files are generated containing the sequences of the regions listed by makeCleanBeds. These are extracted from the host chromosomes using bedtools getfasta (https://bedtools.readthedocs.io/en/latest/content/tools/getfasta.html).

11.4.3 findERVRegions

Input Files `clean_fastas.dir/*.fasta`

Output Files `ERV_regions.dir/all_ORFs.bed` `ERV_regions.dir/all_regions.bed` `ERV_regions.dir/multi_gene_regions.bed` `ERV_regions.dir/regions.fasta`

Parameters `[regions] maxdist`

Combines the files containng the ORF regions for the different retroviral genes and merges any regions which are within `regions_maxdist` of each other to find larger regions containing multiple genes. The `all_ORFs.bed` output file is the concatenated and sorted bed files, `all_regions.bed`contains the merged regions with any ORFs within regions_maxdist of each other (end to end) combined, plus all regions with a single ORF, generated from all_regions.bed using bedtools merge (https://bedtools.readthedocs.io/en/latest/content/tools/merge.html). The name, strand and score columns are concatenated for merged regions, delimited with a ",". `multi_gene_regions.bed` contains only the regions which were found to contain multiple ORFs, `regions.fasta` is the sequence of these regions in FASTA format. At this point this includes regions with multiple ORFs from the same gene (e.g. two *pol* ORFs).

11.4.4 makeRegionTables

Input Files `ERV_regions.dir/multi_gene_regions.bed` `grouped.dir/*_groups.tsv` `genome.fa`

Output Files `ERV_regions.dir/ERV_regions_final.tsv` `ERV_regions.dir/ERV_regions_final.bed` `ERV_regions.dir/ERV_regions_final.fasta`

Parameters `[regions] maxoverlap`

Takes a merged bed file consisting of regions of the genome identified as having more than one ERV-like ORF, finds the regions within this file which contain more than one different gene (e.g. gag and pol instead of two gag ORFs) and outputs a formatted table of information about these regions.

The output table (`ERV_regions_final.tsv`) will usually have 37 columns:

- `name` - the final ID of the ERV region - the genes found plus an integer e.g. gag_pol_12
- `chrom` - chromosome
- `start` - start position of the ERV region
- `end` - end position of the ERV region
- `strand` - strand of the ERv region
- `genus` - genus of the ERV region, can be multiple genera delimted by "|" if different genes had different genera
- for each gene screened for (usually gag, pol and env)
  - `GENE_name` - the names of the ORFs for this gene in this region

  - GENE_ID - the original IDs of the ORFs for this gene in this region
  - GENE_start - the start position of this gene in this region (genome co-ordinates)
  - GENE_relative_start - the start position of this gene in this region (relative to the start of the region)
  - GENE_end - the end position of this gene in this region (genome co-ordinates)
  - GENE_relative_end - the end position of this gene in this region (relative to the start of the region)
  - GENE_strand - the strand for this gene in this region
  - GENE_match - the closest reference retrovirus to this gene in this region
  - GENE_group - the group of the closest reference retrovirus to this gene in this region
  - GENE_genus - the genus of the closest reference retrovirus to this gene in this region
- orig_name - the name of the region in the input table

If not all genes are screened for the table will not have the columns for this gene.

A bed file (ERV_regions_final.bed) is generated with the co-ordinates of the identified regions and a FASTA file (ERV_regions_final.fasta) containing their sequences.

11.4.5 plotERVRegions

Input_Files ERV_regions.dir/ERV_regions_final.tsv

Output_Files ERV_region_plots.dir/*FMT

Parameters [plots] format [plots] dpi [plots] gag_colour [plots] pol_colour [plots] env_colour

For each region containing ORFs resembling more than one retroviral gene, a plot is generated showing how these ORFs are distributed on the genome relative to each other.

Each gene is shown on a different line on the y axis, the x axis is chromosome co-ordinates.

