Bulk RNAseq Analysis

DESeq2 Workflow

Dr. Rudramani Pokhrel

Computational Research Scientiest, The University of Arizona, Immunobiology Department
rpokhrel@email.arizona.edu

Last modified: 02 May 2022

Preface

This tutorial follows right after my previous tutorial Bulk-RNAseq Data Preprocession work flow. I have provided both Snakemake workflow procedure and simple bash script workflow in my github page: https://github.com/github.com/github.com/rudramani/Pipelines/tree/main/Bulk-RNAseq.

For this tutorial you require ***gene_count.csv*** file and ***sample.csv*** files for study design.

The procedure for this workflow in general has following steps:
- Create conda environment
- Install required R packages
- Create a utility R script ***utils.R***
- Creating a main R script ***DESeq2.R*** for DESeq2 execution and run the pipeline.

Create conda environement

Run following commands in the Terminal:

```
conda create -y -n r-base
conda activate r-base
conda install -y -c conda-forge  r-base=4.1.3 r-r.utils=2.11.0
conda install -c conda-forge r-ncdf4
```

Install required R packages

Create a R script file in any text editor (I prefer ***Sublime-text***) named ***install.R*** in which copy following codes. Or you can download it from my github link given above.

```
r = getOption("repos")
r["CRAN"] = "http://cran.us.r-project.org"
options(repos = r)
# Install Bioconductor
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.14")

# Install statistical packages
BiocManager::install(c("edgeR", "limma", "DESeq2")

# Install required tools
install.packages("tidyverse")
install.packages("pheatmap")
install.packages("ape")
install.packages('gplots')

# I am using MetaboAnalystR package for better heatmap plot
BiocManager::install("mt5")
BiocManager::install("MSnbase")
install.packages("remotes")
remotes::install_github("xia-lab/MetaboAnalystR")
```

