Preprocess Heart Failure RNA Seq. Data

Aaron Troy

2022-05-01

This notebook serves as the first step in making use of the publicly available RNAseq data from heart failure patients made available by the Kass group at Johns Hopkins.

Tissue samples were collected from the right ventricular septum during heart catherterization. Bulk RNA seq was performed followed by alignment with Ensembl 92. For full details see the publication: Hahn, V. S., Knutsdottir, H., Luo, X., Bedi, K., Margulies, K. B., Haldar, S. M., Stolina, M., Yin, J., Khakoo, A. Y., Vaishnav, J., Bader, J. S., Kass, D. A., & Sharma, K. (2021). Myocardial Gene Expression Signatures in Human Heart Failure with Preserved Ejection Fraction. Circulation, 120–134. https://doi.org/10.1161/CIRCULATIONAHA.120.050498

The objectives of this notebook are as follows. Given the raw counts: - Eliminate NAs - Eliminate genes associated with red blood cells, if desired - Compute and output normalized, variance stabilized counts using DESeq's median of ratio's method and an appropriate transformation - Adjust for covariates as desired on a sample-by-sample basis - Compute DEG values using DESeq and linear modelling - Map the results, both transformed counts and DEG values to external gene ids for downstream use. - Export normalized, transformed counts for each sample and DEG values for each group.

```
suppressMessages(library(biomaRt))
suppressMessages(library(DESeq2))
```

Lets start by loading the required libraries and defining some controlling flags

```
## Warning: package ’matrixStats’ was built under R version 4.2.2
```

```
suppressMessages(library(limma))
suppressMessages(library(dplyr))
```

```
## Warning: package ’dplyr’ was built under R version 4.2.2
```

```
suppressMessages(library(textshape))
suppressMessages(library(EnhancedVolcano))
```

```
## Warning: package ’ggplot2’ was built under R version 4.2.2
```

```
suppressMessages(library(vsn))
suppressMessages(library(data.table))
```

```
## Warning: package 'data.table' was built under R version 4.2.2
```

```
suppressMessages(library(DGEobj.utils))
```

```
## Warning: package 'DGEobj.utils' was built under R version 4.2.2
```

```
suppressMessages(library(GenomicFeatures))

options(warn=-1)

#Method for variance stabilization for normalized counts. Default is log2(x+1). Other options as 'vst'
vsMeth <- 'vst'

#Flag to regress out the selected covariates in the transformed counts
rgCovs <- TRUE

#Set to false to retain genes tied to RBCs
removeGenesRBC <- TRUE

#List of covariates to control for in the normalized counts. NOTE: for the time being, correction for s
covFlags <- data.frame('age' = TRUE, 'sex' = TRUE, 't2dm' = FALSE, 'bmi' = FALSE, 'eGFR' = FALSE)

#Flag for continuous covariates. These will be transformed to a normal dist.
isCont <- data.frame('age' = TRUE, 'sex' = FALSE, 't2dm' = FALSE, 'bmi' = TRUE, 'eGFR' = TRUE)
```

Load in the raw sequencing counts and patient data. The later includes all clinical info. Perform some basic conditioning.

