RNA Sequencing Differential Analysis Project

Gavin Fortenberry

August 15th, 2018

Contents

Introduction

What is Myleofibrosis?

- Myleofibrosis is a type of bone marrow cancer in which an rapidly increasing number of blood forming cells form a fibrous like structure that sometimes leads to acute leukemia. Certain genotypes like the JAK2 V617F mutation have been a determining factor of Blast Transformation in Myleofibrosis.

What is differential gene expression?

- Differential gene expression is a way to analyze factors in different groups that may or may not be associated with different gene counts, from RNA sequence data.

What is gene expression?

- Gene expression is the process of which information inside a gene is used to make RNA and proteins. The genotype leads to the phenotype.

Why is differential gene expression important to a biological question?

How to do differential gene expression:

What inputs go in:

24 patients, 12 MF, 12 normal ### What is the analysis going to do

What are the outputs/what is the meaning of the outputs

(My) Experimental Design

What groups am I going to compare? (Bio factors vs. tech factors)

- The groups of phenotype data being used can be sorted into two overarching categories: biological factors and technical factors. Biological factors include "Tissue_type", "genotype_jak2", "genotype_calr", and "genotype_mpl". Technical factors include: "collection_type", "time_to_processing", and "extraction_type".
- The reason why I am splitting the data/analysis into two categories is to attempt to see whether certain factors from each category have an impact on specific gene counts or not.
- To be completed: Why im picking each of the columns within those sections
- I will separate demographic groups by phenotype, and compare gene expression counts for each of the genes between the two groups, groups defined by conditions of that factor. For example, the conditions of the factor of age could be different age groups, sex could be male or female, genotype could be present or not present, etc. ultimate goal is to make observations/conclusions about possible differences in gene counts between different groups that may or may not be significant in diagnosis of Myleofibrosis.

Data Import

Install R Packages

Installs packages of functions that can be used to help manipulate variables and data libraries. These packages can be installed through base R code that downloads them and installs them into R studio. Then certain programming phrases can be used to do new functions in connection with base R programming phrases, with an indicator of the package being used mentioned before using that package's functions. For example, if I wanted to use a function of the "dplyr" package, I would write -> "dplyr::(insert function here)(insert arguments of functions here)". Note that functions or phrases that are in base R programming vocabulary do not require a name indicator of the package being used, as it is from the base software and the software knows how to interperet those phrases without an idicator of a package. It is still required to write names of functions from Base R that are wanting to be used though, and if shortcut variables for functions are desired then you can assign new variables to those functions.

Bioconductor packages need to be installed by "biocLite" rather than install.packages which is for CRAN/base R

```
install.packages("dplyr")
install.packages("knitr")
install.packages("rmarkdown")

### Install Bioconductor, the DESeq2 Package and ggplot2
source("https://bioconductor.org/biocLite.R")
biocLite("DESeq2")
install.packages("ggplot2")
```

Load Additional Packages

```
library(DESeq2); library(ggplot2)
library(plyr); library(dplyr)
library(gghighlight)
```

Read in Project Metadata to R

Reads the two CSV's and combines them into one data frame: The code below creates three data frames: one labeled as "df_pheno" for data of a file called phenotable, which contains demographics and general information from patients and samples taken from them, another labeled as "df_molecular" for data from another file called molecularDataSets, which contains more specific info on samples collected from patients and their diagnoses from these samples, and another labeled as "df_combined" that combines the two data frames according to common columns.

The overall purpose of this code chunk is to create the combined data frame.

The first two data frames created ("df_pheno" and "df_molecular") were created using base R code "read.csv" function, which reads in data into a data frame from a CSV, a comma seperated values document.

In the process of formatting the df_pheno dataframe in order to be compatible when joining with the df_pheno dataframe, the "rename" function from the dplyr package is used on the df_pheno dataframe to change the name of a mismatching column name.

The function used to actually combine the two dataframes into one is the "full_join" function from the dplyr package which joins two indicated dataframes together by a common column.

"str()" is short for structure, and is a base R function that allows the user to get an idea of the format of the data they are looking at

```
df_pheno <- read.csv(file="../RNASeqData/phenotypeTable.csv",
                     header =TRUE)
df_pheno <- df_pheno %>% rename(assay_material_id = ends_with("assay_material_id"))

df_molecular <- read.csv(file="../RNASeqData/molecularDataSets.csv",
                         header =TRUE, sep=",")

df_combined <- dplyr::full_join(df_pheno, df_molecular,
                                by = "assay_material_id")
head(df_combined, n = 3)
```

```
##   assay_material_id     sex    race age_range diagnosis collection_event
## 1             R0297 unknown unknown   unknown        mf        diagnosis
## 2             R0299    male unknown elderly61        mf        diagnosis
## 3             R0298 unknown unknown   unknown        mf        diagnosis
```

```
##   acquisition_date     tissue_type tissue_type_origin collection_type
## 1          8/14/13 peripheralblood                 NA            edta
## 2          8/22/13 peripheralblood                 NA            edta
## 3          8/19/13 peripheralblood                 NA            edta
##    processing_type   time_to_processing       storage shipping
## 1 mononuclearcells repositoryprocessing cryopreserved    fresh
## 2 mononuclearcells repositoryprocessing cryopreserved    fresh
## 3 mononuclearcells repositoryprocessing cryopreserved    fresh
##   genotype_jak2 genotype_calr  genotype_mpl material_type extraction_type
## 1      positive      negative notdetermined           RNA          column
## 2      negative      positive notdetermined           RNA          column
## 3      negative      positive notdetermined           RNA          column
##   na_260280 na_260230   rin_range jak2_vaf molecular_id genomics_types
## 1      2.05      1.86 highquality       70    M00000298         rnaseq
## 2      2.05      1.81 highquality        0    M00000300         rnaseq
## 3      2.07      1.78 highquality        0    M00000299         rnaseq
##   omics_sample_name omics_contact_id omics_date
## 1        JAK2-6-1-D          jradich    2/27/14
## 2       JAK2-10-1-D          jradich    2/27/14
## 3           MF-D-07          jradich    3/24/14
##                seq_flowcell_id seq_readlength seq_paired seq_libtype
## 1 140227_SN367_0370_AH8JPDADXX             99        yes      truseq
## 2 140227_SN367_0370_AH8JPDADXX             99        yes      truseq
## 3 140324_SN367_0381_BH929TADXX             99        yes      truseq
```