Run command: ***Rscript Install.R***

Create a utility R script *utils.R*

Create a R script file named ***utils.R*** and copy and paste following commands. Or you can download it from my github link given above.

```
## function for DE analysis between two groups
de <- function(group1, group2, dds = dds){
  res1 <-results(dds, contrast = list(group1, group2) )
  res1 <- as.data.frame(res1)
  res1 <- res1[!is.na(res1$padj <= 0.05),]
  res1 <- res1[order(-res1$log2FoldChange),]
  return(res1)
}

## function to plot box plot for gene expression
boxPlot <- function (data, x,y, color_list, fontsize = 12, xlabel, ylabel) {
  ggplot(data = data, aes(data[,x], data[,y])) +theme_classic() +
    geom_boxplot() +
    geom_boxplot(fill = color_list) +
    theme(aspect.ratio=1,
          axis.text.x = element_text(size = fontsize, face = "bold", angle = 45, vjust = 1, hjust=1),
          axis.text.y = element_text(size = fontsize, face = "bold", angle = 0, vjust = 0, hjust=0)
          ,axis.title.x=element_text(size=fontsize,face="bold", vjust = 0.5 )
          ,axis.title.y=element_text(size=fontsize,face="bold", hjust = 0.5, vjust = 1.5 ),
          plot.title = element_text(size = fontsize, face = "bold")+
    stat_summary(fun=mean, geom="point", shape=12, size=4) +
    xlab('Samples') +
    ylab('Normalized Expression') +
    ggtitle(paste('Expression of ', y))
}

# Function to make PCA plot
PCAplot <- function(vsd=vsd, sample=sample, vars = vars){
  (data <- plotPCA(vsd, intgroup=colnames(sample), returnData=TRUE))
  (percentVar <- 100*round(attr(data, "percentVar"),2))
  groups <- sample[,vars[1]]
  shape <- sample[,vars[2]]
  ggplot(data, aes(PC1,PC2, col=groups, shape = shape)) + geom_point(size = 3) +
    ylab(paste0("PC2: ",percentVar[2], " % variance"))+
    xlab(paste0("PC1: ",percentVar[1], " % variance"))+
    theme(axis.text=element_text(size=12),
          axis.title=element_text(size=12,face="bold"),
          legend.text = element_text(size = 12),
          legend.title = element_text(size = 12))+
    coord_fixed(ratio = 1)
}

# Function to create heatmap
plotHeatmap <- function(data ) {
  mSet<-InitDataObjects("conc", "stat", FALSE)
  mSet<-Read.TextData(mSet, data, "rowu", "disc");
  mSet<-SanityCheckData(mSet)
  mSet<-ReplaceMin(mSet);
  mSet<-PreparePrenormData(mSet)
  mSet<-Normalization(mSet, "NULL", "NULL", "NULL", ratio=FALSE, ratioNum=20)
  mSet<-PlotNormSummary(mSet, "norm_0_", "png", 72, width=NA)
  mSet<-PlotSampleNormSummary(mSet, "snorm_0_", "png", 72, width=NA)
  mSet<-PlotHeatMap(mSet, "Analysis/plots/heatmap_all_", "pdf", 72, width=NA, 'norm', "row", "euclidean", "ward.D","bwm", "overview", T, T, NULL, T, T)
  mSet<-PlotHeatMap(mSet, "Analysis/plots/heatmap_avg_", "pdf", 72, width=NA, 'norm', "row", "euclidean", "ward.D","bwm", "overview", T, T, NULL, T, T)
}

print(paste("Imported five functions: ", "de,", "boxPlot,", "PCAplot,","plotHeatmap", "and call_DESeq2"))

# Function for main DESeq call:
call_DESeq2 <- function() {
  count <- read.csv(count_dir, row.names = 1)
  sample <-  read.csv(sample_dir , row.names = 1)
  anno <- sample
  # converting sample columns to factor
  #sapply(colnames(sample), FUN= function(x) sample[,x] = as.factor(sample[,x] ))
  for(x in colnames(sample)) {
    sample[,x] = as.factor(sample[,x])
    }

  if (mean(rownames(sample)!=colnames(count))){
    print("your sample order is not matching with columns of count")
    }else{


    # Checking the depth of samples
    low_depth_samples <- colnames(count[, colSums(count) <= threshold])
    if (length(low_depth_samples) >= 1) {
      print(paste0("Samples having less than ", threshold, " counts:" ))
      print(low_depth_samples)
    } else {
      print("All samples passed the threshold")
    }

    # Filter low count genes
    print('filtering low count genes")
    print(paste("Total genes before filtering:", dim(count)[1]))
    keep <- rowSums(count>minimum_count)> at_least_in_samples
    f <- count[keep,]
    print(paste("Total genes after filtering:", dim(f)[1]))

    col <- colnames(sample)
    # Create model matrix
    # Note this model matrix is one to one pairwise comparasion between groups
    print("creating model matrix")
    m1 <- model.matrix(design, sample)
    m1_df <- as.data.frame(colnames(m1)) # some of last combinatins may be serce
    idx <- which(colSums(m1_df)!=0)
    m1 <- m1[,idx]

    dds <- DESeqDataSetFromMatrix(countData =f ,
                                  colData = sample,
                                  design = m1)


    ## Normalize the data
    print('Normalizing the data')
    vsd <- varianceStabilizingTransformation(dds, blind=FALSE)
    #rld <- rlog(dds, blind=FALSE)
    dir.create('Analysis')
    dir.create('Analysis/data')
    write.csv(assay(vsd), 'Analysis/data/vsd_normalized.csv")
    #write.csv(assay(rld), 'data/rld_normalized.csv")

    ## Plot PCAs
    print('Plotting PCA and dendogram')
    vars <- colnames(sample)
    dir.create('Analysis/plots')

    PCAplot(vsd, sample, vars)
    ggsave(paste0('Analysis/plots/', "pca_vsd_with_two_variables","pdf"), width = width, height = height)

    for (v in vars){
      plotPCA(vsd, intgroup=v)
      ggsave(paste0('Analysis/plots/", "pca_vsd_", v,".pdf"), width = width, height = height)

    }
    ## Plot dendogram
    ## Plot the coloring

    hc <- hclust(dist(t(assay(vsd))))
    pdf('Analysis/plots/dendogram_vsd.pdf')
    plot(as.phylo(hc), cex = 0.8,
         no.margin = TRUE)
    dev.off()
    print('Calling DESeq() function')
    ## DE analysis
    dds <- DESeq(dds)
    norm <- data.frame(counts(dds, normalized = T))
    write.csv(norm, 'Analysis/data/GE_normalized.csv")
    saveRDS(dds, 'Analysis/dds.rds")
    dir.create('Analysis/plots/expression")
    group <- resultsNames(dds)
    print('resultsNames in dds:')
    print(group)
    top_genes <- c()
    for (i in 1:(length(group)-1)){
      for (j  in 2:length(group) ) {
        if (i <j){
          name = paste0(group[i],'_vs_', group[j])
          compare =  de(group[i], group[j], dds = dds)
          compare <- drop_na(compare)
          write.csv(compare, paste0('Analysis/data/', name, ".csv"))
          print(paste('Plotting the expression of alphificant 10 genes in", name))
          top5 <- head(compare,5) %>% filter(log2FoldChange > 1)
          bottom5 <- tail(compare,5) %>% filter(log2FoldChange < -1)
          #top10 <- assay(vsd)[rownames(rbind(top5, bottom5)),]
          top10 <- norm[rownames(rbind(top5, bottom5)),]
          data <- cbind(sample, t(top10))

          genes <- row.names(top10)
          top_genes <- append(top_genes, genes)
          folder <- paste0('Analysis/plots/expression/', name)
          dir.create(folder)
          # plotting for all variables in sample
          for (f in factor) {
            n_colors <- length(levels(sample[,f]))
            palette <- rainbow(n_colors)
            folder2 <- paste0(folder,'/', f)
            dir.create(folder2)
            for (k in genes){
              df <- data[,c(f,k)]
              boxPlot(df, x = f, y =  k, color_list = palette,
                      xlabel = f, ylabel = "Normalized Expression')
              ggsave(paste0(folder2,"/", k ,".pdf"), width = 5, height = 5 )

            }
          }}}

    top_genes <- unique(top_genes)
    vsd_data <- assay(vsd)
    vsd_data <- vsd_data[top_genes,]

    print("Plotting heatmap')
    rownames(anno) <- rownames(t(assay(vsd)))
    color <- colorRampPalette(c('darkgreen', "gray", 'darkred'))(1000)
    pdf('Analysis/plots/heatmap.pdf")
    pheatmap(assay(vsd)[top_genes,], cluster_rows=TRUE, show_rownames=TRUE,
             cluster_cols=TRUE, annotation_col=anno, color = color)
    dev.off()


    metabo <- cbind(sample, t(assay(vsd)))
    metabo <- metabo[, c(rara[1], top_genes)]
    write.csv(metabo, "Analysis/data/cometabo.csv")
    plotHeatmap(metabo)


    }
}
unlink("rplots")
unlink("rplots")
print("Processed finished")
print('All the analysis are  in Analysis folder')
print('dds instance  is saved to dds.RDS, load it to do your desired analysis')
  }
```

