RepExpress: TE analysis pipeline

1. Obtaining and generating RepeatMasker TE gtf files from UCSC:

Downloads of TEs from repeatmasker tracks from UCSC are available from:

`https://genome.ucsc.edu/cgi-bin/hgTables`

but some steps are necessary to retrieve the data in an appropriate and complete form.

`assembly:` must be set to the required build (hg19)

`group:` set to '`Repeats`'

`output format:` set to '`all fields from selected table`' in order to retrieve details that are omitted from the '`GTF - gene transfer format (limited)`' format.

`output file:` should be set to something to avoid having the data sent directly for display by the browser (e.g. `hg19_ucsc_repeats.txt`).

2. Obtain genome and annotations:

This documentation describes the use of hg19 (GRCh37) which we have preferred for our work. Locations, file names and chromosome IDs for the genome tend to change with time - those used in this document relate to GRCh37 release 65 in which chromosome IDs are of the form: 1, 2, 3 .. X, Y, MT. Recent releases of the human genome from NCBI have inconvenient chromosome IDs, but the versions available from Gencode (at `https://www.gencodegenes.org/human/release_32lift37.html`) are more readable.

The GRCh37 files are most easily accessed by `ftp` from `ftp.ebi.ac.uk` in the directory `/pub/databases/gencode/Gencode_human/release_32/GRCh37_mapping`.

Gencode chromosome IDs are of the form chr1, chr2, etc.

Gene annotations are available in the same ftp directory as the file `gencode.v32lift37.annotation.gtf.gz` which can be uncompressed by:

```
gzip -dc gencode.v32lift37.annotation.gtf.gz > \
  gencode.v32lift37.annotation.gtf
```

3. Install STAR for genomic alignment

`STAR` (v 2.7.3a) is available from `https://github.com/alexdobin/STAR` and built and installed as documented.

4. **Installing `stringtie` for transcript assembly**

Stringtie (v 2.0.6) is available from `https://ccb.jhu.edu/software/stringtie/#install` and can be built and installed as documented.

5. **Installing featureCounts for TE quantification**

featureCounts (v 2.0.1) is available from `http://bioinf.wehi.edu.au/subread-package/` and can be built and installed as documented.

6. **Installing DMAP for adaptor trimming and TE characterisation**

DMAP (v 0.23) is available from `https://github.com/peterstockwell/DMAP` and can be built as described.

7. **Adaptor trimming of datasets**

It is necessary to remove Illumina primer sequences from any reads which contain them. If left, they will interfere with mapping. Various adaptor trimming packages exist: here we describe the DMAP program `cleanadaptors` which is an effective means of trimming, especially since it will retain correct pairing between the forward and reverse reads.

```
cleanadaptors -I contam.fa -z -x 20 \
 -F SRR3647483_1.fastq.gz -G SRR3647483_2.fastq.gz \
 -o SRR3647483_1_at.fastq.gz -O SRR3647483_2_at.fastq.gz
```

where:

`-I contam.fa` defines the file of adaptor sequences, in this case the fasta file of all current adaptors distributed with the DMAP package. The file in this case is assumed to be in the current directory along with the read 1 and 2 fastq data

`-z` directs that the source and, in this case output files, are gzip compressed

`-x 20` directs that any reads trimmed to less than 20bp should be omitted from the output. In general short reads will not map uniquely, so they can be rejected at this stage. 20 is a reasonable minimum to use.

`-F` & `-G` indicate the input files for reads 1 & 2 respectively

`-o` & `-O` indicate the output files for reads 1 & 2 respectively

For single read data the `-G` and `-O` options would be omitted. For uncompressed fastq the `-Z` (upper case) option can be used - this is the default. Mixed compression input and output can be indicated by the location on the command line of `-z` and `-Z` options.

8. RepExpress running

RepExpress requires all of the above software installed in an appropriate directory that is either specified in the **basic_params.sh** file (Appendix I) or in the working directory from where it is being run.

