ExplorATE vignette

Martin Femenias

5/12/2021

#ExporATE vignette

This vignette implements the ExplorATE package on different data sets, both for model and non-model organisms. You can find detailed information on each function in the users' guide.

1. Analysis with model organisms

1.1 Obtaining the data set

We will analyze the *Drosophila melanogaster* ovarian cells data set from Ohtani *et al.* (2013). Ohtani *et al.* observed negative regulation of transposable elements after altering a *Drosophila* homologous of gametocyte-specific factor 1 (DmGTSF1). DmGTSF1 mutations caused the derepression of transposons suggesting that DmGTSF1 is an integral factor in Piwi-piRISC-mediated transcriptional silencing. The FASTQ files are available from Gene Expression Omnibus (accession no. `GSE47006`). For this vignette, we will use the control samples (`SRR851837`) and Piwi (`SRR851838`). First, create a working directory (here `dm_ExplorATE_wd`) in the desired location and navigate to it:

```
mkdir dm_ExplorATE_wd
cd dm_ExplorATE_wd
```

You can use the SRA Toolkit to download the files. To install SRA Toolkit follow the detailed instructions here. If you have an older SRA Toolkit version (releases <2.9.1) you can download and format the headers to be used with Trinity as shown below:

```
fastq-dump --defline-seq '@$sn[_$rn]/$ri' --split-files --accession SRR851837 > SRR851837.log
fastq-dump --defline-seq '@$sn[_$rn]/$ri' --split-files --accession SRR851838 > SRR851838.log
```

If the files downloaded successfully you should get something like this:

```
tail *.log

==> SRR851837.fastq-dump.log <==
Read 42134407 spots for SRR851837
Written 42134407 spots for SRR851837

==> SRR851838.fastq-dump.log <==
Read 48277060 spots for SRR851838
Written 48277060 spots for SRR851838
```

Once the files are downloaded, you have to rename them:

```
mv SRR851837.fastq control.fastq
mv SRR851838.fastq piwi.fastq
```

If you are using a version of SRA Toolkit 2.9.1 or higher you can use the fasterq-dump tool (much faster) as detailed in the program's web page, and later format the headers as follows. Replace `<threads>`argument appropriately and run:

```
fasterq-dump SRR851837 -e <threads> -t /dev/shm -p > SRR851837.log
fasterq-dump SRR851838 -e <threads> -t /dev/shm -p > SRR851838.log
```

If the files downloaded successfully you should get something like this:

```
tail *.log
spots read      : 42,134,407
reads read      : 42,134,407
reads written   : 42,134,407

spots read      : 48,277,060
reads read      : 48,277,060
reads written   : 48,277,060
```

Now, you have to modify the headers:

```
for i in SRR*;
do cat $i | sed 's/\sl:\w*\sm:\w*\sh:\w*//' | sed -r '/(^[\@]SRR\S+ )/s/\slength=\w*$'/\\/1/ |
sed -r 's/(^[\@]SRR\S+ )/@/' | sed -r 's/(^[\+]SRR\S+ )/+/' > headerok_$i;
done
```

This code above formats the headers to be used with Trinity and creates a new copy with the headerok_ tag. Verify that the file was modified correctly:

```
for i in headerok_*;
do grep -c '^@HW' $i;
done
```

If the files are correct, rename them:

```
mv headerok_SRR851837.fastq control.fastq
mv headerok_SRR851838.fastq piwi.fastq
```

Compress the .fastq files and move them to a new "reads" folder.

```
mkdir reads
gzip *.fastq
mv *.gz reads
```

Next, we will download other necessary files. All of these files are in the ExplorATE_data_test/drosophila directory. The user can clone the directory and continue with step #1.2 or download them one by one as shown below: Downloading the reference genome for *D. melanogaster*

```
wget https://hgdownload.soe.ucsc.edu/goldenPath/dm3/bigZips/dm3.fa.gz
```

Downloading the GFF annotation file

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/dm3/bigZips/genes/dm3.refGene.gtf.gz
```

Downloading the transposable elements annotation file from RepeatMasker

```
wget https://www.repeatmasker.org/genomes/dm3/RepeatMasker-rm405-db20140131/dm3.fa.out.gz
```

Renaming files

```
mv dm3.fa.gz dm_genome.fa.gz
mv dm3.refGene.gtf.gz dm_genannot.gtf.gz
mv dm3.fa.out.gz dm_RMgen.out.gz
```

uncompress .gz files

```
gzip -d *.gz
```

The next step is to carry out the *de novo* assembly of the transcriptome. To run this vignette we provide a *de novo* transcriptome for *Drosophila melanogaster* that you can download with the code below. We suggest that you follow the recommendations detailed in this link when analyzing your data. Downloading the *de novo* transcriptome

```
wget https://github.com/FemeniasM/ExplorATE_data_test/drosophila/dm_transcriptome.fa.gz
```

Next, you have to mask the transcriptome with RepeatMasker. You should use a library suitable for the study organism (see the user guide for details). You can also incorporate repeats *de novo* for the species of interest. This link provides useful information on how to generate *de novo* libraries and combine them with RepeatMasker libraries. For the *Drosophila* dataset we will use the RepeatMasker database assigning as "DNA source" `Drosophila`.The user can download a file provided by the program or can generate it using RepeatMasker. The two options are shown below: Downloading the file provided by ExplorATE:

```
wget https://github.com/FemeniasM/ExplorATE_data_test/drosophila/dm_RMtr.out.gz
```

Generating the file with RepeatMasker:

```
RepeatMasker -species Drosophila -a -e rmblast -pa <threads> -nolow -xm -u -gff
-dir <output directory> dm_transcriptome.fa
```

Replace `<threads>` and `<output directory>` arguments appropriately.

Alternatively you can use the RepeatMasker web service to get your RepeatMasker output file. In this case, select the appropriate species in "DNA source" (`Fruit fly (Drosophila melanogaster)` to our example) and specify "tar file" as "Return Format". Then go to the item "Lineage Specific Masking" and select "Do not mask satellites and simple repeats". Once the run is finished, RepeatMasker will offer you a link "Masked sequence and matches in compressed format" where you can download the files. Then extract the files, place them in your working directory and rename them as follows:

```
mv dm_transcriptome.fa_*.out dm_RMtr.out
```

You should have the files `dm_genome.fa`, `dm_genannot.gtf`, `dm_RMgen.out`, `dm_transcriptome.fa` and `dm_RMtr.out` in your desired working directory (here `dm_ExplorATE_wd`) and within this directory a "`reads`" folder where the `control.fastq.gz` and `piwi.fastq.gz` files are located.

1.2 Run ExplorATE in model organisms

ExplorATE uses Selective Alignment strategy and decoys (Srivastava et al. 2020) of Salmon to solve the multimapping and co-transcription drawbacks in transposable elements quantification. Therefore, a reference of active transposable elements must first be created from the transcriptome, and then it takes the information from the genome repeats to supply decoys derived from intronic regions, UTRs, and other repetitive regions. You can generate these references in two ways, either using a shell script provided by the program or with the `ExplorATE::mk.references_mo()` function. This function will create a sorted sequence file `trmeSalmon.fasta` and a decoys file `decoys.txt` to subsequent Salmon run. To run the shell script, first download the script and place it in a directory on your system:

```
wget https://github.com/FemeniasM/ExplorATEtools/mk.references_mo.sh
```

Assign appropriate arguments `<threads>` and `<bedtools binary path>`, and run:

```
bash mk.references_mo.sh -p <threads> -b <bedtools binary path> -a dm_genannot.gtf -g dm_genome.fa
-t dm_transcriptome.fa -o . -r dm_RMtr.out -s dm_RMgen.out -j 'HS' -c 'namRep'
```

By default, this script assumes `awk` and `bedtools` to be available in your `$PATH` environment variable.

Using the R function from the ExplorATE package:

```
setwd("path/to/dm_ExplorATE_wd")#Assign the folder with input files as the working directory
dm_ref <- ExplorATE::mk.reference_mo(bedtools="bedtools",
                        GFFgen="dm_genannot.gtf",
                        genome="dm_genome.fa",
                        trme="dm_transcriptome.fa",
                        RMgen="dm_RMgen.out",
                        RMtr="dm_RMtr.out",
                        overlapping=T,
                        over.res="HS",
                        by="namRep",
                        threads=12,
                        outdir=".")
```

Once the references are created you can run Salmon. Linux users can use the `run.salmon()` function in R or run Salmon with the appropriate arguments.

```
ExplorATE::run.salmon(index = "index_dm",
                      decoys = "decoys.txt",
                      salmon_path = "path/to/salmon-latest_linux_x86_64/bin/salmon",
                      pe_se = "se",
                      kmer = 31,
                      trme = "trmeSalmon.fasta",
                      lib_dir = "reads",
                      threads = 12
                      )
```

The above function uses the `decoys.txt` and `trmeSalmon.fasta` files generated by `mk.references_mo.sh` or `mk.reference_mo()` functions. If you decide to run Salmon in another way, be sure to use these files and the –gcBias, –validateMappings, –useVBOpt flags. Next, the user must import the Salmon estimates into the R environment and will be able to perform the differential expression analyzes as shown in section 2.3

2. Analysis with non-model organisms

2.1 Obtaining the data set

We will analyze the data set for *Liolaemus parthenos*, the only parthenogenetic lizard of the entire Pleurodonta (Iguanidae) clade. The FASTQ files are available from Gene Expression Omnibus (accession no.`GSE173261`). However in this vignette, we will use a data set with FASTQ files randomly sampled from the original file in order to reduce library sizes and generate replicates for differential expression analysis. Similarly, we randomly sampled 1,000,000 rows from the RepeatMasker output file and reduced the transcriptome size to run the pipeline in a reasonable time for the vignette. First clone the repository to your local directory and descompress all files.

