# RNA-Seq Analysis Pipeline

ApoorvaBabu

4/10/2021

## Contents



The following pipeline analyses RNA-Seq data from the bamfiles (aligned using STAR) and makes it compatible to our Shiny App, NGSViewer.

### Load packages

The required libraries can be installed from CRAN and/or Bioconductor. Only exception is the ExpressExtras package which has all the helper functions. ExpressExtras can be installed from our github page

```
#Load required libraries
library(Rsubread)
require(devtools)
library(ExpressExtras)
library(plyr)
library(dplyr)
library(tidyr)
library(limma)
library(NMF)
library(RColorBrewer)
library(edgeR)
library(ggplot2)
require(SPIA)
```

## Getting Count data from aligned bamfiles

The fastq files are first aligned using the STAR aligner. The count matrix is generated in R using the RSubread package

```
#Specify location of gtffile
gtffile='hg19_data/Homo_sapiens.GRCh37.75.gtf'

#specify location of the bamfiles
bamfiles <- list.files('STAR',pattern=".bam$",full.names=TRUE)

#Run featurecounts function to get count information
cts.dedup <- featureCounts(files=bamfiles,annot.ext=gtffile,strandSpecific=0,ignoreDup=T,isPairedEnd = T

#save the mapping statistics and extract out count matrix
stats= cts.dedup$stat
cts=cts.dedup$counts

#Remove the .bam from colnames
colnames(cts)=gsub(".bam","",colnames(cts))

write.csv(stats,file="Subread_stats.csv",row.names = F)

cts <- cts[order(rownames(cts)),]
cts <- cts[,order(colnames(cts))]
```

## Specifying input parameters

Note : If contrastmaker is set to 'auto', the contrasts for differential expression are auto-generated. To manually specify contrasts, set contrastmaker to 'file' and include a csv file in the path. For example csv file, click here

```
dir='STAR'
projectname='exampleproject'
librarytype='unstranded'
constrastmaker='auto' #set to either file or auto
```

## Prepare phenoData

Prepare sample information file using this example and read it in. Make sure the samplenames match the filenames of your bamfiles. If not rename the count matrix to reflect the same sample names

```
#read phenodata
pData<- read.csv('data/phenodata.csv') %>% arrange(sample_name)
rownames(pData)=pData$sample_name
phenoData <- new("AnnotatedDataFrame", data=pData)
constrastmaker='auto' #set to either file or auto
```

## Annotate genes from count matrix

Note : You will require the ExpressExtras package to run this function

```
# Get a data.frame of gene annotations
genenames <- GeneAnnotate(as.character(rownames(cts)),organism = "human")
rownames(genenames)=genenames$ENSEMBL
```

## Filter counts and Normalize data

```
# Create a DGEList object from count matrix and filter CPM
dge <- DGEList(counts=cts[rownames(cts) %in% genenames$ENSEMBL,], genes=genenames)

#Filter low expressed genes. Genes with cpm<1 in 25% of samples are filtered out
cpms = cpm(dge)
keep = rowSums(cpms>1)>=.25*dim(dge)[2]
dge <- dge[keep,]
genenames = genenames[keep,]

# Normalize read counts using TMM normalization
dge <- calcNormFactors(dge)
plotMDS(dge)
```

```
boxplot(log2(cts))
```

```
rownames(genenames)=genenames$ENSEMBL
```

## Voom transform data

```
#Create design matrix
(design <-model.matrix(~0+maineffect,data=pData))
```

```
##      maineffectmut maineffectwt
## Mut1             1            0
## Mut2             1            0
## Mut3             1            0
## WT1              0            1
## WT2              0            1
## WT3              0            1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$maineffect
## [1] "contr.treatment"
```

```
#Clean up the colnames of the design matrix, don't need the colname in.
colnames(design)<-gsub('maineffect','',colnames(design))

# VOOM transform the data
v <- voom(dge,design,plot=TRUE)
```

```
plotMDS(v)
```

```
#Save the lowest expression value in the phenodata
pData$minexpr=abs(min(v$E))+1
v$E <- v$E + abs(min(v$E))+1
```

## Save the data as an Expression set

The data has to be saved in the eset format to be properly read in by NGSViewer

```
# Save sample information as phenoData
rownames(pData) <- (pData$sample_name)
if(all(rownames(pData)!=colnames(v$E))){
  stop('Sample names not equal')
}
phenoData <- new("AnnotatedDataFrame",data=pData)

# Save gene information as featureData
fData <- new("AnnotatedDataFrame",
             data=genenames)
all(rownames(genenames)==rownames(v$E))
```

```
## [1] TRUE
```

```
#Save expression, sample and feature data into eset
eset<- ExpressionSet(assayData=v$E,phenoData=phenoData,featureData=fData,annotation=unique(as.character
```