The commands given below assume that the RepExpress scripts are in the present working directory. It may be necessary to make the scripts executable by

```
chmod a+x *.sh
```

Steps:

8a. To generate gtf files in the required format and create the STAR index:

```
./generate_combined_gtfs.sh basic_params.sh
```

This step only needs to run once for a whole series of samples, creating:

STAR genome index files in the nominated directory (`star_genome_dir`)

Repeat sequence GTF file with unique identifiers in the `repeat_gene_gtf_dir` directory

Combined Repeat + Gene GTF file in file `gene_repeat_gtf` (defaults to `GenesPlusRepeats.gtf`)

Ensembl gene ID *vs* gene name file `ensid_vs_gname.txt` for appending gene names to genloc files.

8b. To map and count reads for each sample:

```
./map_count_reads.sh basic_params.sh map_params.sh
```

where each sample has its own **map_params.sh**, for which an example is in Appendix II. A series of such files (e.g. **map1_params.sh**, **map2_params.sh**, ...) should be used, one for each sample. This produces the following files:

STAR ouput:

```
<Read1FastaPrefix>Aligned.sortedByCoord.out.bam
<Read1FastaPrefix>Log.final.out
<Read1FastaPrefix>Log.out
<Read1FastaPrefix>Log.progress.out
<Read1FastaPrefix>ReadsPerGene.out.tab
<Read1FastaPrefix>SJ.out.tab
```

featureCounts multiple count, TPM and summary files:

```
<Read1FastaPrefix>M_FC.tpm
<Read1FastaPrefix>M_FC.txt
<Read1FastaPrefix>M_FC.txt.summary
```

featureCounts unique count, TPM and summary files:

```
<Read1FastaPrefix>U_FC.tpm
<Read1FastaPrefix>U_FC.txt
<Read1FastaPrefix>U_FC.txt.summary
```

stringtie gtf and gene abundance files:

```
<Read1FastaPrefix>gene_abund_e.out
<Read1FastaPrefix>stringtie_e.gtf
```

stringtie working files:

```
e2t.ctab
e_data.ctab
i2t.ctab
i_data.ctab
t_data.ctab
```

Combined **featureCounts** TPM files and the **identgeneloc** output with appended gene names:

```
<Read1FastaPrefix>M_U_FC.tpm
<Read1FastaPrefix>M_U_FC_tpm.genloc
```

Sorted version of **identgeneloc** file with **stringtie** gene abundance appended:

```
<Read1FastaPrefix>sorted_gene_abund.genloc
```

For the Appendix II example file `<Read1FastaPrefix>` would be `SRR2422921_1_at_`.

NOTE: the script will look for the **STAR** output file (`<Read1FastaPrefix>Aligned.sortedByCoord.out.bam`) and won't run **STAR** the file exists. In the event that the **STAR** run failed or produces a corrupted bam file, it is necessary to delete the empty or corrupted file in order to repeat the run.

8c. To combine expression counts for a series of related samples:

```
./compare_express.sh basic_params.sh combine_params.sh
```

where **combine_params.sh** (example in Appendix III) specifies the set of samples to be combine, either as a file of read 1 fastq file names or, (preferably) as a list of file names for the **genloc** output files from the runs of steps **8b**. An example list is in Appendix IV. The main output from this step is:

```
genloc_file_list.txt_combined.txt
```

where `genloc_file_list.txt` is the `genloc_name_file` value in `combine-params.sh`.

8d. To contrast expression counts for two different sample sets:

```
./contrast_express.sh basic_params.sh contrast_params.sh
```

where an example **contrast_params.sh** is in Appendix V. The parameters therein are the names of two combined expression output files, from step **8c** and criteria for count minima and valid sample percentages for each of the two groups. Output files from this step are:

```
contrast_params.sh_joined.txt
contrast_params.sh_passed.txt
```

where the former is the total set of joined records, the latter those that have passed the criteria in **contrast_params.sh**.