- Remove NA values
- If desired, remove genes strongly tied to RBCs given by a predefined list.

```
setwd("~/Bioinformatics/Network Analysis of HFpEF/Hahn_FGN_Analysis/")

sampInfo <- read.csv("data/clinicalData.csv")
rawCnts <- read.csv("data/RawReads_VS_Ensembl_geneName.csv", check.names = FALSE, header = TRUE) %>%
  na.omit() %>%
  column_to_rownames("geneID") %>%
  as.matrix()

#If selected, eliminate transcripts tied strongly to red blood cells
if(removeGenesRBC) {

  #Read in a list of RBC genes to exclude. This come directly from the Hahn et al. publication.
  genesRBC <- read.csv("data/ExcludeRBCgenes.csv")

  #Remove the genes
  clnCnts <- rawCnts[!(rownames(rawCnts) %in% genesRBC$Ensembl), ]
}

head(sampInfo)
```

```
##     id tissue disease age    sex t2dm  bmi eGFR
## 1 7542    RVS   HFpEF  63 Female  Yes 42.6   48
## 2 7546    RVS   HFpEF  63 Female   No 30.7   51
## 3 7531    RVS   HFpEF  41 Female  Yes 47.8  106
## 4 7715    RVS   HFpEF  65 Female   No 36.8   61
## 5 7529    RVS   HFpEF  51 Female   No 35.8   13
## 6 7549    RVS   HFpEF  68   Male  Yes 45.6   31
```

```
head(clnCnts)
```

```
##                  7562 7563 7564 7565 7566 7568 7569 7570 7571 7572 7574 7575
## ENSG00000000003  165  295  155  198  227  258  254  198  335  216  316  220
## ENSG00000000005    1    5    3    5    1    5    1    5    5    2    2    8
## ENSG00000000419  956  961  869  863 1005  806  758  986 1029 1253 1075 1036
## ENSG00000000457  231  253  247  207  241  216  215  198  243  303  279  303
## ENSG00000000460   33   33   36   33   38   40   25   34   34   48   37   35
## ENSG00000000938   73  117  120  173   84   86   77   87  155  166  158   96
##                  7576 7577 7578 7579 7580 7581 7582 7583 7584 7585 7586 7587
## ENSG00000000003  208  278  274  308  209  323  163  273  348  177  220  279
## ENSG00000000005    8    5    4    2   10    1    4    4    4   58   15   15
## ENSG00000000419 1017  834  933 1463  829 1129  936 1352 1205 1382 1688 1532
## ENSG00000000457  345  265  252  313  249  392  263  286  299  256  305  321
## ENSG00000000460   53   41   36   55   65   64   46   30   55   23   47   32
## ENSG00000000938  146  119  212  213   53  368   78  127  298   59   50   43
##                  7588 7589 7590 7591 7592 7593 7594 7595 7596 7725 7726 7727
## ENSG00000000003  225  256  161  194  346  214  208  275  233  257  260  151
## ENSG00000000005    4    7   10    9   12   14    3   42    2   95   17    6
## ENSG00000000419 1396 1430 1399 1406 1579 1457 1348 1424 1418 1264 1139 1073
## ENSG00000000457  245  275  297  221  295  313  244  268  298  217  296  179
## ENSG00000000460   36   58   69   49   56   64   30   50   47   29   34   44
## ENSG00000000938   35   62   49   47   62   68   13   21   74   67   44   51
##                  7728 7729 7730 7731 7732 7733 7734 7735 7736 7737 7738 7739
## ENSG00000000003  160  313  206  184  254  215  310  254  220  280  135  196
## ENSG00000000005   27   10   45   60    6   11    2    4   19    9   14    7
## ENSG00000000419 1560 1521 1419 1107 1273 1299 1147 1317 1097 1854 1411 1092
## ENSG00000000457  281  313  247  288  300  267  369  338  316  440  180  257
## ENSG00000000460   25   46   53   59   41   33   73   28   50   56   28   65
## ENSG00000000938   74   99   99   38   87   50  107   53   12   79   51  105
##                  7740 7741 7742 7743 7744 7468 7469 7470 7471 7472 7473 7475
## ENSG00000000003  318  133  204  225  187  240  337  182  241  239  204  226
## ENSG00000000005    1    0    2   62   15    2    4   10    3    3    2    3
## ENSG00000000419 1453 1237 1404  982 1391 1002  932  880  899  868  836  737
## ENSG00000000457  253  254  345  369  286  249  282  268  329  286  305  308
## ENSG00000000460   30   19   32   38   23   46   53   35   51   63   54   64
## ENSG00000000938   67   69   64   54   68   95   79  118  158  283   40   85
##                  7476 7477 7478 7479 7481 7482 7483 7484 7485 7486 7487 7488
## ENSG00000000003  300  279  361  283  271  307  204  285  283  