Breast Cancer Circulating Tumor Cells Analysis

January 12, 2021

Introduction

This document contains the workflow used in the analysis of Breast Cancer Circulating Tumor Cells (CTCs) analysis. It contains code used in each step of the analysis. The code and output files are archived at https://github.com/wanjauk/breast_cancer_ctc

Setting up R for the analysis

Metadata acquisition

Data used in this study is obtained from European Nucleotide Archive under accession number SRP135702.

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

```r
#!/usr/bin/env Rscript
#Obtain FASTQ urls to download the data used in this study from ENA database.

# ENA metadata
# code adopted from:
# https://wiki.bits.vib.be/index.php/Download_read_information_and_FASTQ_data_from_the_SRA

# required libraries
library(tidyverse)
library(here)

# accession numbers from Lang et al (PMID: 29868978) - (doi:10.1245/s10434-018-6540-4)
accession_num <- "SRP135702"

# create directories to store the files if they don't exist.
lang_dir <- here::here("data", "raw", "lang_raw")
if (!dir.exists(lang_dir)) {
  dir.create(lang_dir, recursive=TRUE)
}

    # construct the url to ENA database
    ena.url <- paste("https://www.ebi.ac.uk/ena/portal/api/filereport?accession=",
                     accession_num,
                     "&result=read_run",
                     "&fields=study_accession,sample_accession,",
                     "experiment_accession,run_accession,scientific_name,fastq_ftp,",
                     "submitted_ftp,sra_ftp",
                     "&download=true",
                     sep="")

    # Load the metadata from ENA
    ENA.metadata <- read.table(url(ena.url), header=TRUE, sep="\t")

    # coerce the fastq_ftp column from factor to character.
    ENA.metadata$fastq_ftp <- as.character(ENA.metadata$fastq_ftp)

    # ensure that R1 and R2 fastq files are in separate rows
    ENA.metadata <- ENA.metadata %>% separate_rows(fastq_ftp, sep=";")

    # get the fastq urls
    fastq.urls <- ENA.metadata[grepl("fastq_ftp", names(ENA.metadata))]

    # create a text file with urls to fastq files in ENA database
    write.table(fastq.urls, here::here(lang_dir, "lang.fastq.urls.txt"),
                eol = "\n",
                quote = FALSE,
                col.names = FALSE,
                row.names = FALSE)
```

Prepare the samples metadata.

```r
# Load SRA metadata from Savage et al 2016 study
SRA.metadata <- read.table(here::here("data", "raw", "savage","SraRunTable.metadata.txt"),
                           header = TRUE, sep = "\t")

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

# create a grouping factor that will place each sample in the one of three tissues i.e.
# midgut (MG), proventriculus(PV) and salivary glands (SG)
tissue <- factor(c("MG", "MG", "MG", "MG", "MG", "PV", "PV", "SG", "SG", "SG", "SG",
                   "MG", "MG", "PV", "SG", "SG", "PV"))

# append factor to samples.metadata to group samples
samples.metadata["Tissue"] <- tissue

# Add sample from Telleria et al 2014 study (SRR965341) to sample metadata.
samples.metadata$Run <- as.character(samples.metadata$Run)
samples.metadata <- rbind(samples.metadata, "18" = c("SA2", "SRR965341", "SA2", "SG"))

# include batch information for the 18 samples
batch <- factor(c(1, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2))
samples.metadata["Batch"] <- batch

# save the metadata to an R object
saveRDS(samples.metadata, here::here("data", "raw", "samples.metadata.RDS"))
```

```r
# print out the sample metadata table
samples.metadata
```

Download the FASTQ data

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

```
#!/usr/bin/env bash
# script generated using https://sra-explorer.info/# (Accession number: SRP135702)

# change directory to output directory.
cd /data/kwanjau/lang

ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
ascp -QT -l 300m -P33001 -i $HOME/.aspera/connect/etc/asperaweb_id_dsa.openssh era-fasp@fasp.sra.ebi.ac
```