Save Prepared Metadata

Writes a CSV (a comma seperated values document) from the new combined data frame(df_combined) of dataframes df_pheno and df_molecular into a document called "combineddata.csv" stored in the working directory (the main location of the files created from the Rstudio

```
write.csv(df_combined, file = "../ProjectFiles/combineddata.csv",
          row.names = FALSE)
```

The read.csv function is used to re-read the newly written CSV to make sure the data is the same as it was when written. The "str()" and "summary" functions are used to compare the statistics of the new dataframe to the original created one to confirm similarity.

```
test_set_combined_data <- read.csv("combineddata.csv")
str(test_set_combined_data)
```

Read in RNA Sequencing Data from HiSat2/htseqcount

Makes a list of directories(folders) with the R base function "list.dirs" which takes the indicated path of the main directory containing the directories its making a list of in, and the econd argument written "recursive" is set to false becuse the main directory is not desired to be listed in the list of its components.

```
#":/Users/gfortenb/Documents/GitHub/bioDS-bootcamp/RNASeqData"
RNADirectoryList = list.dirs(path = "../RNASeqData", recursive = FALSE)
```

Makes a List of files within each folder of each directory in the main directory. Uses a function to go through the files within each folder and only list files with a certain phrase in the name of the file, "htseq.txt", using the pattern function (only argument usedin function is name of character phrase its looking for).

Binds path's of files(locations of the files in the computer) on the list made to the molecular ID of the files with "as.data.frame"" function"/"cbind" function, created by finding key character sequences in the title of the folders containing the "htseq.txt"" files, using the gsub function (from base R). The first argument used in the gsub function is a character phrase that indicates where in the string to look for the character phrase of the molecular ID, and the second argument is the list of data of file locations/names to look for the ID in.

The "colnames" function (base R) sets the column names if the newly created dataframe.

```
FileList_htseq1 = sapply(RNADirectoryList,
                  function(x){list.files(path = x,
                                         full.names = TRUE,
                                         pattern = "htseq.txt") })

htseq_FileID_df <- as.data.frame(cbind(FileList_htseq1,
                                 gsub("^.*-","", RNADirectoryList)),
                            stringsAsFactors = F)
colnames(htseq_FileID_df) <- c("Path", "molecular_id")
head(htseq_FileID_df, n = 3)
```

```
##
## ../RNASeqData/JAK2-10-1-D-R0299-M00000300 ../RNASeqData/JAK2-10-1-D-R0299-M00000300/JAK2-10-1-D.htseq
## ../RNASeqData/JAK2-30-D-R0301-M00000302       ../RNASeqData/JAK2-30-D-R0301-M00000302/JAK2-30-D.htseq
## ../RNASeqData/JAK2-36-D-R0303-M00000304       ../RNASeqData/JAK2-36-D-R0303-M00000304/JAK2-36-D.htseq
##                                           molecular_id
## ../RNASeqData/JAK2-10-1-D-R0299-M00000300    M00000300
## ../RNASeqData/JAK2-30-D-R0301-M00000302      M00000302
## ../RNASeqData/JAK2-36-D-R0303-M00000304      M00000304
```

Reads all data from list of selected files and compiles into different data frames/list of different data frames using LApply function.

```
listOf_alldf <- lapply(seq(1:nrow(htseq_FileID_df)),
                  function(i){
                    X <- read.delim(file = htseq_FileID_df$Path[i],
                                    header = FALSE);
                  colnames(X) <- c("Gene", htseq_FileID_df$molecular_id[i]);
                  return(X)
                  } )
```

SALMON HERE<<<—

```
### Reads in data/creates salmon counts dataframe
salmonCounts_df <-read.csv(file="../RNASeqData/2018-08-06_SalmonGeneLevelCounts.csv",
                      header =TRUE)
```

Data Summarizing

Summarizing PHENOTYPES/defining experimental design

Biological factors

```
bio_factors_summary <- df_pheno %>% group_by(diagnosis, genotype_jak2, genotype_calr, age_range, sex) %
head(bio_factors_summary, n = 3)
```

```
## # A tibble: 3 x 6
## # Groups:   diagnosis, genotype_jak2, genotype_calr, age_range [3]
##   diagnosis genotype_jak2 genotype_calr age_range sex     `n()`
##   <fct>     <fct>         <fct>         <fct>     <fct>   <int>
## 1 mf        negative      negative      unknown   unknown     1
## 2 mf        negative      positive      elderly61 male        1
## 3 mf        negative      positive      unknown   unknown     5
```