11.4.6 summariseERVRegions

Input Files ERV_regions.dir/ERV_regions_final.tsv

Output Files erv_regions_results.dir/results.tsv [erv_regions_results.dir/erv_regions.FMT](introduction.html#id3

Parameters [plots] other_colour

Combines the results of the ERVregions steps to generate additional summary files.

The results.tsv output file is a copy of the output of the makeRegionTables functions. This will usually have 37 columns:

- name - the final ID of the ERV region - the genes found plus an integer e.g. gag_pol_12
- chrom - chromosome
- start - start position of the ERV region
- end - end position of the ERV region
- strand - strand of the ERv region
- genus - genus of the ERV region, can be multiple genera delimted by "|" if different genes had different genera

- for each gene screened for (usually gag, pol and env)
  - `GENE_name` - the names of the ORFs for this gene in this region
  - `GENE_ID` - the original IDs of the ORFs for this gene in this region
  - `GENE_start` - the start position of this gene in this region (genome co-ordinates)
  - `GENE_relative_start` - the start position of this gene in this region (relative to the start of the region)
  - `GENE_end` - the end position of this gene in this region (genome co-ordinates)
  - `GENE_relative_end` - the end position of this gene in this region (relative to the start of the region)
  - `GENE_strand` - the strand for this gene in this region
  - `GENE_match` - the closest reference retrovirus to this gene in this region
  - `GENE_group` - the group of the closest reference retrovirus to this gene in this region
  - `GENE_genus` - the genus of the closest reference retrovirus to this gene in this region
- `orig_name` - the name of the region in the input table

A bar chart - `erv_regions_results.dir/erv_regions.FMT` is also generated showing the number of ERV regions found with each combination of genes.

11.4.7 ERVRegions

Input Files None

Output Files None

Parameters None

Helper function to run all screening functions and ERVRegions functions.

11.4.8 Full

Input Files None

Output Files None

Parameters None

Helper function to run all functions.

CHAPTER

TWELVE

MINOR OUTPUT FILES

1.*Screen* 2.*Classify* 3.*ERVRegions*

12.1 Screen

1. *initiate*
2. *genomeToChroms*
3. *prepDBs*
4. *runExonerate*
5. *cleanExonerate*
6. *mergeOverlaps*
7. *makeFastas*
8. *renameFastas*
9. *makeUBLASTDb*
10. *runUBLASTCheck*
11. *classifyWithExonerate*
12. *getORFs*
13. *checkORFsUBLAST*
14. *assignGroups*
15. *summariseScreen*
16. *Screen*

12.1.1 initiate

`init.txt` Placeholder file to show initial checks have been run.

12.1.2 genomeToChroms

`host_chromosomes.dir/*fasta` FASTA format files containing the input genome (or other sequence), divided into regions based on the genomesplits parameters.

12.1.3 prepDBs

`gene_databases.dir/GENE.fasta` Copies of the fasta files of reference retroviral amino acid sequences.

12.1.4 runExonerate

`raw_exonerate_output.dir/GENE_*.tsv` Raw "vulgar" output of Exonerate as described here

12.1.5 cleanExonerate

`clean_exonerate_output.dir/GENE_*_unfiltered.tsv` Raw exonerate output converted into a table. Columns:

- **query_id** Reference amino acid sequence ID for retroviral gene
- **query_start** Start position of match within reference sequence
- **query_end** End position of match within reference sequence
- **query_strand** Strand of match relative to reference sequence
- **target_id** Nucleotide sequence from input which matched the retroviral gene.
- **target_start** Start position of match within input sequence.
- **target_end** End position of match within output sequence.
- **target_strand** Strand of match relative to input sequence
- **score** Exonerate score of match
- **details** Additional columns (columns 11+) from Exonerate vulgar output, delimited by "|".
- **length** Length of match on input sequence

`clean_exonerate_output.dir/GENE_*_filtered.tsv` Output of Exonerate filtered to remove regions containing introns and regions which are shorter than exonerate_min_hit_length. Columns are the same as in the unfiltered table.

