QC, trimming, alignment, and counts for Heaney et al. 2021

Sanjeev V Namjoshi (snamjoshi87@utexas.edu)

February 23rd, 2017 (revised June 15th, 2019)

There are two different sets of technical replicates reflecting the runs from July 2016 (two replicates) and August 2016 (two replicates). QC and trimming was performed separately on each set of runs. Reads from the runs were combined together for analysis after alignment.

Load packages

```
library(R.utils)
library(Rsubread)
```

Quality Control

Command: `fastqc`

QC was performed with FastQC **(version 0.11.5)** in the terminal. Then, all FASTQ files were moved to a new directory.

```
system("fastqc *.gz")
dir.create("fastqc")
system("mv *.zip *.html fastqc")
```

The QC files show high-quality Q-scores across all sequences. There is minor adaptor contamination. Repetitive sequences are generally below 1% of all sequences in a given library and consist primarily of ribosomal RNA or adaptors. The replicates from the Ro knockout show low library complexity and unusual GC content (compared to the other libraries). This is likely because of the small amount of starting material. We were only able to obtain one replicate for the Ro knockout because the other replicates had too little material to sequence. Overall, the QC shows that the data is of high quality.

Trimming

Command: `trim_galore --fastqc -o trimmed_fastqc *.gz`

Adaptor/quality trimming was performed with **Trim Galore (version 0.4.1)** using **Python 2.7.11**. Trim Galore is wrapped around **cutadapt 1.10** and has many known adaptors pre-loaded so they do not need to be specified in advance. The basic options for trim galore do not perform any aggressive trimming and in general, a high amount of trimming was not needed.

```
dir.create("trimmed_fastqc")
system("trim_galore --fastqc -o trimmed_fastqc *.gz")
```

Alignment

Command: `tophat2 library-type fr-firststrand -G [GTF] -p 4 -o [library][index][files]`

For alignment, we first created a metatable to help organize all the different libraries which aids in downstream processing steps. The following code chunk also creates a new alignment directory:

```
metaTable <- data.frame(LibraryName = c("DMSO1_T1", "DMSO1_T2", "DMSO2_T1", "DMSO2_T2",
                                        "RO_KO2_T1", "RO_KO2_T2",
                                        "RO_RAP_KO1_T1", "RO_RAP_KO1_T2", "RO_RAP_KO2_T1", "RO_RAP_KO2_T
                                        "RO_RAP_WT1_T1", "RO_RAP_WT1_T2", "RO_RAP_WT2_T1", "RO_RAP_WT2_T
                                        "RO_WT1_T1", "RO_WT1_T2", "RO_WT2_T1", "RO_WT2_T2", "RO_WT3_T1"
                                        "SAL_KO1_T1", "SAL_KO1_T2", "SAL_KO2_T1", "SAL_KO2_T2", "SAL_KO
                                        "SAL_WT1_T1", "SAL_WT1_T2", "SAL_WT2_T1", "SAL_WT2_T2", "SAL_WT
                                        "TOT1_T1", "TOT1_T2", "TOT2_T1", "TOT2_T2", "TOT3_T1", "TOT3_T2
                         LibraryLayout = rep("SINGLE", 42),
                         fastq = grep("fq.gz", list.files(getwd()), value = T),
                         condition = c(rep("CTL", 4), rep("KO", 8), rep("WT", 12), rep("KO", 6), rep("WT
                         technical = rep(1:2, 21))

metaTable$LibraryName <- as.factor(metaTable$LibraryName)

dir.create("alignment")
setwd("alignment")
```

Gene model annotations (mm9) were downloaded from Ensembl.org and pre-built reference indices were downloaded from the Illumina website.

```
### Gene model annotation
URL1 <- "ftp://ftp.ensembl.org/pub/release-84/gtf/mus_musculus/Mus_musculus.GRCm38.84.gtf.gz"
download.file(URL1, "Mus_musculus.GRCm38.84.gtf.gz")
gunzip("Mus_musculus.GRCm38.84.gtf.gz")

### Pre-built reference indices
URL2 <- "ftp://igenome:G3nom3s4u@ussd-ftp.illumina.com/Mus_musculus/Ensembl/GRCm38/Mus_musculus_Ensembl_
download.file(URL2, "Mus_musculus_Ensembl_GRCm38.tar.gz")
gunzip("Mus_musculus_Ensembl_GRCm38.tar.gz")
untar("Mus_musculus_Ensembl_GRCm38.tar")
```