Technological factors

```
tech_factors_summary <- df_pheno %>% group_by(time_to_processing, collection_type, collection_event, ex
head (tech_factors_summary, n = 1)
```

```
## # A tibble: 1 x 5
## # Groups:   time_to_processing, collection_type, collection_event [1]
##   time_to_processi~ collection_type collection_event extraction_type `n()`
##   <fct>             <fct>           <fct>            <fct>           <int>
## 1 repositoryproces~ edta            diagnosis        column              4
```

Bio & tech factors

```
bio_and_tech_summary <- df_pheno %>% group_by(diagnosis, time_to_processing, genotype_jak2, collection_
head(bio_and_tech_summary, n = 3)
```

```
## # A tibble: 3 x 10
## # Groups:   diagnosis, time_to_processing, genotype_jak2, collection_type,
## #   genotype_calr, collection_event, age_range, extraction_type [3]
##   diagnosis time_to_process~ genotype_jak2 collection_type genotype_calr
##   <fct>     <fct>            <fct>         <fct>           <fct>
## 1 mf        repositoryproce~ negative      edta            positive
## 2 mf        repositoryproce~ negative      edta            positive
## 3 mf        repositoryproce~ negative      unknown         negative
## # ... with 5 more variables: collection_event <fct>, age_range <fct>,
## #   extraction_type <fct>, sex <fct>, `n()` <int>
```

Create a Summarized Experiment Data set

("Summarized Experiment" - something specific to DESeq2 package) - Normalize RNA Sequencing Counts - in new normalized data frame - (Normalize each sample's counts data based on over all library size for each sample.)

Data Setup For Analysis

Differential Gene Expression Analysis with DESeq2

HERE DOWN->

Make the list of dataframes into one dataframe w/all contents of each dataframe in the dataframe of dataframes as columns using the "join_all" function from the plyr package. This produces a dataframe with the molecular ID and Genes in each sample assigned to a molecular ID columns.

```
###THIS MATTERS- MOLECULAR ID's LINKED TO THE PHENOTABLE
htseq_genecounts_df <- plyr:: join_all(listOf_alldf, by = NULL,
                                type = "full", match = "all")
head(htseq_genecounts_df, n = 1)
```

```
##   Gene M00000300 M00000302 M00000304 M00000305 M00000298 M00000297
## 1 A1BG       226       138       233       438       130       148
##   M00000299 M00000301 M00000303 M00000306 M00000307 M00000308 M00000019
## 1        88       163        88       132       232       240       119
##   M00000020 M00000021 M00000022 M00000002 M00000001 M00000003 M00000004
## 1       164       230       254       129        63       140       128
##   M00000007 M00000005 M00000006 M00000008
## 1       160       125       133       203
```

Optimizing compatability for creating DataSets for DESEQ2 1(EDITS = DONE)

```
#Creats HTSEQ counts matrix
#Imports everything from HTSEQ Counts dataframe except the first column (Doesn't delete column from ori
htseqCountsMat = as.matrix(htseq_genecounts_df[,-1]); ncol(htseqCountsMat)          #COLS
```

```
## [1] 24
```

```
#Sets rownames of new matrix of HTSEQ count data to gene names in 1st column of original dataframe
rownames(htseqCountsMat)<- htseq_genecounts_df[,1]
head(htseqCountsMat, n=1)
```

```
##      M00000300 M00000302 M00000304 M00000305 M00000298 M00000297 M00000299
## A1BG       226       138       233       438       130       148        88
##      M00000301 M00000303 M00000306 M00000307 M00000308 M00000019 M00000020
## A1BG       163        88       132       232       240       119       164
##      M00000021 M00000022 M00000002 M00000001 M00000003 M00000004 M00000007
## A1BG       230       254       129        63       140       128       160
##      M00000005 M00000006 M00000008
## A1BG       125       133       203
```

```
#Repeated processes for SALMON
```

Repeated processes w/salmon.txt files

```
# Assigns gene names from salmon counts dataframe to variable in order to keep them set as characters/s
salmonCounts_genes <- as.character(salmonCounts_df[ , 25] )
# Creates matrix of salmon counts from salmon count data frame
# Converts numbers with decimals to integers (no decimals)  in salmon counts matrix,
salmonCountsMat <- sapply(salmonCounts_df[ , -25],as.integer)
#head(salmonCounts_df)
# Sets rownames of new salmon counts matrix to gene names of salmon counts dataframe
rownames(salmonCountsMat) <- salmonCounts_genes
head(salmonCountsMat, n = 1)
```

```
##      M00000300 M00000302 M00000304 M00000305 M00000298 M00000297 M00000299
## A1BG       226       139       230       446       131       154        85
##      M00000301 M00000303 M00000306 M00000307 M00000308 M00000019 M00000020
```

```
## A1BG       174        88       131       244       243       115       150
##      M00000021 M00000022 M00000002 M00000001 M00000003 M00000004 M00000007
## A1BG       212       247       137        64       127       138       164
##      M00000005 M00000006 M00000008
## A1BG       125       130       194
```

```
#str(salmonCountsMat)
```

```
#Creates matrix phenoMat from df_combined dataframe
# Imports all data/columns except molecular ID column (doesnt delete from original dataframe, just does
phenoMat = as.matrix(subset(df_combined, select=-molecular_id)); nrow(phenoMat)
```

```
## [1] 24
```

```
#Sets rownames of phenoMat to data from df_combined  column of molecular ID's
rownames(phenoMat)<- df_combined$molecular_id
```

```
#Link column names AND column values  of htseqCountsMat to phenoMat row names in same order
phenoMat <- phenoMat[match(colnames(htseqCountsMat),row.names(phenoMat)),]
head(phenoMat, n = 1)
```

```
##           assay_material_id sex    race      age_range   diagnosis
## M00000300 "R0299"           "male" "unknown" "elderly61" "mf"
##           collection_event acquisition_date tissue_type
## M00000300 "diagnosis"      "8/22/13"        "peripheralblood"
##           tissue_type_origin collection_type processing_type
## M00000300 NA                 "edta"          "mononuclearcells"
##           time_to_processing     storage         shipping genotype_jak2
## M00000300 "repositoryprocessing" "cryopreserved" "fresh"  "negative"
##           genotype_calr genotype_mpl    material_type extraction_type
## M00000300 "positive"    "notdetermined" "RNA"         "column"
##           na_260280 na_260230 rin_range     jak2_vaf genomics_types
## M00000300 "2.05"    "1.81"    "highquality" " 0.0"   "rnaseq"
##           omics_sample_name omics_contact_id omics_date
## M00000300 "JAK2-10-1-D"     "jradich"        "2/27/14"
##           seq_flowcell_id               seq_readlength seq_paired
## M00000300 "140227_SN367_0370_AH8JPDADXX" "99"           "yes"
##           seq_libtype
## M00000300 "truseq"
```