8e. Using ***DESeq2*** for differential gene expression for two different sample sets:

DESeq2 is a Bioconductor package run under *R* or *Rscript* for generating statistics on differential gene expression. The scripts provided with *RepExpress* take *featureCounts* unique expression tpm data files, generated by step 8b for each sample, produce an expression matrix count file and run *DESeq2*. The input parameter file (example as **deseq2_params.sh** in Apppendix VI) specifies lists of the *featureCounts* unique expression files and the sample groups (e.g. 'test' and 'control') for each set. Examples of unique expression file lists are in Appendix VIIa and VIIb.

The *Rscript* commands are based on https://lashlock.github.io/compbio/R_presentation.html, noting that some 6 Gb of RAM are needed and that large data sets will take an appreciable time to complete (the brain and testis data sets - 3 of each - took about 1.5hrs on a well-resourced Linux system). The *R* implementation must be built with an appropriate version of Bioconductor prior to running the **run_deseq2.sh** script. The version used in developing this script was Bioconductor v3.8, running under R v3.5.1.

The command for this is:

```
./run_deseq2.sh basic_params.sh deseq2_params.sh
```

which produces an output file as named in `deseq2_params.sh` defaulting to a messy name created from the expression file list names.

Other files:

Other working files are generated by RepExpress, most but not all will be removed if the **basic_params.sh** variable `delete_temp_files` is set to `"yes"`. It is also possible to redefine this by putting an appropriate value in each of the step-by-step parameter files.