Downloading Human genome and annotation files

Genomes are obtained from their respective databases before alignment.

```
# Download genome and annotation files from:
# https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.33_GRCh38.p7/
# Download genome Build 38 patch release 7 (GRCh38.p7)
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.33_GRCh38.p7/GCF_000001405.3
-P ../../data/scratch/

# Download GFF file
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/GCF_000001405.33_GRCh38.p7/GCF_000001405.3
-P ../../data/scratch

# Download GTF file associated with GRCh38.p7 from https://www.gencodegenes.org/human/release_25.html
wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_25/gencode.v25.annotation.gtf.gz \
-P ../../data/scratch

# unzip the genome file
gunzip ../../data/scratch/GCF_000001405.33_GRCh38.p7_genomic.fna.gz

# unzip the gff file
gunzip ../../data/scratch/GCF_000001405.33_GRCh38.p7_genomic.gff.gz

# unzip the GTF file
gunzip ../../data/scratch/gencode.v25.annotation.gtf.gz
```

Data quality assessment - FASTQC

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

```
# #!/bin/bash
#
#Script to generate FastQC reports using the FastQC tool
#
# USAGE:
# bash generate-fastqc-reports.sh ../../results/figures/fastqc_reports /data/kwanjau_shared/lang_data
#
# make directory to store the results
mkdir -p ../../results/figures/fastqc_reports

# fastqc reports directory
REPORT_DIR=$1

# fastq files directory
FASTQ_DIR=$2

for file in $FASTQ_DIR/*.fastq.gz; do
   fastqc ${file} -o ${REPORT_DIR} -f fastq
done
```

Aligning the reads to the genome (Read Mapping)

Alignment of the reads to the genome. The output is BAM files.

```
# #!/bin/bash
#
```

```
#Script to align reads to the indexed genome using HISAT2
#

# load the samtools module
module load samtools/1.9

#USAGE:
# ./reads-alignment.sh \
# /data/kwanjau_shared/lang_data \
# ../../data/scratch/indexed_genome \
# /data/kwanjau_shared/scratch/reads-alignment-output \
# /data/kwanjau_shared/scratch/bam-output


# create alignment output directory
mkdir -p /data/kwanjau_shared/scratch/reads-alignment-output
mkdir -p /data/kwanjau_shared/scratch/bam-output

# Fastq directory
FASTQ_DIR=$1

# genome index directory
INDEX_DIR=$2

# alignment output directory
ALIGN_OUT=$3

# bam file output
BAM_OUT=$4

for fastq in `ls -1 ${FASTQ_DIR}/*_1.fastq.gz | sed 's/_1.fastq.gz//'` ; do
fqname=$(basename "$fastq" .fastq)

hisat2 \
         -x ${INDEX_DIR}/GCF_000001405.33_GRCh38.p7_index_hisat2 \
         -1 ${fastq}*_1.fastq.gz \
     -2 ${fastq}*_2.fastq.gz \
         -p 16 \
        --summary-file ${ALIGN_OUT}/${fqname}.txt \
        --new-summary | \
        samtools sort -@ 16 -o ${BAM_OUT}/${fqname}.sorted.bam

done
```

Assessing the strandedness of the reads

Before this quality control can be performed, we need to verify whether the reads are stranded or not as this is a required parameter for HTSeq tool later in the analysis. This can be done using RSeQC package - An RNA-seq Quality Control Package. `infer_experiment.py` module is used in this case. RSeQC documentation and tutorial can be found here.

First we convert the Human genome annotation GTF file into `bed` format required by RSeQC package. Then we use `infer_experiment.py` to verify strandedness using a few samples.

```
# convert GTF genome annotation to BED format using a custom script from:
#https://github.com/ExpressionAnalysis/ea-utils/tree/master/clipper/gtf2bed

# USAGE:
# # savage data
# ./check-reads-strandedness.sh \
# ../../data/scratch/tbrucei/TriTrypDB-43_TbruceiTREU927.gtf \
# ../../data/scratch/tbrucei/TriTrypDB-43_TbruceiTREU927.bed \
# ../../data/scratch/sam-to-bam-output/savage/SRR039381.sorted.bam

# # telleria data
# ./check-reads-strandedness.sh \
# ../../data/scratch/tbrucei/TriTrypDB-43_TbruceiTREU927.gtf \
# ../../data/scratch/tbrucei/TriTrypDB-43_TbruceiTREU927.bed \
# ../../data/scratch/sam-to-bam-output/telleria/SRR965341.sorted.bam

# GTF file
GTF_FILE=$1

# BED file directory
BED_FILE=$2

# BAM file
BAM_FILE=$3


# create a BED file from GTF
# ../utils/gtf2bed.pl $GTF_FILE > $BED_FILE

infer_experiment.py -i $BAM_FILE -r $BED_FILE

# output for savage's SRR039381.sorted.bam
# This is SingleEnd Data
# Fraction of reads failed to determine: 0.0000
# Fraction of reads explained by "++,--": 0.4058
# Fraction of reads explained by "+-,-+": 0.5942

# output for telleria's SRR965341.sorted.bam
# This is PairEnd Data
# Fraction of reads failed to determine: 0.0019
# Fraction of reads explained by "1++,1--,2+-,2-+": 0.4999
# Fraction of reads explained by "1+-,1-+,2++,2--": 0.4982
```