Creating original DeseqDatasets

(EDITS = DONE)

```
#DataSet for DESEQ from HTSEQ Matrix created
# variable comparing is diagnosis [levels  = mf (myleofibrosis) and normal (no myleofibrosis) ]
dseq_set_htseq <- DESeqDataSetFromMatrix(htseqCountsMat,phenoMat, design = ~ diagnosis)

# with base level being at "normal" (not myleofibrosis)
dseq_set_htseq$diagnosis <- relevel(dseq_set_htseq$diagnosis, "normal")

#gets rid of rows of gene counts of 1's and 0's
dseq_set_htseq <- dseq_set_htseq[ rowSums(counts(dseq_set_htseq)) > 1, ]
```

```
dseq_set_salmon <- DESeqDataSetFromMatrix(salmonCountsMat,phenoMat, design = ~ diagnosis)
```

```
dseq_set_salmon$diagnosis <- relevel(dseq_set_salmon$diagnosis, "normal")

dseq_set_salmon <- dseq_set_salmon[ rowSums(counts(dseq_set_salmon)) > 1, ] #gets rid of 1's and 0's
#head(dseq_set_salmon)
```

Creating copies of original datasets for analysis/design formulas

```
###base lvl = age range
dseq_set_htseq_copy2 <- dseq_set_htseq
design(dseq_set_htseq_copy2) <- formula(~ age_range)
dseq_set_htseq_copy2 <- DESeq(dseq_set_htseq_copy2)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 5261 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

```
dseq_set_salmon_copy <- dseq_set_salmon
design(dseq_set_salmon_copy ) <- formula(~ diagnosis)
dseq_set_salmon_copy  <- DESeq(dseq_set_salmon_copy )
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 623 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

Analysis

use Deseqdataset to do analysis using deseq2 tools???

Following paragraph/description Copied from https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html#differential-expression-analysis . . . IMPORTANT!!!!!!!!!!

A DESeqDataSet object must have an associated design formula. The design formula expresses the variables which will be used in modeling. The formula should be a tilde (~) followed by the variables with plus signs between them (it will be coerced into an formula if it is not already). The design can be changed later, however then all differential analysis steps should be repeated, as the design formula is used to estimate the dispersions and to estimate the log2 fold changes of the model.

Note: In order to benefit from the default settings of the package, you should put the variable of interest at the end of the formula and make sure the control level is the first level.

Gives results of HTSEQ design formula 1:

```
res_dseq_set_htseq_copy <- results(dseq_set_htseq_copy)
head(res_dseq_set_htseq_copy, n = 2)
```

```
## log2 fold change (MLE): diagnosis mf vs normal
## Wald test p-value: diagnosis mf vs normal
## DataFrame with 2 rows and 6 columns
##           baseMean log2FoldChange     lfcSE      stat       pvalue
##          <numeric>      <numeric> <numeric> <numeric>    <numeric>
## A1BG     170.68405      0.7958968  0.195343  4.074356 4.614177e-05
## A1BG-AS1  16.17128     -1.5957025  0.337824 -4.723473 2.318504e-06
##                  padj
##             <numeric>
## A1BG     1.291139e-04
## A1BG-AS1 7.867270e-06
```

Gives results of HTSEQ design formula 2:

```
res_dseq_set_htseq_copy2 <- results(dseq_set_htseq_copy2)
head(res_dseq_set_htseq_copy2, n = 2)
```

```
## log2 fold change (MLE): age range unknown vs adult18to60
## Wald test p-value: age range unknown vs adult18to60
## DataFrame with 2 rows and 6 columns
##           baseMean log2FoldChange     lfcSE      stat       pvalue
##          <numeric>      <numeric> <numeric> <numeric>    <numeric>
## A1BG     170.68405      0.4789206 0.2418622  1.980138 4.768799e-02
## A1BG-AS1  16.17128     -1.6439004 0.3623803 -4.536395 5.722386e-06
##                  padj
##             <numeric>
## A1BG     8.137994e-02
## A1BG-AS1 2.475171e-05
```

```
###p val dif = low, chance there is a sig. dif.. threshhold = 0.05?
```

```
res_dseq_set_salmon_copy <- results(dseq_set_salmon_copy)
head(res_dseq_set_salmon_copy, n = 1)
```

```
## log2 fold change (MLE): diagnosis mf vs normal
## Wald test p-value: diagnosis mf vs normal
## DataFrame with 1 row and 6 columns
##       baseMean log2FoldChange     lfcSE      stat       pvalue
##      <numeric>      <numeric> <numeric> <numeric>    <numeric>
## A1BG  171.5493      0.8942849 0.1972155  4.534557 5.772439e-06
##              padj
##         <numeric>
## A1BG 1.952722e-05
```