Appendix I: `basic_params.sh`

```
# basic_params.sh: define basic RepExpress env variables to show:
#
# 1. Control verbosity as script runs
# 2. Locations of required executables (e.g. STAR, Stringtie,
featureCounts)
# 3. Locations of raw genomic sequence file(s) and generated index for
mapping:
# 4. Locations of gtf annotation files
#
# Edit this file to reflect the parameters required on your target system.
# This information is expected to suit a whole series of runs which would
# be made against the same genome and annotations.  Definitions for each
# individual run should be made in a run parameter script (run_params.sh,
# for instance).

# 1. Control verbosity: set to empty string to reduce feedback

verbose="yes";

# Control retention of working scripts, set this string to empty to retain
them

delete_temp_files="yes";

# 2.  Locations of required executables and invariant run parameters:

# path to STAR executable: can be left empty if STAR is already on your
exec PATH
#  The command 'which STAR' will indicate this for you.

path_to_star="";

# STAR mapping run parameters:

starfiltermax="150";
staranchormax="150";
starthreads="4";

# path to featureCounts executable: empty if on your path
# 'which featureCounts' will indicate this

path_to_featurecounts="";

# featureCounts parameters

featurecounts_threads="4";
featurecounts_overlap="--minOverlap 25";

# path to stringtie executable: empty if already on your path
# 'which stringtie' will indicate this

path_to_stringtie="";

# stringtie parameters

stringtie_threads="4";

# path to DMAP executables, particularly identgeneloc.
#  leave empty if these are already on your path
# 'which identgeneloc' will indicate this
```

```
path_to_dmap="";

# 3. Locations of raw genomic sequence file(s) and generated index for
mapping:

# genome fasta files - this is for all sequences in one large fasta file.
#  It is possible to have separate files for each chromosome, but it
#  tends to get a bit messy.

genome_fasta_file="/mnt/hcs/dsm-pathology-
ecclesRNA/Erin_Macaulay_placental/hs_gencode_GRCh37/GRCh37.primary_assembly
.genome.fa.gz"

# the location where the index files are written and read from

star_genome_dir="/mnt/hcs/dsm-pathology-
ecclesRNA/Erin_Macaulay_placental/sra_data/RepExpress_building/dsm_hs_ref_G
RCh37/";

# 4. Locations of gtf annotation files

# name of the UCSC repeat element source file downloaded from
# https://genome.ucsc.edu/cgi-bin/hgTables
# with web page settings:
#
# 'assembly': must be set to the required build
#
# 'group': set to Repeats
#
# 'output format': set to 'all fields from selected table' in order to
retrieve
#   details that are omitted from the GTF - gene transfer format (limited)
format.
#
# 'output file': should be set to something to avoid having the data sent
directly
#  for display by the browser (e.g. hg19_ucsc_repeats.txt).
#
# The file can be gzip compressed or not.  The gzip compression at the web
interface
#  didn't seem to work.

ucsc_repeat_src="/mnt/hcs/dsm-pathology-
ecclesRNA/Erin_Macaulay_placental/sra_data/STAR_chi/FeatureCount/hg19_ucsc_
repeats.txt.gz"

# name of gencode gene annotation gtf file, available from
# https://www.gencodegenes.org/human/release_32lift37.html or
# by ftp from ftp.ebi.ac.uk at
/pub/databases/gencode/Gencode_human/release_32/GRCh37_mapping
#
# The desired file for GRCh37/hg19 is gencode.v32lift37.annotation.gtf.gz
#
# The GRCh38 release is available from the related directory.
#

gencode_gene_gtf_src="/mnt/hcs/dsm-pathology-
ecclesRNA/Erin_Macaulay_placental/hs_gencode_GRCh37/gencode.v32lift37.annot
ation.gtf"

# Name of dir to save repeat and gencode gtfs: blank will use current
default
#  this assumes both are going to be in the same location.
```

```
repeat_gene_gtf_dir="/mnt/hcs/dsm-pathology-
ecclesRNA/Erin_Macaulay_placental/sra_data/RepExpress_building/dsm_hs_ref_G
RCh37/";

# Name of combined gene+repeat gtf file - leave blank for a default name

gene_repeat_gtf=""

# Stuff below here is combining and processing information from above.
# It should not be necessary to change anything below

# check the gencode gene gtf details:
# if it has a gzip extension then modify the name appropriately

if [[ "${gencode_gene_gtf_src}" == *".gz" ]]; then

  gencode_gene_gtf="${repeat_gene_gtf_dir}"$(basename
"${gencode_gene_gtf_src}" ".gz");


else

  gencode_gene_gtf="${gencode_gene_gtf_src}";

fi

if [[ -z ${gene_repeat_gtf} ]]; then

gene_repeat_gtf="${repeat_gene_gtf_dir}""GenesPlusRepeats.gtf"

fi

# name for ensembl ID vs gene name file

ensid_vs_gname="${repeat_gene_gtf_dir}""ensid_vs_gname.txt";

# name for unique repeat gtf file:

ucsc_repeats_uniq_gtf="${repeat_gene_gtf_dir}""ucsc_repeats_uniq.gtf"
```

Appendix II: `map_params.sh`

```
# map_params.sh: to define RepExpress env variables
# for individual mapping runs
#
# Variables that are consistent across a series of runs
# are already defined in basic_params.sh
#
# Edit this to reflect the parameters required for each mapping run.

# the output directory for STAR mapping results: leave blank for
# output into the current directory.  Don't put the value "./"
# there, since this will cause the stringtie run to fail.

#mapping_output_dir="./star_out/";
mapping_output_dir="";

# Your data files for mapping: Read1 required, Read 2 optional, leave blank if not
used
# can be gzip compressed if your file system allows this.

read1_fastq="/mnt/hcs/dsm-pathology-
ecclesRNA/Erin_Macaulay_placental/sra_data/SRR2422921/SRR2422921_1_at.fastq"
read2_fastq="/mnt/hcs/dsm-pathology-
ecclesRNA/Erin_Macaulay_placental/sra_data/SRR2422921/SRR2422921_2_at.fastq"

# featureCounts needs a strandedness parameter reflecting the way the
# library was generated.  Values are:
#   unstranded       "-s 0"  (default if setting is left blank, necessary for
single ended reads)
#   stranded         "-s 1"
#   reverse stranded "-s 2"

featurecounts_strandedness="-s 1"
```