Reads quantification

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

```
# #!/bin/bash
#
#Script to counts the number of reads aligned to Human genome using HTSeq.
#Resource: HTSeq documentation https://htseq.readthedocs.io/en/latest/count.html
#
```

```
# load the htseq module
module load htseq/0.9

# USAGE:
# ./reads-quantification.sh \
# /data/kwanjau_shared/scratch/reads-alignment-output \
# ../../data/scratch/gencode.v25.annotation.gtf \
# ../../data/intermediate/read_counts


# make directory for reads count output from htseq
mkdir -p ../../data/intermediate/read_counts


# path to bam files
BAM_DIR=$1

# T. brucei genome annotation file.
GTF_FILE=$2

# read counts path
READ_COUNTS_DIR=$3

# reads counting (Savage single-end reads)
for bam_file in ${BAM_DIR}/*.sorted.bam; do
    bam_file_name=$(basename "$bam_file" .sorted.bam)

        htseq-count \
            -f bam \
            -r pos \
            -s no \
            -t exon \
            -i gene_id \
            $bam_file \
            $GTF_FILE \
            > ${READ_COUNTS_DIR}/${bam_file_name}.counts.txt
done
```

Generating MultiQC report

```
# #!/bin/bash
#
#Script to generate MultiQC reports using the MultiQC tool
#
# USAGE:
# bash generate-multiqc-reports.sh \
# ../../results/figures/fastqc_reports/savage \
# ../../data/scratch/reads-alignment-output/savage \
# ../../data/intermediate/tbrucei_read_counts/savage \
# ../../results/figures/multiqc_reports/savage

# bash generate-multiqc-reports.sh \
# ../../results/figures/fastqc_reports/telleria \
# ../../data/scratch/reads-alignment-output/telleria \
# ../../data/intermediate/tbrucei_read_counts/telleria \
# ../../results/figures/multiqc_reports/telleria

# make directory to store the results
mkdir -p ../../results/figures/multiqc_reports/savage/
mkdir -p ../../results/figures/multiqc_reports/telleria/

# fastqc directory
FASTQC_DIR=$1

# Hisat2 directory
HISAT2_DIR=$2

# Htseq directory
HTSEQ_DIR=$3

# Multiqc output directory
OUT_DIR=$4

# run mutiqc
multiqc ${FASTQC_DIR}/*_fastqc.zip ${HISAT2_DIR}/*.txt ${HTSEQ_DIR}/*.counts.txt --outdir ${OUT_DIR}
```

Analysis in R

Importing samples count data into R

For further analysis, samples read counts are read into R. To read the sample counts data into R using the script below, simply type `source(here::here("scripts","analysis","import-read-counts-into-r.R")` on the R console and hit enter.

```
#!/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

# required packages
library(tibble)

# # where are we?
cntdir <- here::here("data", "intermediate", "read_counts")
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
```

```r
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
reads_count <- DT[[myfiles[1]]]

# inspect
#head(reads_count)

# we now add each other table with the ID column as key
for (i in 2:length(myfiles)) {
  y <- DT[[myfiles[i]]]
  z <- merge(reads_count, y, by = c("ID"))
  reads_count <- z
}

# ID column becomes rownames
rownames(reads_count) <- reads_count$ID
reads_count <- reads_count[,-1]

## add total counts per sample
reads_count <- rbind(reads_count,
                                 tot.counts=colSums(reads_count))

# inspect and look at the top row names!
#head(reads_count)

#tail(reads_count)

####################################
# take summary rows to a new table
# ( not starting with Tb and tmp with invert=TRUE )

# transpose table for readability
reads_count_summary <- reads_count[grep("^Tb|^tmp", rownames(reads_count),
                                                   perl=TRUE, invert=TRUE), ]

# review
#reads_count_summary

# transpose table
t(reads_count_summary)

# write summary to file
write.csv(reads_count_summary,
          file = here::here("data", "intermediate","reads_count_summary.csv"),
          row.names = FALSE)

####################################
# take all data rows to a new table
reads_count <- reads_count[grep("^Tb|^tmp", rownames(reads_count), perl=TRUE, invert=FALSE), ]

# inspect final merged table
#head(reads_count, 3)

# write data to files
saveRDS(reads_count, file = here::here("data", "intermediate", "reads_count.RDS"))

reads_count <- rownames_to_column(reads_count,"transcript_id")
write.csv(reads_count,
          file = here::here("data", "intermediate", "reads_count.csv"),
          row.names = FALSE)

# cleanup intermediate objects
rm(y, z, i, DT, reads_count)
```