To align reads to genome, we used **Tophat 2.1.1**, **Bowtie 2.2.9**, and **samtools 0.1.19**. The `gtf` and `index` variables are strings leading to the directory where these files are stored. The library type was specified as `fr-firststrand` in accordance with the library prep protocol.

```
### Move all trimmed FASTQ data files to the alignment directory
setwd("..")
system("mv *.gz alignment")
setwd("alignment")

gtf <- "/media/sanjeev/MyPassport/Sanjeev/fmrp_ip_analysis/fastq_data/trimmed_fastqc/alignment/Mus_muscu
index <- "/media/sanjeev/MyPassport/Sanjeev/fmrp_ip_analysis/fastq_data/trimmed_fastqc/alignment/Mus_mus

### Create a vector that stores the tophat terminal commands for each file
tophat <- with(metaTable, paste("tophat2 --library-type fr-firststrand -G ", gtf, "-p 4 -o", LibraryNam
```

```
### This loop runs all the tophat commands in the terminal
for(i in 1:length(tophat)) {
  system(tophat[i])
}
```

We created a table from the alignment results file found in each Tophat alignment file. These data can be found in the `full_alignment_summary.csv` file.

```
output = data.frame()

for(i in 1:length(metaTable$LibraryName)) {
  dat <- read.table(paste(metaTable[i,1], "/align_summary.txt", sep = ""), header = FALSE, sep = "\t", i
  input <- substr(dat[2,1],25,nchar(dat[2,1]))
  mapped <- substr(dat[3,1],25,nchar(dat[3,1]))
  multiple <- substr(dat[4,1],26,nchar(dat[4,1]))
  rate <- substr(dat[5,1], 1, nchar(dat[5,1])-1)

  summary <- data.frame(metaTable[i,1], input, mapped, multiple, rate)

  output <- rbind(output, summary)
}

names(output) <- c("Library", "Input_Reads", "Mapped_Reads", "Multiple_Alignments", "Mapping_Rate")

write.csv(output, "full_alignment_summary.csv")   # Save alignment stats to working dir
```

The alignment data shows that most libraries had relative strong alignment percentages (greater than 75%). The only exception to this is the Ro KO replicate 2. As mentioned previously, there was very low starting material for this library and low library complexity. While some of this can be explaining through ribosomal sequence contamination, there are also further sequences in this sample that may be useful in differential gene expression analysis. Thus, we have opted not to remove this library.

Calculate counts from BAM files

We used **Rsubread 1.20.6** to calculate counts. The `strandSpecific = 2` option was chosen in accordance with the library prep protocol. We used the `countMultiMappingReads = TRUE` option specifically for the purpose of counting ambiguously mapped reads. At this point, all BAM files from both runs (July 2016 and August 2016) were combined into one directory so that the the final output includes counts for four technical replicates.

```
### Create an extry column for counts and a new vector that contains the directory for all the BAM files
metaTable$Counts <- paste(metaTable$LibraryName, "count", sep = ".")
alignmentFilesDir <- file.path(getwd(), paste0(metaTable$LibraryName, "/", "accepted_hits.bam"))

path <- "/media/sanjeev/MyPassport/Sanjeev/fmrp_ip_analysis/fastq_data/trimmed_fastqc/alignment"
gtfDir <- file.path(path, "Mus_musculus.GRCm38.84.gtf")

### Run Rsubread on all BAM files
for(i in 1:length(metaTable$LibraryName)) {
  assign(paste0(metaTable[i,1],"_Count"), featureCounts(alignmentFilesDir[i],
                                                        annot.ext = gtfDir,
                                                        isGTFAnnotationFile = T,
```

```
                                                    isPairedEnd = FALSE,
                                                    countMultiMappingReads = TRUE,
                                                    strandSpecific = 2,
                                                    nthreads = 4))
}

### Bind columns together into one dataframe and export
listObjects <- lapply(grep("*Count",ls(), value = T), get)
countData <- as.data.frame(lapply(listObjects, function(x) x$counts))
names(countData) <- metaTable$LibraryName
write.csv(countData, "countData.csv")
```

Session info:

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] backports_1.0.4 magrittr_1.5    rprojroot_1.1   tools_3.3.2
##  [5] htmltools_0.3.5 yaml_2.1.14     Rcpp_0.12.8     stringi_1.1.2
##  [9] rmarkdown_1.2   knitr_1.15.1    stringr_1.1.0   digest_0.6.10
## [13] evaluate_0.10
```