`clean_exonerate_output.dir/GENE_*.bed` ERV-like regions from the table above in bed

12.1.6 mergeOverlaps

`gene_bed_files.dir/GENE_all.bed`, ERV-like regions from `clean_exonerate_output.dir/GENE_*.bed` combined into a single bed file.

`gene_bed_files.dir/GENE_merged.bed` ERV-like regions from the previous file merged using Bedtools merge

12.1.7 makeFastas

`gene_fasta_files.dir/GENE_merged.fasta` Fasta files of the merged regions in `gene_bed_files.dir/GENE_merged.bed`

12.1.8 renameFastas

`gene_fasta_files.dir/GENE_merged_renamed.fasta` Fasta files from the `makeFastas` step with the ERV-like regions renamed with unique IDs.

12.1.9 makeUBLASTDb

`UBLAST_db.dir/GENE_db.udb` UBLAST databases for the reference retroviral amino acid sequences.

12.1.10 runUBLASTCheck

`ublast.dir/GENE_UBLAST_alignments.txt` Raw UBLAST output files for the Exonerate regions vs the retroviral amino acid databases. Equivalent to the BLAST pairwise output.

`ublast.dir/GENE_UBLAST.tsv` UBLAST tabular output for Exonerate regions vs the retrovrial amino acid databases. Equivalent to the BLAST tabular output.

`ublast.dir/GENE_filtered_UBLAST.fasta` Fasta file of the regions which passed the UBLAST filter.

12.1.11 classifyWithExonerate

`exonerate_classification.dir/GENE_all_matches_exonerate.tsv` Raw output of ungapped Exonerate algorithm for the UBLAST verified regions against the ERV amino acid database. Columns:

- ID of newly identified ERV-like region
- ID of reference ERV amino acid
- Exonerate score

`exonerate_classification.dir/GENE_best_matches_exonerate.tsv` The previous table filtered to list only the highest scoring hit for each ERV-like region.

`exonerate_classification.dir/GENE_refiltered_matches_exonerate.fasta` Highest scoring hits from the previous table in FASTA format.

12.1.12 getORFs

`ORFs.dir/GENE_orfs_raw.fasta` Raw output of running EMBOSS transeq -frame 6 on the regions output from `classifyWithExonerate`.

`ORFs.dir/GENE_orfs_nt.fasta` ORFs longer than ORFs_min_orf_length as nucleotide sequences, with IDs redefined to include the chromosome, start and end position and strand of the ORF.

`ORFs.dir/GENE_orfs_aa.fasta` ORFs longer than ORFs_min_orf_length as amino acid sequences, with IDs redefined to include the chromosome, start and end position and strand of the ORF.

12.1.13 checkORFsUBLAST

`ublast_orfs.dir/GENE_UBLAST_alignments.txt` Raw UBLAST output files for the newly identified ORFs vs the retroviral amino acid databases. Equivalent to the BLAST pairwise output.

`ublast_orfs.dir/GENE_UBLAST.tsv` UBLAST tabular output for newly identified ORFs vs the retrovrial amino acid databases. Equivalent to the BLAST tabular output.

`ublast_orfs.dir/GENE_filtered_UBLAST.fasta` Fasta file of the newly identified ORFs which passed the UBLAST filter.

12.1.14 assignGroups

`grouped.dir/GENE_groups.tsv` Summarised output for the previous steps. This is identical to `screen_results.dir/results.tsv`.

12.1.15 summariseScreen

FMT can be png, svg, pdf, jpg depending on the `plot_format` parameter

The major outputs of this function are stored in the screen_results.dir directory. Further details of these files are provided in the Main Output Files section.

The other files show the output of the intermediate steps.