Appendix III: **`combine_params.sh`**

```
# combine_params.sh: to define RepExpress env variables
# for combining GENLOC files generated by map_count_reads.sh
# runs.
#
# Variables that are consistent across a series of runs
# are already defined in basic_params.sh
#
# Edit this to reflect the parameters required for each combine run.

# the output directory for the combine: leave blank for
# output into the current directory.

combine_output_dir="./combined_out/";

# We need to define the source files, either the original
# fastq files or the derived genloc output files from
map_count_reads.
# At least one of these should be set to the name of such a file.


fastq_name_file="";

genloc_name_file="./genloc_file_list.txt";

# if fastq files are specified (fastq_name_file is used)
# we need to specify where the resulting genloc files are,
# noting that they will all need to be in the same directory.
# leave this blank if genloc_name_file is used or if all
# generated genloc files are in the current directory

genloc_file_dir="";
```

Appendix IV: `genloc_file_list.txt`

```
SRR2422921_1_at_M_U_FC_tpm.genloc
SRR2422922_1_at_M_U_FC_tpm.genloc
SRR2422924_1_at_M_U_FC_tpm.genloc
```

Appendix V: `contrast_params.sh`

```
# contrast_params.sh: defines RepExpress env variables
# to compare TE expression matrices for two
# different compare_express.sh runs which have previously been run
# on all the two sets of samples
#
# Variables that are consistent across a series of runs
# are already defined in basic_params.sh
#
# Edit this to reflect the parameters required for each contrast run.

# the output directory for the contrast: leave blank for
# output into the current directory.

contrast_output_dir="";

# We need to define the TE expression files containing
# the count matrices from previous compare_express.sh runs.

TE_matrix1_file="brain_genloc_list.txt_combined.txt";

TE_matrix2_file="testis_genloc_list.txt_combined.txt";

# Now define the minimum proportion (%) of samples needed to
# qualify for each set

matrix1_sample_min="75";

matrix2_sample_min="75";

# And then the minimum number of hit counts required for
# each sample - an integer value

matrix1_hit_min="50";

matrix2_hit_min="50";

# The above could have been defined with 1 value each, but
# separating them increases the flexibility.
```

Appendix VI: **`deseq2_params.sh`**

```
# DESeq2_params.sh: to define RepExpress env variables
# for running DESeq2 expression comparison on featureCounts
# unique counts files, or TPM files from that, generated
# by map_count_reads.sh runs.
#
# Variables that are consistent across a series of runs
# are already defined in basic_params.sh
#
# Edit this to reflect the parameters required for each DESeq2 run.

# the output directory for DESeq2: leave blank for
# output into the current directory.

deseq2_output_dir="";

# We need to define the count or tpm files by having a list of each set
in
# a text file, one per line.

deseq2_file_list_1="brain_tpm_list.txt";

sample_name_1="brain";

deseq2_file_list_2="testis_tpm_list.txt";

sample_name_2="testis";

# the sample_name values can be left blank and will default
# to "control" and "treatment"

# name for the final output file: can be left empty for a fairly
awkward default

DESeq2_output_file="brain_testis_deseq2_express.txt";
```

Appendix VIIIa: **`brain_tpm_list.txt`** example list of sample files for `desq2_params.sh`

```
../brain_testis_trial/SRR2422921_1_at_U_FC.tpm
../brain_testis_trial/SRR2422922_1_at_U_FC.tpm
../brain_testis_trial/SRR2422924_1_at_U_FC.tpm
```

Appendix VIIIb: **`testis_tpm_list.txt`** example list of sample files for `deseq2_params.sh`

```
../brain_testis_trial/SRR8454398_1_at_U_FC.tpm
../brain_testis_trial/SRR8454399_1_at_U_FC.tpm
../brain_testis_trial/SRR8454400_1_at_U_FC.tpm
```