```
gzip -d Lp_*
```

The `Lparthenos` folder contains the output files for `BLAST`, `TransDecoder`, `RepeatMasker`, and the *de novo* transcriptome. You can use these files directly (continuing with step #2.2) or generate them as shown below. ExplorATE has a script that can help users to generate the input files. First download the `mkinputs.sh` file:

```
wget https://github.com/FemeniasM/ExplorATEtools/mkinputs.sh
```

And run the script:

```
bash mkinputs.sh -p <threads> -b <blastp binary path> -h <hmmer binary path>
-r <RepeatMasker binary path> -d <TransDecoder directory path> -s <swissprot database>
-f <Pfam database> -l <Repeats library> -t <transcriptome file> -o <output directory>
```

The script will generate the corresponding input files, except for the transcriptome which must be *de novo*

assembled by the user or use the provided transcriptome(`Lp_trme.fasta`). For the execution of the next steps in the vignette we assume that all the input files are in a folder named `Lparthenos`.

2.2 Making TEs references and quantifying abundances

The first step will be to assign the location of the `Lparthenos` folder as the working directory in the R environment.

```
setwd("path/to/folder/Lparthenos")
Lp.references <- ExplorATE::mk.reference(
                                        RepMask = "Lp_RM.out",
                                        gff3 = "Lp_anot.gff3",
                                        anot = "Lp_blastx.outfmt6",
                                        cleanTEsProt = T,
                                        featureSum = T,
                                        outdir = "Lparthenos_out",
                                        rm.cotrans = T,
                                        overlapping = T,
                                        trme = "Lp_trme.fasta",
                                        stranded = F,
                                        by = "classRep",
                                        threads = 12,
                                        rule = c(0,0,0),
                                        over.res= "HS",
                                        annot_by = "transcripts"
                                        )
```

First, we will get a message `"please press 'y' if the names are correct"` to verify that the names assigned to each repetition do not contain ambiguities (consult the users' guide. Press `'y'+ENTER` and you will continue with the exclusion of elements co-transcribed with genes and the resolution of overlaps by the highest score. Finally, the annotation of transcripts by class of the repeats will be carried out and the `decoys.txt` and `trmeSalmon.fasta` files will be returned for the estimation of abundances in Salmon.

Next, we carry out the quantification with Salmon:

```
ExplorATE::run.salmon(index = "Lparthenos_index",
                      decoys = "Lparthenos_out/decoys.txt",
                      salmon_path = "~/programs/salmon-latest_linux_x86_64/bin/salmon",
                      pe_se = "pe",
                      kmer = 31,
                      trme = "Lparthenos_out/trmeSalmon.fasta",
                      lib_dir = "reads/",
                      threads = 12
)
```

2.3 Importing estimates to R and estimating differential expression

The next step is to import the estimates to perform the differential expression analysis later. The `import.RTEs()` function allows to import the estimates and create at the same time `DGEList` or `DESeqDataSet` objects ready-to-use in edgeR or DESeq respectively. The user must perform the differential expression analysis conveniently for their data. Below is an example of how to import the estimates and perform the expression analysis in edgeR:

```
y <- ExplorATE::import.RTEs(
                      path.sal = "salmon_out",
                      ref.sal = Lp.references,
                      import_to = "edgeR",
```

```
                        conditions = rep(c("Brain", "Liver", "Ovary"), each = 3)
                        )
```

Now "y" is a `DGEList` object that is ready to perform the dispersion estimation. We explore the data using an MDS plot from the limma package:

```
limma::plotMDS(y)
abline(h=0, v=0)
```

Since we will use a GLM approach to compare the tissues, we must first construct a design matrix. And then we will estimate dispersions using the `estimateDisp()` function:

```
group.0 <- factor(rep(c("Brain", "Liver","Ovary"), each = 3))
des <- model.matrix( ~ 0 + group.0, data = y$samples)
colnames(des) <- c("Brain", "Liver","Ovary")
y <- edgeR::estimateDisp(y,des)
```

And finally we determine the differential expression using quasi-likelihood (QL) F-test:

```
fit_qlf <- edgeR::glmQLFit(y, des)
my.contrasts <- limma::makeContrasts(OvsB=Ovary-Brain,
                                     OvsL=Ovary-Liver,
                                     BvsL=Brain-Liver,
                                     levels=des)

qlf_OvsB <- edgeR::glmQLFTest(fit_qlf, contrast=my.contrasts[,"OvsB"])
qlf_OvsL <- edgeR::glmQLFTest(fit_qlf, contrast=my.contrasts[,"OvsL"])
qlf_BvsL <- edgeR::glmQLFTest(fit_qlf, contrast=my.contrasts[,"BvsL"])
```

We explore the differentially expressed TEs in each contrast with FDR $<.05$:

```
topTags_OvsB <- edgeR::topTags(qlf_OvsB, n = nrow(qlf_OvsB$table),sort.by = "none")
topTags_OvsB[topTags_OvsB$table$FDR<.05,]

topTags_OvsL <- edgeR::topTags(qlf_OvsL, n = nrow(qlf_OvsL$table),sort.by = "none")
topTags_OvsL[topTags_OvsL$table$FDR<.05,]

topTags_BvsL <- edgeR::topTags(qlf_BvsL, n = nrow(qlf_BvsL$table),sort.by = "none")
topTags_BvsL[topTags_BvsL$table$FDR<.05,]
```