Prep for plotting visual analysis

Log2 fold change

HTSEQ

```
###full join by gene

###create new column with values of rownames
res_dseq_set_htseq_copy$Gene <- rownames(res_dseq_set_htseq_copy)

head(res_dseq_set_htseq_copy)
```

```
## log2 fold change (MLE): diagnosis mf vs normal
## Wald test p-value: diagnosis mf vs normal
## DataFrame with 6 rows and 7 columns
##             baseMean log2FoldChange     lfcSE       stat       pvalue
##            <numeric>      <numeric> <numeric>  <numeric>    <numeric>
## A1BG     170.6840521      0.7958968 0.1953430  4.0743564 4.614177e-05
## A1BG-AS1  16.1712821     -1.5957025 0.3378240 -4.7234733 2.318504e-06
## A1CF       0.4571053     -2.0708240 1.4841992 -1.3952467 1.629415e-01
## A2M       47.3634595     -1.5682386 0.4316494 -3.6331307 2.800031e-04
## A2M-AS1   29.7386172     -0.3417973 0.3672911 -0.9305897 3.520659e-01
## A2ML1      7.2833133      1.4117902 0.5805970  2.4316181 1.503155e-02
##                  padj        Gene
##             <numeric> <character>
## A1BG     1.291139e-04        A1BG
## A1BG-AS1 7.867270e-06    A1BG-AS1
## A1CF     2.238507e-01        A1CF
## A2M      6.915249e-04         A2M
## A2M-AS1  4.314028e-01     A2M-AS1
## A2ML1    2.700069e-02       A2ML1
```

```
###Subset/put in a copy of the gene column and logfold2 column from the htseq data into new object/df
gene_l2fc_htseq1 <- as.data.frame(res_dseq_set_htseq_copy[ ,c(7, 2)])

###Getting rid of rownames (gene list is in new column)
rownames(gene_l2fc_htseq1) = NULL
###Repeating same process for Salmon as HTSEQ
```

SALMON

```
res_dseq_set_salmon_copy$Gene <- rownames(res_dseq_set_salmon_copy) #salmon
gene_l2fc_salmon1 <- as.data.frame(res_dseq_set_salmon_copy[ ,c(7, 2)])
rownames(gene_l2fc_salmon1) = NULL
```

Tests

```
###Tests
head(gene_l2fc_htseq1, n = 1) #gene_l2fc_salmon1
```

```
##   Gene log2FoldChange
## 1 A1BG      0.7958968
```

```
str(gene_l2fc_htseq1) #gene_l2fc_salmon1
```

```
## 'data.frame':    21920 obs. of  2 variables:
##  $ Gene          : chr  "A1BG" "A1BG-AS1" "A1CF" "A2M" ...
##  $ log2FoldChange: num  0.796 -1.596 -2.071 -1.568 -0.342 ...
```

HTSEQ & Salmon

```
###Joining two DF's of LF2C by Gene
gene_l2fc_htseqSalmon1_combo <- full_join(gene_l2fc_htseq1, gene_l2fc_salmon1, by = "Gene")
#Setting Column names
colnames(gene_l2fc_htseqSalmon1_combo) <- c("Gene", "L2FC_HTSEQ","L2FC_SALMON")
head(gene_l2fc_htseqSalmon1_combo, n = 2)
```

```
##       Gene L2FC_HTSEQ L2FC_SALMON
## 1     A1BG  0.7958968   0.8942849
## 2 A1BG-AS1 -1.5957025  -1.7277027
```

P-value

HTSEQ

```
#res_dseq_set_htseq_copy #htseq

#gene_l2fc_htseq1_all <- as.data.frame(res_dseq_set_htseq_copy)

#res_dseq_set_salmon_copy #Salmon

gene_pvalue_htseq1 <- as.data.frame(res_dseq_set_htseq_copy[ ,c(7, 5)])
###Getting rid of rownames (gene list is in new column)
rownames(gene_pvalue_htseq1) = NULL
```

Salmon

```
###Repeating same process for Salmon as HTSEQ
gene_pvalue_salmon1 <- as.data.frame(res_dseq_set_salmon_copy[ ,c(7, 5)])
rownames(gene_pvalue_salmon1) = NULL
```

Tests

```
###Tests
head(gene_pvalue_htseq1, n = 2) #head(gene_pvalue_salmon1)
```

```
##       Gene       pvalue
## 1     A1BG 4.614177e-05
## 2 A1BG-AS1 2.318504e-06
```

```
str(gene_pvalue_htseq1) # str(gene_pvalue_salmon1)
```

```
## 'data.frame':    21920 obs. of  2 variables:
##  $ Gene  : chr  "A1BG" "A1BG-AS1" "A1CF" "A2M" ...
##  $ pvalue: num  4.61e-05 2.32e-06 1.63e-01 2.80e-04 3.52e-01 ...
```