221  220  172
## ENSG00000000005    1    3    7    2    0    8    5    3    0    4   10    0
## ENSG00000000419  773 1058  977 1157 1021 1060  765  925  728  976  773  974
## ENSG00000000457  310  260  328  383  344  427  290  247  300  281  307  338
## ENSG00000000460   62   46   59   49   58   66   43   81   60   31   52   32
## ENSG00000000938  101   69  191  158  118   95  105  119  183  115   69  322
##                  7489 7490 7491 7492 7493 7494 7495 7496 7497 7498 7499 7500
## ENSG00000000003  175  307  289  154  304  159  278  293  206  296  245  255
## ENSG00000000005    1    3    3    5    5    0   32    6    6   54   91    5
## ENSG00000000419  878 1036  858  815  825  749  899 1168 1325 1462 1056  953
## ENSG00000000457  312  352  249  265  230  268  315  330  221  233  203  244
## ENSG00000000460   63   49   40   65   41   42   60   47   29   38   34   57
## ENSG00000000938   86  141  154   83   84  120   44   44   51  112   95   78
##                 7501 7502 7503 7504 7505 7506 7507 7508 7509 7510 7511 7512
## ENSG00000000003  159  220  217  210  241  233  195  149  188  178  176  216
## ENSG00000000005    6   27  161   10   15   33    6    2   20   35    0    7
## ENSG00000000419  890 1210 1056  997 1034  979 1164  956 1121 1056 1273 1179
## ENSG00000000457  209  307  339  333  325  328  337  138  214  195  275  344
## ENSG00000000460   26   52   68   67   56   57   37   31   41   34   54   52
## ENSG00000000938   22   56   58  161   59   58   79   56   45   34   53   91
##                 7513 7514 7515 7516 7517 7518 7519 7520 7521 7522 7523 7524
## ENSG00000000003  221  131  275  140  138  186  274  202  149  156  196  187
## ENSG00000000005  835   18    2   32   11   29    3    6    8   26   16    1
## ENSG00000000419  835  924  771 1007  963 1103 1158 1217 1133 1173 1123 1229
## ENSG00000000457  289  231  274  223  178  248  263  269  259  245  173  230
## ENSG00000000460   70   40   46   35   31   30   36   33   50   39   25   62
## ENSG00000000938   50   31   29   43   56   40   48   51   44   35   36   61
##                 7525 7526 7527 7528 7529 7530 7531 7532 7533 7534 7535 7536
## ENSG00000000003  167  301  191  268  245  210  298  284  156  201  256  235
## ENSG00000000005   11   37   23   21    0    2    0    4    1    9   15    4
## ENSG00000000419 1052 1372 1296 1236  967 1098 1017  987  903 1082 1231 1195
## ENSG00000000457  217  296  277  309  298  257  333  284  179  286  270  231
## ENSG00000000460   39   69   42   38   46   55   55   47   33   43   39   43
## ENSG00000000938   23   48   92   71   52   81   86   56   37   88  118   70
##                 7537 7538 7539 7540 7541 7542 7543 7544 7545 7547 7548 7549
## ENSG00000000003  231  321  382  287  225  282  351  244  284  305  240  273
## ENSG00000000005    5    3    4   23   10    8    3    5    6    1    0    3
## ENSG00000000419 1132 1262 1399 1206 1049  937 1157 1481  972 1240 1081 1112
## ENSG00000000457  202  288  373  303  253  318  397  284  247  339  291  281
## ENSG00000000460   28   59   87   52   47   52   54   51   60   46   37   56
## ENSG00000000938  359  401  136  117   61   62   84   17   90   54  209   63
##                 7550 7551 7552 7553 7554 7555 7556 7557 7558 7559 7711 7712
## ENSG00000000003  289  309  202  309  323  259  300  238  244  247  333  401
## ENSG00000000005    3    6   12    3    4    7    5    3    3   16    5    6
## ENSG00000000419 1026 1202 1116 1302 1077 1076 1285  912 1037 1212 1343 1601
## ENSG00000000457  268  335  283  346  338  302  290  199  244  311  359  334
## ENSG00000000460   43   64   37   56   55   54   43   40   50   59   60   65
## ENSG00000000938   85   82   82   45   54   60   72   47   84   97   84   82
##                 7713 7714 7716 7717 7718
## ENSG00000000003  248  301  264  351  174
## ENSG00000000005   10    1    1    9    5
## ENSG00000000419 1151 1306 1204 1205 1363
## ENSG00000000457  280  330  286  341  282
## ENSG00000000460   51   61   49   70   54
## ENSG00000000938   65   69  132   99   97
```