Sample quality check

The quality of the samples is checked before further analysis to check for outlier and batch effects.

```
# load required packages and data
source(here::here("scripts","analysis","libraries.R"))
samples.metadata.clean <- readRDS(here::here("data","raw","samples.metadata.clean.RDS"))
reads_count <- readRDS(file = here::here("data", "intermediate", "tbrucei_reads_count.RDS"))

# Create a DGEList object
counts <- DGEList(reads_count, group = samples.metadata.clean$Tissue)

# check the number of genes with no expression in all samples
table(rowSums(counts$counts==0)==15)
#FALSE  TRUE
# 9184   792

# Filtering non-expressed and lowly-expressed genes.
keep.exprs <- filterByExpr(counts, group=samples.metadata.clean$Sample_Name)
filtered.counts <- counts[keep.exprs,, keep.lib.sizes=FALSE]

# # change transcript ids to corresponding gene ids ------------------------------------------
gtf_file <- import(here::here("data","scratch","tbrucei","TriTrypDB-43_TbruceiTREU927.gtf"))
gene_and_transcript_id <- mcols(gtf_file)[,c("gene_id","transcript_id")]
gene_and_transcript_id <- unique(gene_and_transcript_id)


# replace transcript ids with gene ids as rownames
filtered.counts.tmp <- tibble::rownames_to_column(as.data.frame(filtered.counts$counts),
                                                  "transcript_id")

filtered.counts.tmp$gene_id <- gene_and_transcript_id$gene_id[match(filtered.counts.tmp$transcript_id,
                                                                  gene_and_transcript_id$transcript_i

filtered.counts.tmp <- as.data.frame(filtered.counts.tmp) %>% remove_rownames %>%
  column_to_rownames(var = "gene_id")

filtered.counts.tmp$transcript_id <- NULL
filtered.counts$counts <- as.matrix(filtered.counts.tmp)

# obtain logCPM unnormalized for plotting purposes ------------------------------------------
# Here, the norm.factors value is 1 for all samples
logcpm.unnorm.counts <- cpm(filtered.counts, log = TRUE, prior.count = 2, normalized.lib.sizes = TRUE)

# Normalize for composition bias using TMM
filtered.counts <- calcNormFactors(filtered.counts, method = 'TMM')

# Convert counts per million per gene to log counts per million for further downstream analysis.
logcpm.norm.counts <- cpm(filtered.counts, log = TRUE, prior.count = 2, normalized.lib.sizes = TRUE)

# use ComBat to remove batch effects
modcombat <- model.matrix(~Tissue, data=samples.metadata.clean)
logcpm.norm.counts.combat <- ComBat(dat=logcpm.norm.counts, batch = samples.metadata.clean$Batch,
                                    mod = modcombat)

# save outputs for later analysis
saveRDS(filtered.counts, here::here("data","intermediate","filtered.counts.RDS"))
saveRDS(logcpm.unnorm.counts, here::here("data","intermediate","logcpm.unnorm.counts.RDS"))
saveRDS(logcpm.norm.counts, here::here("data","intermediate","logcpm.norm.counts.RDS"))
saveRDS(logcpm.norm.counts.combat, here::here("data","intermediate","logcpm.norm.counts.combat.RDS"))
saveRDS(gene_and_transcript_id, here::here("data","intermediate","gene_and_transcript_id.RDS"))

# clean up the environment
rm(gtf_file, filtered.counts.tmp)
```