T. brucei co-expression network analysis

Kennedy Mwangi

April 15, 2019

Introduction

This document contains the workflow used in the analysis of *T. brucei* gene co-expression network analysis. It contains code used in each step of the analysis.

Data acquisition

Data used in this study is obtained from European Nucleotide Archive under accession number SRP002243.

First, metadata for the data is obtained from EBI as follows:

```
#Obtain metadata information for the data used in this study from ENA and SRA databases.

# ENA metadata
# code adopted from: https://wiki.bits.vib.be/index.php/Download_read_information_and_FASTQ_data_from_t
accession <- "SRP002243"
ena.url <- paste("http://www.ebi.ac.uk/ena/data/warehouse/filereport?accession=",
                 accession,
                 "&result=read_run",
                 "&fields=run_accession,library_name,",
                 "read_count,fastq_ftp,fastq_aspera,",
                 "fastq_galaxy,sra_ftp,sra_aspera,sra_galaxy,",
                 "&download=text",
                 sep="")
ENA.metadata <- read.table(url(ena.url), header=TRUE, sep="\t")

# SRA metadata
SRA.metadata <- read.table("../data/SraRunTable.metadata.txt", header = TRUE, sep = "\t")

# Obtain sample metadata to be used later in analysis.
matches <- c("Run","Library_Name","Sample_Name")
sample.metadata <- SRA.metadata[grepl(paste(matches, collapse="|"), names(SRA.metadata))]

# create a text file with urls to fastq files in ENA database
fastq.urls <- ENA.metadata[grepl("fastq_ftp", names(ENA.metadata))]
write.csv(fastq.urls, file="fastq.urls.txt", eol = "\r\n", quote = FALSE, row.names = FALSE)
```

Next, RNASeq data is downloaded from EBI database's FTP site.

```
cat ../scripts/fastq_download.sh
```

```
## #!/bin/bash
## #
## #Script to download fastq files from European Nucleotide Archive
## #
## FILE=$1 #File containing fastq url links to EBI FTP site
##
## OUT_DIR=../data/raw_data/
```

```
##
## cat ${FILE} | xargs -n1 wget$2 -P ${OUT_DIR}
```

Some of the downstream tools require that FASTQ files downloaded in zipped form are unzipped.

```
cat ../scripts/unzip.sh
```

```
## #!/bin/bash
## #
## #Script to decompress fastq.gz files
## #
## FASTQ_FILES=../data/raw_data/*.fastq.gz
##
## for file in ${FASTQ_FILES}; do
##     gunzip ${file}
## done
```

Data quality assessment

After downloading the RNASeq data, its quality is checked through the FASTQC tool whose output is a report in HTML format.

```
cat ../scripts/fastqc_reports.sh
```

```
## #!/bin/bash
## #
## #Script to run FastQC reports using the FastQC tool
## #
## #load fastqc module
## module load fastqc/0.11.4
##
## FASTQ_DIR=../data/raw_data/*.fastq
##
## #create output directory if it doesn't exist.
## mkdir -p ../results/fastqc_reports
##
## REPORTS_DIR=../results/fastqc_reports/
##
## for file in ${FASTQ_DIR}; do
##    fastqc -f fastq -o ${REPORTS_DIR} ${file}
## done
```

Downloading *T. brucei* genome and GFF files

After determining that the data is of good quality and that no trimming is required, the reads are aligned on the *T. brucei* genome obtained from TriTrypDB database.

The genome and the Gene Feature Format (GFF) files are downloaded from the TriTrypDB database as follows:

```
#Downloading the genome

#wget https://tritrypdb.org/common/downloads/release-42/TbruceiTREU927/fasta/data/\
#TriTrypDB-42_TbruceiTREU927_Genome.fasta -P ../data/tbrucei_genome/
```

```
#Downloading the GFF file
#wget https://tritrypdb.org/common/downloads/release-42/TbruceiTREU927/gff/data/\
#TriTrypDB-42_TbruceiTREU927.gff -P ../data/genome_annotations_GFF/
```

Alignment of reads on the genome

Here, HISAT2 is used to align reads on the *T. brucei* genome. The first step is indexing the genome using HISAT2 followed by alignment of the reads. The output is SAM files.

Indexing the genome

```
cat ../scripts/hisat2_index.sh
```

```
## #!/bin/bash
## #
## #Script to index T. brucei genome using HISAT2
## #
## module load hisat/2-2.1.0
##
## #Create directory for HISAT2 indexed genome
## mkdir -p ../data/HISAT2_indexed_genome
##
## GENOME_FILE=$1
##
## cd ../data/HISAT2_indexed_genome/
##
## hisat2-build ${GENOME_FILE} tbrucei_genome_index_hisat2
```