Before doing any DEG analysis, we need to performe normalization to account for differing library sizes and compositions. While not required for DEG analsyis using the DESeq pipeline, we will also complete variance stabilization. This will yield counts that can be used for a variety of other downstream analyses (e.g. cliustering). Finally, adjustment of the transformed counts for any flaged covariates will also be completed here.

Normalization For some downstream analyses, we require some type of sample by sample expression signature, rather than simple L2FC values or other stats between groups. There are a number of confounds that need to be dealt with here before proceeding.

- Library size and hence total number of reads differs between samples and must be normalized for
- Differences in the library composition. For example, a gene highly expressed in one sample may be knocked out in another. Reads attributed to this gene in the WT sample will be distributed to others in the KO, which will appear as if many gene are DE, but this is clearly not the case.

These considerations are addressed by DESeq's standard normalization, which use the median of ratios method to compute a scaling factor for each gene. This is done as follows: 1. Take the natural log of each entry 2. Take the average of these value across each gene $(\text{mean}(\ln(e_i)))$ 4. Subtract this average ln value from the ln of each count $(\ln(a) - \ln(b) = \ln(a/b))$ 5. Calculate the median for each row. I.e., the median log ratio 6. The scaling factor for each gene is then given by e exponentiation 7. Divide all counts by this scaling factor for each row

Variance stabilization Within expression datasets the variance in expression for each gene typically displays a high degree of dependence upon the mean expression of the gene. This needs to be corrected for if we wish to perform simple regression, ANOVA, or effect clustering. DESeq provides three possible transformations to complete this

Regularized log transformation:

Very complicated and slow. Takes several minutes to run for ∼100 samples. Regularizes using shrinkage.

Log-2 with psudocounts:

By far the simplest approach. Simply take $\log_2(x_{i,j} + x_0)$ for all entries, where $x_{i,j}$ is the count number for gene i, sample j. x_0 is an added pseudo count constant to avoid infinite values.

Variance stabilizing:

Estimates a function $y = f(x)$ such that $\text{Var}(Y) \perp\!\!\!\perp E(X)$. Assuming $\text{Var}(X) = h(\mu)$, a suitable basis for y is:

$$y \propto \int_0^x \frac{d\mu}{\sqrt{h(\mu)}}$$

This is the approach suggested in most cases by Michael Love

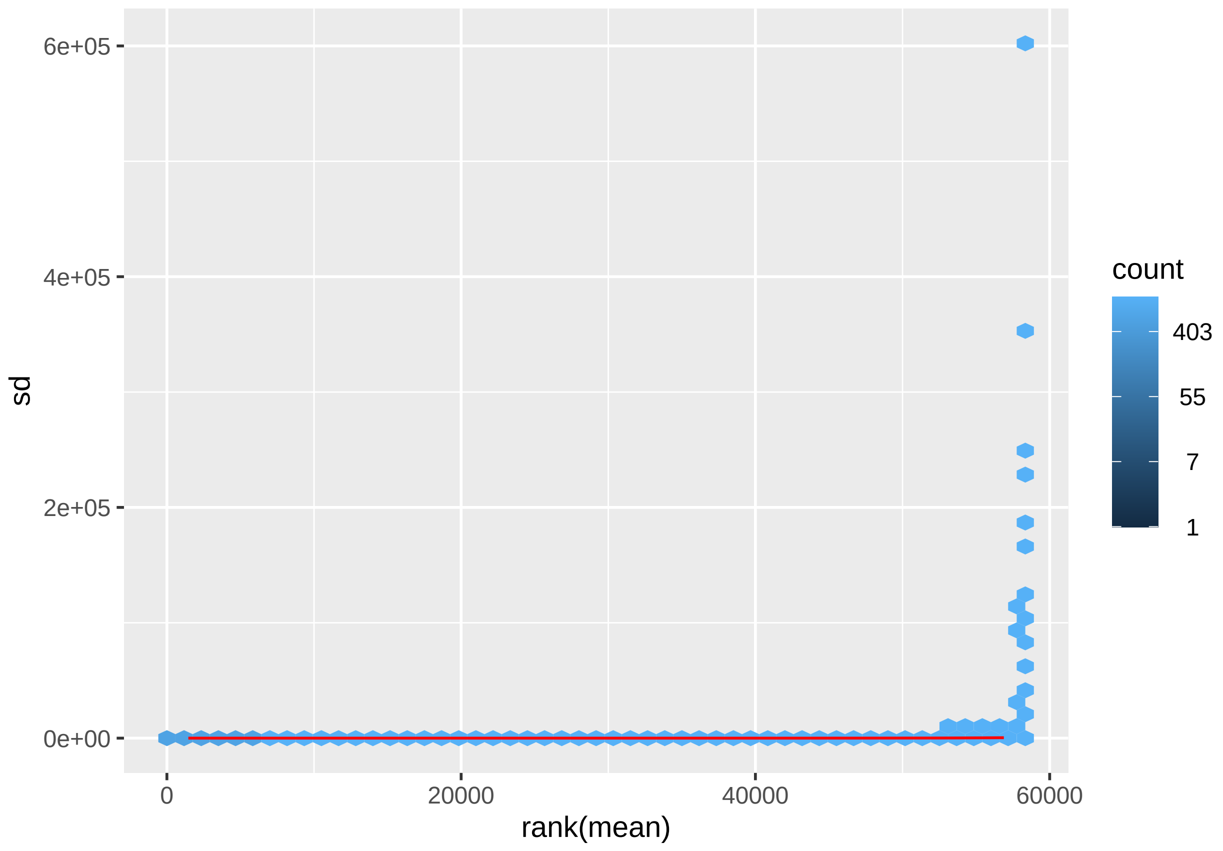
Adjust for covariates (age, sex, batch effects) in the transformed counts Adjust for covariates is often essential. The are myriad tools to complete this in R. I like to use the removeBatchEffect function from limma.