HTSEQ & Salmon

```
###Joining two DF's of LF2C by Gene
gene_pvalue_htseqSalmon1_combo <- full_join(gene_pvalue_htseq1, gene_pvalue_salmon1, by = "Gene")
#Setting Column names
colnames(gene_pvalue_htseqSalmon1_combo) <- c("Gene", "PVAL_HTSEQ","PVAL_SALMON")
head(gene_pvalue_htseqSalmon1_combo, n = 2)
```

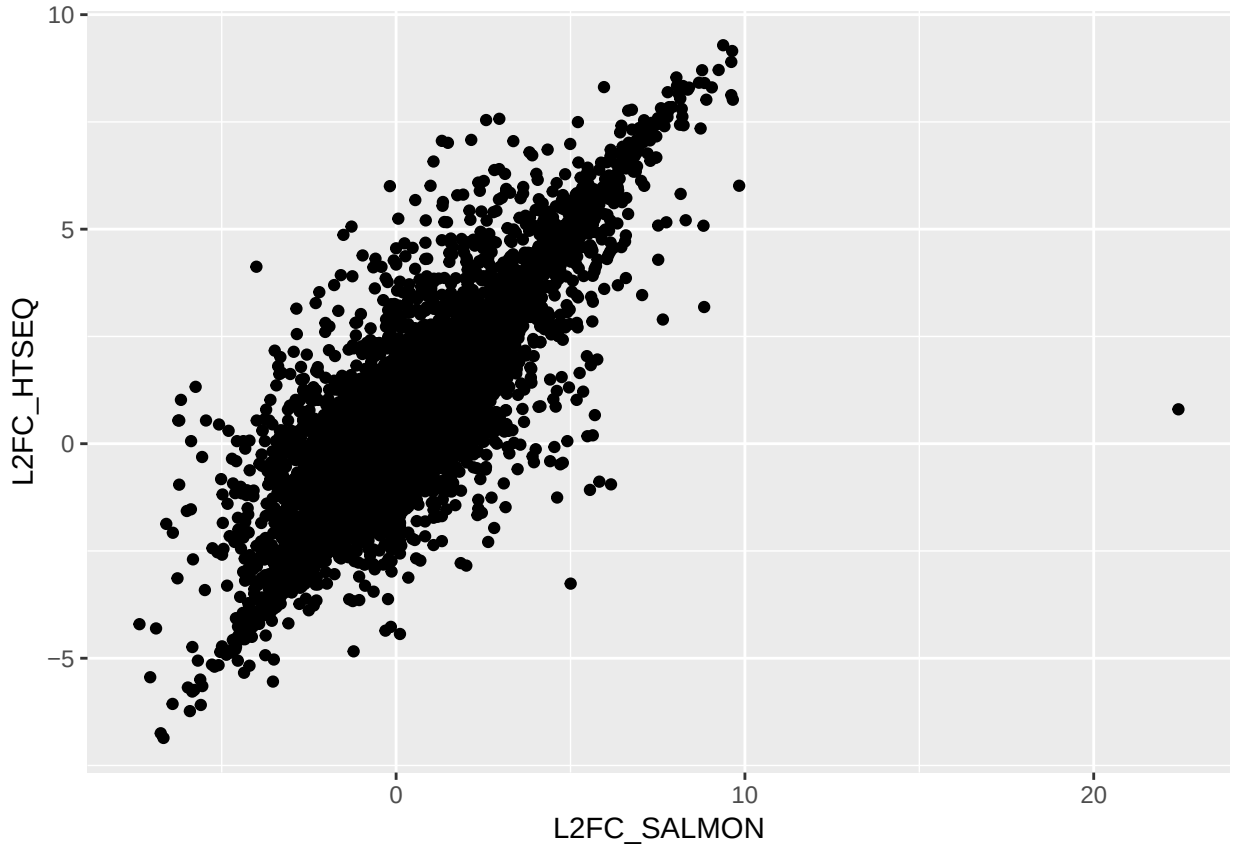
```
##       Gene   PVAL_HTSEQ  PVAL_SALMON
## 1     A1BG 4.614177e-05 5.772439e-06
## 2 A1BG-AS1 2.318504e-06 5.281749e-13
```

Visualizing analysis/plotting

Log2 Fold Change and PVAL Plots

```
p1 = ggplot(data = gene_l2fc_htseqSalmon1_combo, aes(x= L2FC_SALMON, y = L2FC_HTSEQ)) + geom_point()
p1
```

```
## Warning: Removed 4933 rows containing missing values (geom_point).
```