Exonerate Initial

- `summary_tables.dir/exonerate_initial_summary.txt` Summary of the output of the initial Exonerate screening step. Note that these are unfiltered and many will not be true ERVs.
- `summary_tables.dir/ublast_hits_initial_summary.txt` Summary of the results of running UBLAST on the initial Exonerate output.
- `summary_tables.dir/orfs_initial_summary.txt` Summary of the results of the initial ORF identification.
- `summary_tables.dir/ublast_orfs_initial_summary.txt` Summary of the results of running UBLAST on these ORFs.
- `summary_plots.dir/exonerate_initial_lengths.FMT` Histogram showing the lengths of the initial Exonerate regions for each gene.
- `summary_plots.dir/exonerate_initial_scores.FMT` Histogram showing the Exonerate score of the initial Exonerate regions for each gene.
- `summary_plots.dir/exonerate_initial_strands.FMT` Bar chart showing the number of regions identified on each strand in the initial Exonerate screen.
- `summary_plots.dir/exonerate_initial_by_sequence.FMT` Histogram showing the number of ERV-like regions identified on each sequence in the reference genome being screened.
- `summary_plots.dir/exonerate_initial_counts_per_gene.FMT` Bar chart showing the number of ERV regions identified per gene in the initial Exonerate screen.

UBLAST

- `summary_plots.dir/ublast_hits_alignment_length.FMT` Histogram showing the lengths of the alignments of the UBLAST filtered Exonerate regions and the most similar reference ORF, based on the UBLAST output.

- `summary_plots.dir/ublast_hits_perc_similarity.FMT` Histogram showing the percentage identity between the UBLAST filtered Exonerate regions and the most similar reference ORF, based on the UBLAST output.
- `summary_plots.dir/ublast_hits_perc_similarity.FMT` Histogram showing the UBLAST bit score between the UBLAST filtered Exonerate regions and the most similar reference ORF, based on the UBLAST output.
- `summary_plots.dir/ublast_hits_by_match.FMT` Bar chart showing the number of UBLAST filtered Exonerate regions most similar to each reference ORF in the ERVsearch/ERV_db database.
- `summary_plots.dir/ublast_hits_per_gene.FMT` Bar chart showing the number of UBLAST filtered Exonerate regions identified per gene.

ORFs

- `summary_plots.dir/orfs_lengths.FMT` Histogram of the lengths of ORFs identified in the ERV regions.
- `summary_plots.dir/orfs_strands.FMT` Bar chart of the strand (positive (+) or negative (-) sense) of the ORFs identified in the ERV regions.
- `summary_plots.dir/orfs_by_gene.FMT` Bar chart of the number of ORFs identified for each gene.

UBLAST ORFs

- `summary_plots.dir/ublast_orfs_alignment_length.FMT` Histogram showing the lengths of the alignments of the ERV-like ORFs and the most similar reference ORF, based on the UBLAST output.
- `summary_plots.dir/ublast_orfs_perc_similarity.FMT` Histogram showing the percentage identity between the ERV-like ORFs and the most similar reference ORF, based on the UBLAST output.
- `summary_plots.dir/ublast_orfs_bit_score.FMT` Histogram showing the UBLAST bit score between the ERV-like ORFs and the most similar reference ORF, based on the UBLAST output.
- `summary_plots.dir/ublast_orfs_by_match.FMT` Bar chart showing the number of ERV-like ORFs most similar to each reference ORF in the ERVsearch/ERV_db database.
- `summary_plots.dir/ublast_orfs_per_gene.FMT` Bar chart showing the number of ERV-like ORFs identified per gene.

12.2 Classify

1. *makeGroupFastas*
2. *makeGroupTrees*
3. *drawGroupTrees*
4. *makeSummaryFastas*
5. *makeSummaryTrees*
6. *drawSummaryTrees*
7. *summariseClassify*
8. *Classify*

12.2.1 makeGroupFastas

`group_fastas.dir/GENE_(.*)_GENUS.fasta` Fasta files for each small subgroup of ERV-like ORFs and reference sequences.