Aligning the reads to the genome

```
cat ../scripts/hisat2_align.sh
```

```
## #!/bin/bash
## #
## #Script to align T. brucei reads to the indexed genome using HISAT2
## #
## module load hisat/2-2.1.0
##
## #change directory to that of the indexed genome
## cd ../data/HISAT2_indexed_genome/
##
## for fastq in ../raw_data/*.fastq; do
##     fqname=$(echo $fastq | cut -f1 -d '.')
##
##      hisat2 \
##       -x tbrucei_genome_index_hisat2 \
##       -U ${fastq} \
##       -S ${fqname}.sam \
##       -p 8 \
##      --summary-file ${fqname}.txt \
```

```
##      --new-summary
## done
##
## #make directories and move created files into them
##
## mkdir -p ../processed_data
## mkdir -p ../../results/hisat2_alignment_summary
##
## mv ../raw_data/*.sam ../processed_data/
## mv ../raw_data/SRR*.txt ../../results/hisat2_alignment_summary/
```

Reads quantification

HTSeq tool is used to count reads that aligned to the *T. brucei* genome. The output is a text file for each sample that contains the number of reads that were counted for each gene.

```
cat ../scripts/htseq_counts.sh
```

```
## #!/bin/bash
## #
## #Script to counts the number of reads aligned to T. brucei genome using HTSeq.
## #Resource: HTSeq documentation https://htseq.readthedocs.io/en/latest/count.html
## #
## module load htseq/0.11.2
##
## #create output directory if it doesn't exist
## mkdir -p ../results/HTSeq_count_results
##
## GFF_FILE=$1
##
## for sam_file in ../data/processed_data/*.sam; do
##     sam_file_name=$(echo $sam_file | cut -f1 -d '.')
##
##         python /opt/apps/htseq/0.11.2/bin/htseq-count \
##             -f sam \
##             -s no \
##             -t exon \
##             -i Parent \
##             $sam_file \
##             $GFF_FILE \
##             > ../results/HTSeq_count_results/${sam_file_name}.counts.txt
## done
```

Generating MultiQC report

MultiQC aggregates results from FASTQC, HISAT2 and HTSeq analysis into an HTML formatted single report for better visualization.

```
#change directory to results
cd ../results

#Run multiqc
#multiqc .
```

Analysis in R

Setting up R for the analysis

```
# Loading required R packages
library(edgeR)
```

```
## Loading required package: limma
```

Importing samples count data into R

For further analysis, samples read counts are read into R.

```
cat ../scripts/htseq-combine_all.R
```

```
## #!/usr/bin/Rscript
##
## # Take 'all' htseq-count results and melt them in to one big dataframe
##
## #Adapted from: https://wiki.bits.vib.be/index.php/NGS_RNASeq_DE_Exercise.4
##
## # where are we?
## basedir <- "../results"
## setwd(basedir)
##
## cntdir <- paste(basedir, "HTSeq_count_results", sep="/")
## pat <- ".counts.txt"
## hisat2.all <- list.files(path = cntdir,
##                          pattern = pat,
##                          all.files = TRUE,
##                          recursive = FALSE,
##                          ignore.case = FALSE,
##                          include.dirs = FALSE)
##
## # we choose the 'all' series
## myfiles <- hisat2.all
## DT <- list()
##
## # read each file as array element of DT and rename the last 2 cols
## # we created a list of single sample tables
## for (i in 1:length(myfiles) ) {
##   infile = paste(cntdir, myfiles[i], sep = "/")
##   DT[[myfiles[i]]] <- read.table(infile, header = F, stringsAsFactors = FALSE)
##   cnts <- gsub("(.*).counts.txt", "\\1", myfiles[i])
##   colnames(DT[[myfiles[i]]]) <- c("ID", cnts)
## }
##
## # merge all elements based on first ID columns
## data <- DT[[myfiles[1]]]
##
## # inspect
## head(data)
##
## # we now add each other table with the ID column as key
```

```
## for (i in 2:length(myfiles)) {
##   y <- DT[[myfiles[i]]]
##   z <- merge(data, y, by = c("ID"))
##   data <- z
## }
##
## # ID column becomes rownames
## rownames(data) <- data$ID
## data <- data[,-1]
##
## ## add total counts per sample
## data <- rbind(data, tot.counts=colSums(data))
##
## # inspect and look at the top row names!
## head(data)
##
## tail(data)
##
## ####################################
## # take summary rows to a new table
## # ( not starting with Tb and tmp with invert=TRUE )
##
## # transpose table for readability
## data.all.summary <- data[grep("^Tb|^tmp", rownames(data), perl=TRUE, invert=TRUE), ]
##
## # review
## data.all.summary
##
## # transpose table
## t(data.all.summary)
##
## # write summary to file
## write.csv(data.all.summary, file = "htseq_counts_all-summary.csv")
##
## ####################################
## # take all data rows to a new table
##
## data.all <- data[grep("^Tb|^tmp", rownames(data), perl=TRUE, invert=FALSE), ]
##
## # inspect final merged table
## head(data.all, 3)
##
## # write data to file
## write.table(data.all, file = "htseq_counts_all.txt", quote = FALSE, sep = "\t")
##
## # cleanup intermediate objects
## rm(y, z, i, DT)
```