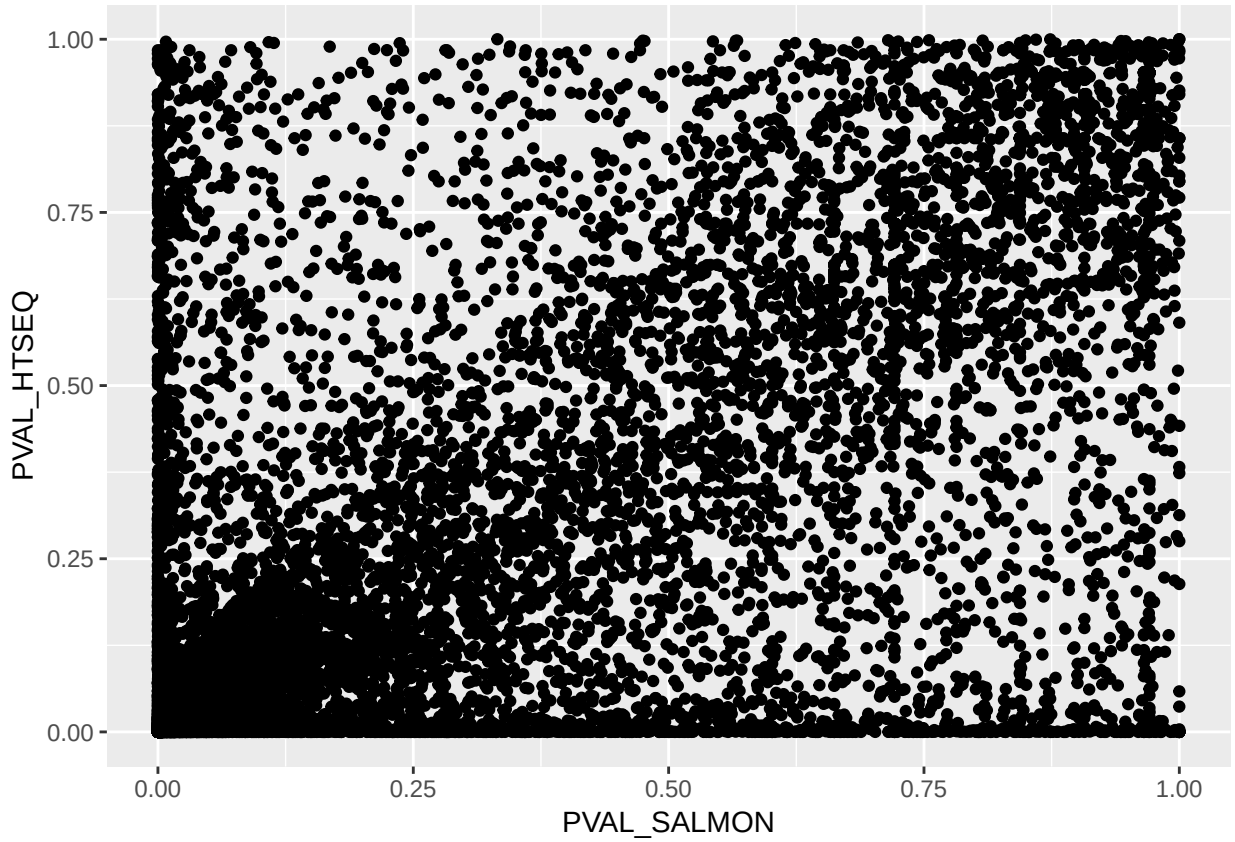



```
#EXPLAIN WHAT L2FC IS, #each dot corresponds to 2 logfoldchanges for one gene, #numbers are l2fc of mf
```

PVAL Htseq/salmon for diagnosis mf vs. normal

```
p2 = ggplot(data = gene_pvalue_htseqSalmon1_combo, aes(x = PVAL_SALMON, y = PVAL_HTSEQ)) + geom_point()
p2
```

```
## Warning: Removed 4933 rows containing missing values (geom_point).
```




```
#geom_boxplot
```

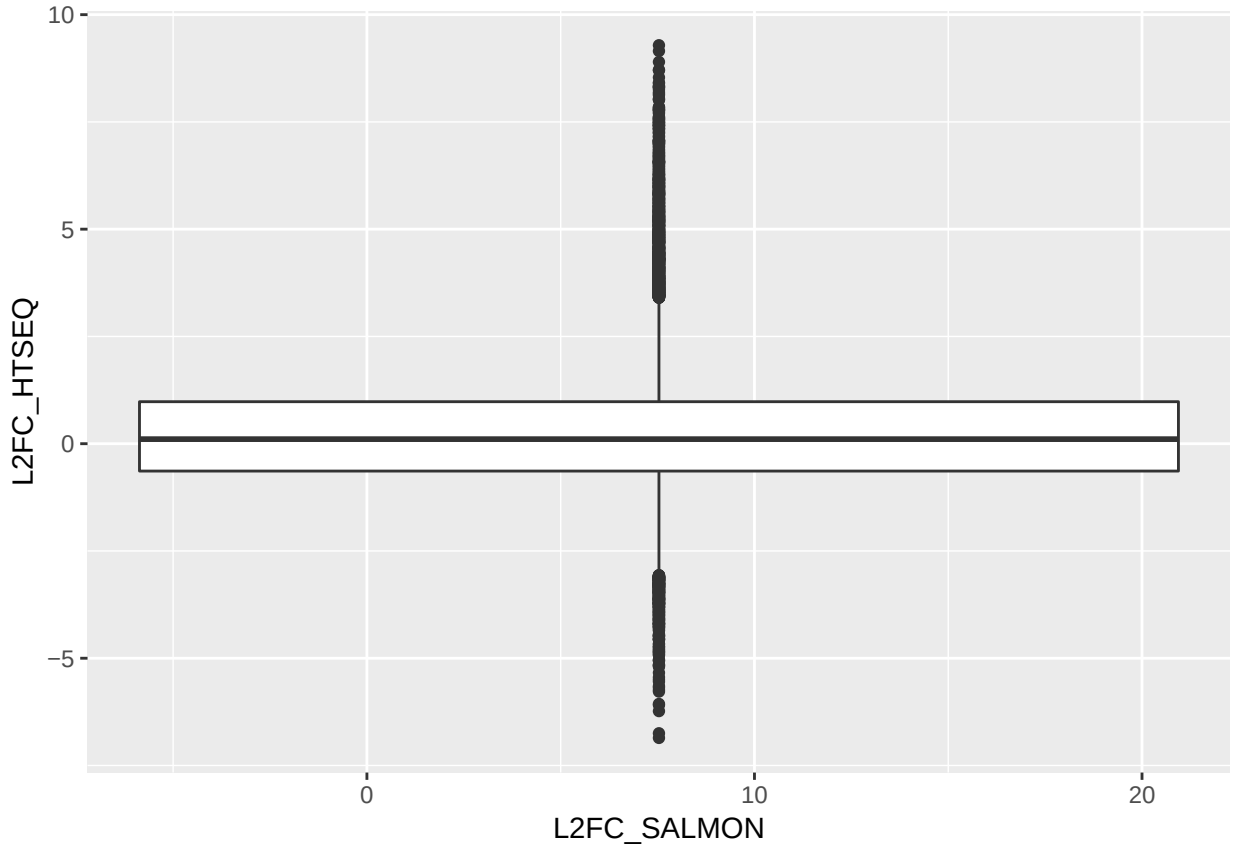
Box Plots

```
#boxplot
# htseq then salmon -  # boxplot of gene expression of gene x based on counts #for mf, #for normal #Cha
box1 = ggplot(data = gene_l2fc_htseqSalmon1_combo, aes(x= L2FC_SALMON, y = L2FC_HTSEQ)) + geom_boxplot()
box1
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```