`group_fastas.dir/GENE_(.*)_GENUS_A.fasta` Aligned version of the above Fasta file generated using MAFFT

12.2.2 makeGroupTrees

`group_trees.dir/GENE_(.*)_GENUS.tre Phylogenetic trees in Newick format for each small subgroup of ERV-like ORFs and reference sequences generated using FastTree

12.2.3 drawGroupTrees

`group_trees.dir/GENE_(.*_)GENUS.FMT` (png, svg, pdf or jpg) Image files of the phylogenetic trees for each small subgroup of ERV-like ORFs and reference sequences, with newly identified sequences highlighted.

12.2.4 makeSummaryFastas

`summary_fastas.dir/GENE_GENUS.fasta` Fasta files for each retroviral gene and genus combining reference and newly identified sequences. Monophyletic groups of newly identified ORfs sequences are represented by a single sequence.

`group_lists.dir/*tsv` Lists of sequences in each monophyletic group of newly identified ORFs which has been collated to be represented by a single sequnce.

12.2.5 makeSummaryTrees

`summary_trees.dir/GENE_GENUS.tre` Tree files in Newick format for each retroviral gene and genus combining reference and newly identified sequences. Monophyletic groups of newly identified ORfs sequences are represented by a single sequence.

12.2.6 drawSummaryTrees

`summary_trees.dir/GENE_GENUS.FMT` (FMT = png, svg, pdf or jpg) Images files of the phylogenetic trees for each retroviral gene and genus. Different sized circles are used to show the relative size of collapsed monophyletic groups. Newly identified ERVs are highlighted.

12.2.7 summariseClassify

`classify_results.dir/results.tsv` Table listing the number of genes which have been collapsed into each monophyletic group in the trees in the `summary_trees.dir` directory. Columns:

- **gene** Retroviral gene for this group
- **genus** Retroviral genus for this group
- **group** Group ID
- **count** Number of sequences in this group

`classify_results.dir/by_gene_genus.png` Bar chart showing the number of genes which have been collapsed into each monophyletic group, organised by gene and genus.

12.3 ERVRegions

1. *makeCleanBeds*
2. *makeCleanFastas*
3. *findERVRegions*
4. *makeRegionTables*
5. *summariseERVRegions*
6. *plotERVRegions*
7. *ERVRegions*
8. *Full*

12.3.1 makeCleanBeds

`clean_beds.dir/GENE.bed` Bed files containing the co-ordinates all the ERV-like ORFs output by the Screen section.

12.3.2 makeCleanFastas

`clean_fastas.dir/GENE.fasta` FASTA files of the ERV-like ORFs output by the Screen section.

12.3.3 findERVRegions

`ERV_regions.dir/all_ORFs.bed` Concatenated version of the bed files output by the `makeCleanBeds` function with all three genes in a single file.

`ERV_regions.dir/all_regions.bed` Previous bed file merged using Bedtools merge, with regions within regions_maxdist merged.

`ERV_regions.dir/multi_gene_regions.bed` Filtered version of the previous bed file with only regions which were merged.

`ERV_regions.dir/regions.fasta` Fasta file of the merged regions.

12.3.4 makeRegionTables

`ERV_regions.dir/ERV_regions_final.tsv` Table showing the details of the combined regions containing ORFs resembling more than one retroviral gene. This table is identical to `erv_region_results.dir/results.tsv`.<br<

`ERV_regions.dir/ERV_regions_final.bed` Bed file with the co-ordinates of the identified regions.

`ERV_regions.dir/ERV_regions_final.fasta` FASTA file of the regions in the bed file above.

12.3.5 plotERVRegions

`ERV_region_plots.dir/*.FMT` Plots showing the distributions of ORFs resembling each retroviral gene on the genome. Each gene is shown on a different line on the y axis, the x axis is chromosome co-ordinates. One plot is generated for each multi-gene region.

12.3.6 summariseERVRegions

`erv_region_results.dir/results.tsv` Table showing the overall results for regions with ORFs resembling multiple retroviral genes. This table is described in full in the Main Output Files section.

`erv_region_results.dir/erv_regions.png` Bar chart showing the number of ERV regions identified with each combination of retroviral genes.