## Apply limma pipeline for differential expression

```
# Create contrast matrix and fit models
if(constrastmaker=='auto'){
  f<-as.vector(unlist(combn(colnames(design),2,function(x)paste(x,collapse="-"))))
}else{
  f<-read.csv('data/contrastlist.csv') %>%
    mutate(c=paste(treatment,control,sep="-")) %>%
    .$c
}
(contrast.matrix <- makeContrasts(contrasts = f,levels=design))
```

```
##       Contrasts
## Levels mut-wt
##    mut       1
##    wt       -1
```

```
#limma pipeline for differential expression
fit <- lmFit(v,design)
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2,robust = T)
```

### Load datasets for Gene set testing using Camera

```
# load and prepare all the genesets from Molecular Signature Database for camera

hum=c("human","Human","Hs","Homo sapiens")
mouse=c("mouse","Mouse","Mm","Mus musculus")
if(unique(pData$organism) %in% hum){
  data('human_H_v5',package="ExpressExtras")
  h.indices <- ids2indices(Hs.H,genenames$ENTREZID)
  data('human_c2_v5',package="ExpressExtras")
  c2.indices <- ids2indices(Hs.c2,genenames$ENTREZID)
  data('human_c3_v5',package="ExpressExtras")
  c3.indices <- ids2indices(Hs.c3,genenames$ENTREZID)
  data('human_c5_v5',package="ExpressExtras")
  GO.indices <- ids2indices(Hs.c4,genenames$ENTREZID)
}else if(unique(pData$organism) %in% mouse){
  data('mouse_H_v5',package="ExpressExtras")
  h.indices <- ids2indices(Mm.H,genenames$ENTREZID)
  data('mouse_c2_v5',package="ExpressExtras")
  c2.indices <- ids2indices(Mm.c2,genenames$ENTREZID)
  data('mouse_c3_v5',package="ExpressExtras")
  c3.indices <- ids2indices(Mm.c3,genenames$ENTREZID)
  data('mouse_c4_v5',package="ExpressExtras")
  c4.indices <- ids2indices(Mm.c4,genenames$ENTREZID)
  data('mouse_GO',package="ExpressExtras")
  GO.indices <- ids2indices(Mm.GO,genenames$ENTREZID)
}else {
  print("Incorrect organism name. ")
}
```

### Run limma, camera, topGO and SPIA for each contrast in a loop

```
#Remove the '-' from the contrast name or it will cause issues downstream
(contrastnames <-gsub('-','_vs_',colnames(contrast.matrix)))
```

```
## [1] "mut_vs_wt"
```

```
#Create lists to save the results for limma,togo, SPIA and camera for all contrasts
limma <-  vector(mode="list", length=length(contrastnames))
names(limma) <- contrastnames
camera <- vector(mode="list", length=length(contrastnames))
names(camera) <- contrastnames
topgo <-vector(mode="list", length=length(contrastnames))
names(topgo) <- contrastnames
spia<-  vector(mode="list", length=length(contrastnames))
names(spia) <- contrastnames

#Loop over all contrasts and run limma, camera, topgo and spia for each one and save each result as a l

for(i in 1:length(contrastnames)){
```

```
  print(contrastnames[i])
  limma[[contrastnames[i]]] <- Cleanup2(topTable(fit2,coef=i,n=Inf,p.value=1))
  topgo[[contrastnames[i]]] <- runTopGO(limma[[contrastnames[i]]],organism ="human")

  k=limma[[contrastnames[i]]]

  #for each limma data (corresponding to the contrast), run SPIA
  limma_sel <- k[which(abs(k$fc) > 2 & k$adj.P.Val < 0.05),]
  if(nrow(limma_sel)>0){
    all_genes = as.numeric(k$ENTREZID)
    sig_genes = limma_sel$fc
    names(sig_genes) = limma_sel$ENTREZID
    sig_genes = sig_genes[complete.cases(names(sig_genes))]
    sig_genes = sig_genes[unique(names(sig_genes))]
    spia[[contrastnames[i]]] <- spia(de=sig_genes, all=all_genes, organism=ifelse(unique(pData$organism
  }else{
    spia[[contrastnames[i]]] <- data.frame()
  }

  #run camera
  res.h <- camera(v, h.indices, design,contrast.matrix[,i],inter.gene.cor=0.01)
  res.c2 <- camera(v, c2.indices, design,contrast.matrix[,i],inter.gene.cor=0.01)
  res.GO <- camera(v, GO.indices, design,contrast.matrix[,i],inter.gene.cor=0.01)
  camera[[contrastnames[i]]] <- list(Hallmark=list(camera_result=res.h,indices=h.indices),Curated=list(
}
```

```
## [1] "mut_vs_wt"
```

```
# Save list of results as and RData file
results <- list(eset=eset,limma=limma,camera=camera, topgo=topgo,spia=spia)
save(results,file=paste(projectname, ".RData",sep=''))
```