```
## Warning: Removed 3835 rows containing missing values (stat_boxplot).
```

```
## Warning: Removed 1098 rows containing non-finite values (stat_boxplot).
```

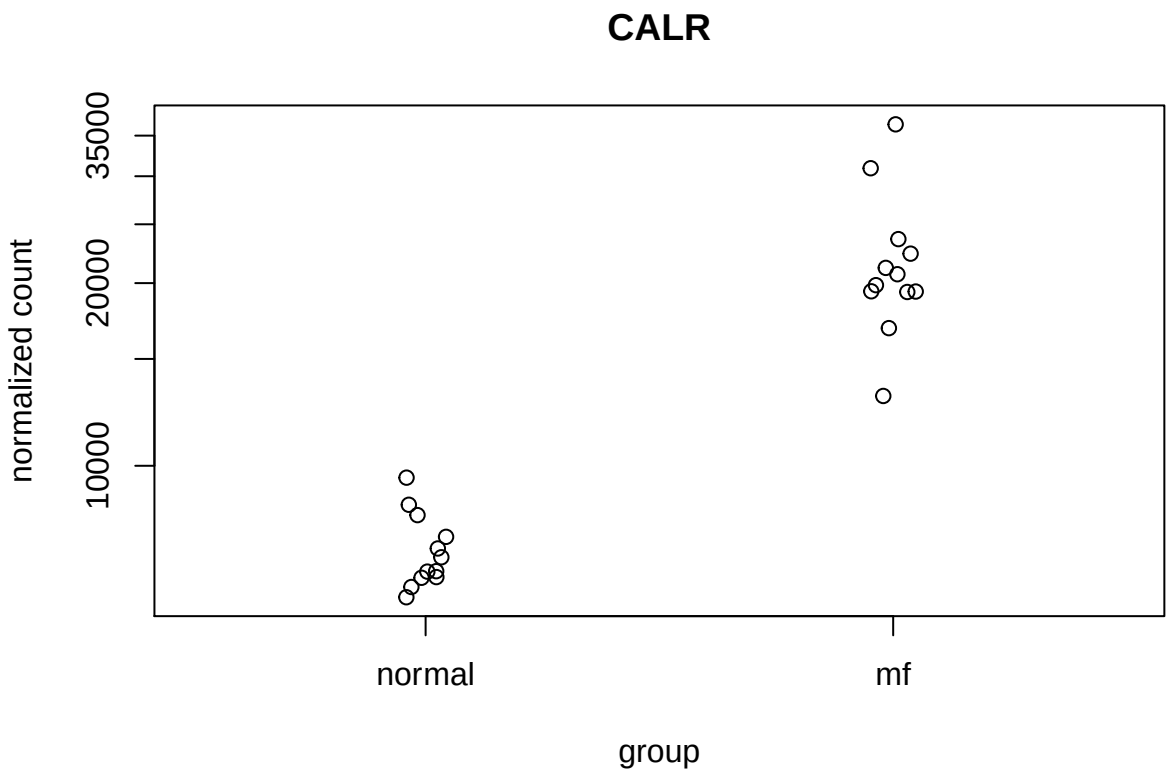



```
#box2 = ggplot(data = gene_pvalue_htseqSalmon1_combo, aes(x= PVAL_HTSEQ, y = PVAL_SALMON)) + geom_boxpl
#box2
```

Exploratory plots

Exploratory plot1

```
DESeq2::plotCounts(dseq_set_htseq, "CALR", intgroup = "diagnosis")
```




```
#DESeq2::plotCounts(dseq_set_htseq, "JAK2", intgroup = "diagnosis")
#DESeq2::plotCounts(dseq_set_htseq, "A1BG", intgroup = "diagnosis")
#DESeq2::plotCounts(dseq_set_htseq, "ADAMTS7", intgroup = "diagnosis")
```