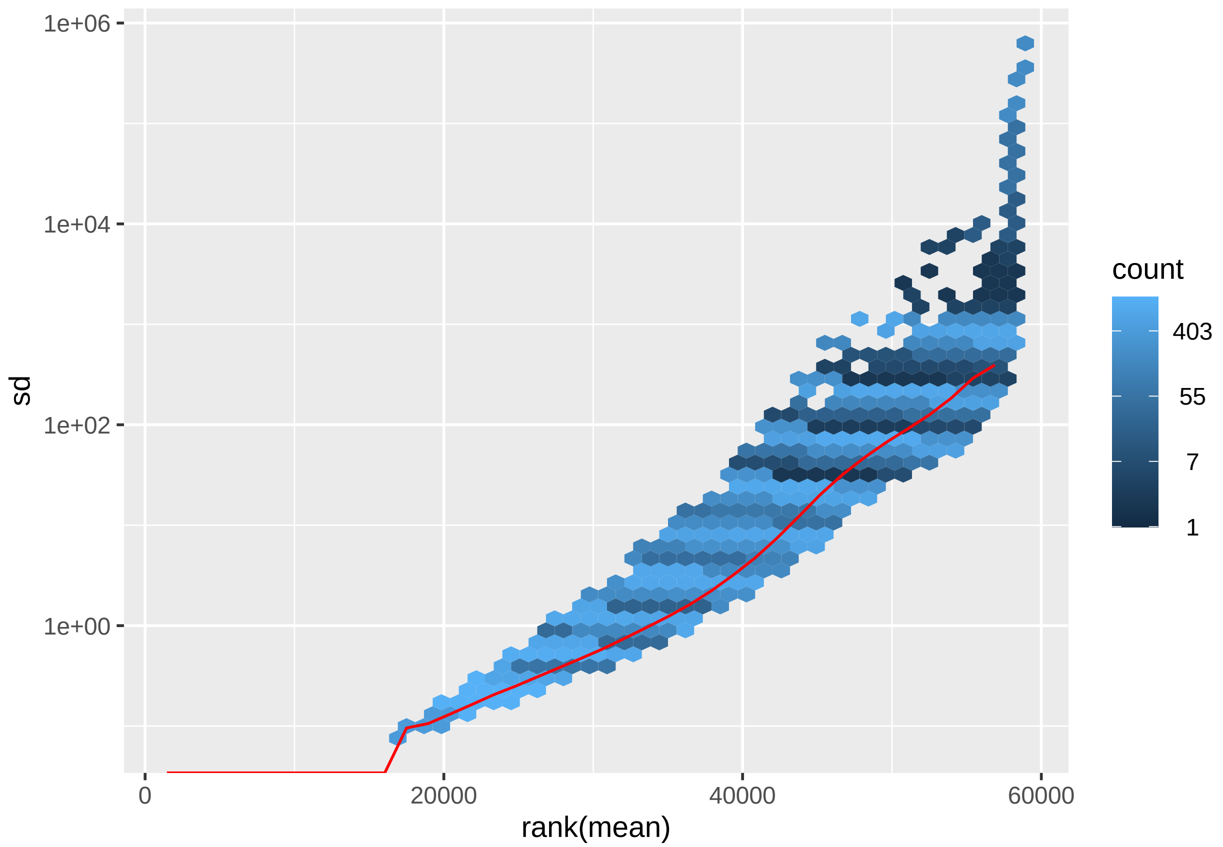
IMPORTANT: Limma wanrs against doing linear modelling after applying removeBatchEffect. You're bettwe off simply including the covariates as terms in your model. On the other hand, if you want to do clustering things should be ok.