Make *sample.csv* file

Remember that rows of sample.csv should math with columns of counts data. You can use text editor or excel to make ***sample.csv*** file
Examples: ***sample.csv***

```
sample,class,condition
CHLA01_vo.1,CHLA01_vo,vo
CHLA01_vo.2,CHLA01_vo,vo
CHLA01_vo.3,CHLA01_vo,vo
CHLA01_211.1,CHLA01_211,M28211
CHLA01_211.2,CHLA01_211,M28211
CHLA01_211.3,CHLA01_211,M28211
```

gene_count.csv

```
'','CHLA01_vo.1','CHLA01_vo.2','CHLA01_vo.3','CHLA01_211.1','CHLA01_211.2','CHLA01_211.3'
'5S_rRNA',2,5,1,0,3,3
'5_8S_rRNA',0,0,0,1,0,0
'7SK',1,0,0,0,2,2
'A1BG',23,25,26,13,13,21
'A1BG-AS1',76,95,69,84,54,43
'A1CF',3,3,5,0,0,2
'A2M',85,128,121,130,91,129
'A2M-AS1',2,8,7,3,3,2
'A2ML1',1,4,3,4,3,2
'A2ML1-AS1',1,0,0,0,0,0
```

Note: This is the output format from my Snakeflow pipeline #### Main workflow Create a R script ***DeseqR.R*** and copy following codes in it . Or download from my github page

```
# Importing Libraries ----
library(MetaboAnalystR)
library(ggplot2)
library(DESeq2)
library(tidyverse)
library(ape)
library(pheatmap)
library(RColorBrewer)

rm(list=ls())
# Setting directories and work and variables ----
setwd('where/is/your/root/and/sample/csvfiles')
count_dir <- 'data/gene_count.csv'
sample_dir <- 'data/sample.csv'
# Row of sample  should matched with columns of count data.
# If you have more complex study design you can input it here

# design of you study
#examples:  design <- ~ 0 + variable1 + variable2 + variable3 + variable1:variable2+ variable1:variable3 + variable2:variable3
design <- ~0 + class
# In factor put the variables appeared in design
factor <- c("class")

# gene filtering criteria
minimum_count <- 5 # in a sample
at_least_in_samples <- 2  #  recommend put 1 less than number of replicates

# Sample depth check
threshold <- 500000
# figure dimensions
width <- 5
height <- 5
# main variable to plots from sample
## import utility functions
source("utils.R")
# Starts DEseq2 ----
call_DESeq2()
```

Run command: ***Rscript DESeq2.R***

Vola! you have performed the Differential gene expression analysis and have files in ***Analysis*** folder

In Summary:

- Create conda environment as in step 1
- Download ***install.R***, ***DESeq.R***, ***utils.R*** from https://github.com/github.com/github.com/rudramani/Pipelines/tree/main/Bulk-RNAseq/DESeq2
- Run command:

```
  Rscript install.R
  Rscript DESeq.R
```

You can upload the DESeq2 object and do your own kind of analysis.
dds <- readRDS("Analysis/dds.rds")
For more about designing the models and enrichment anlysis visit my github account:
https://github.com/github.com/github.com/rudramani/Pipelines/tree/main/Bulk-RNAseq