For continuous covariates (BMI, age, eGFR), we will center and scale the data to improve model fitting (see here: https://support.bioconductor.org/p/9138042/). Note the fitted coefficients can converted back to the original scaling by multiplying by the SD of the raw data.

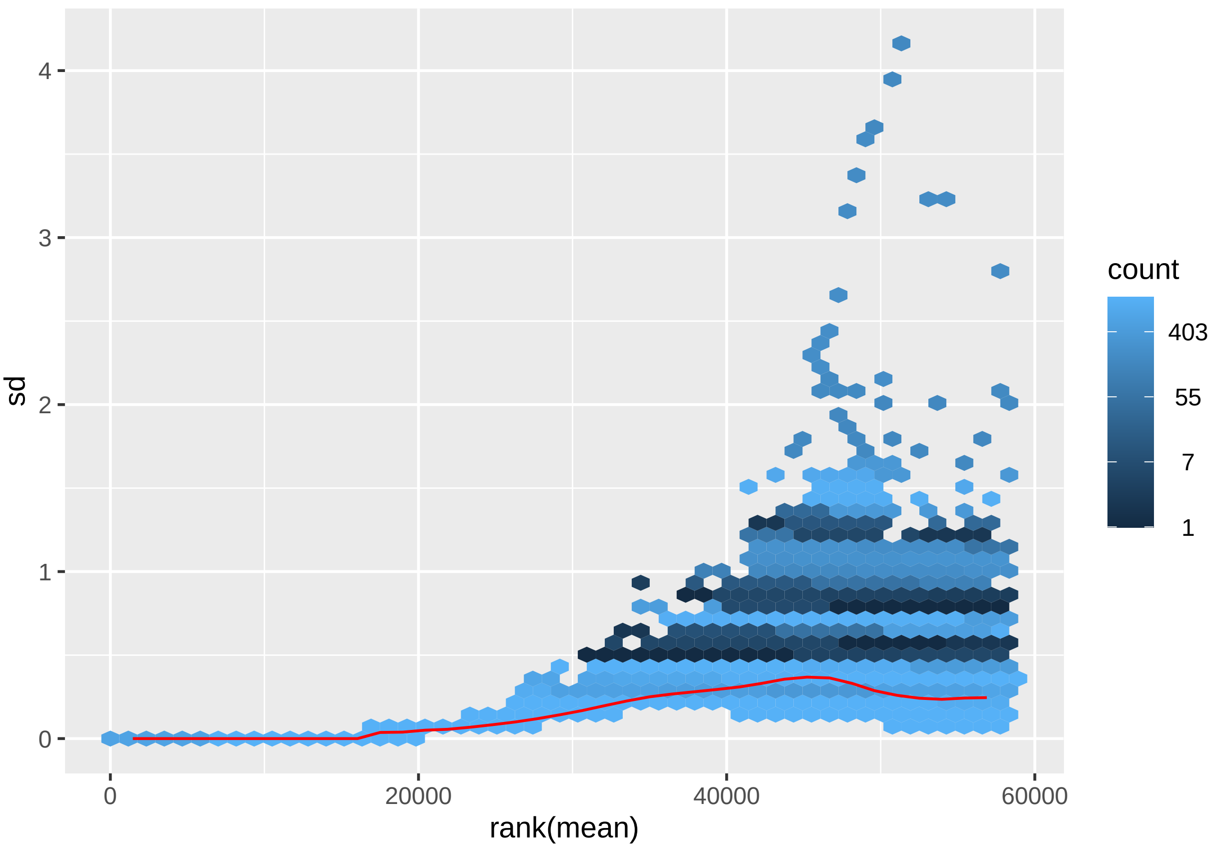
```
#Setup the covariates you want to adjust for. We do this first in order to build the DESeq object corre
contCovs <- colnames(covFlags)[as.logical(covFlags * isCont)]      #Continuous covariates
discCovs <- colnames(covFlags)[as.logical(covFlags * !(isCont))]  #Discrete covariates
covs <- colnames(covFlags)[as.logical(covFlags)]
```

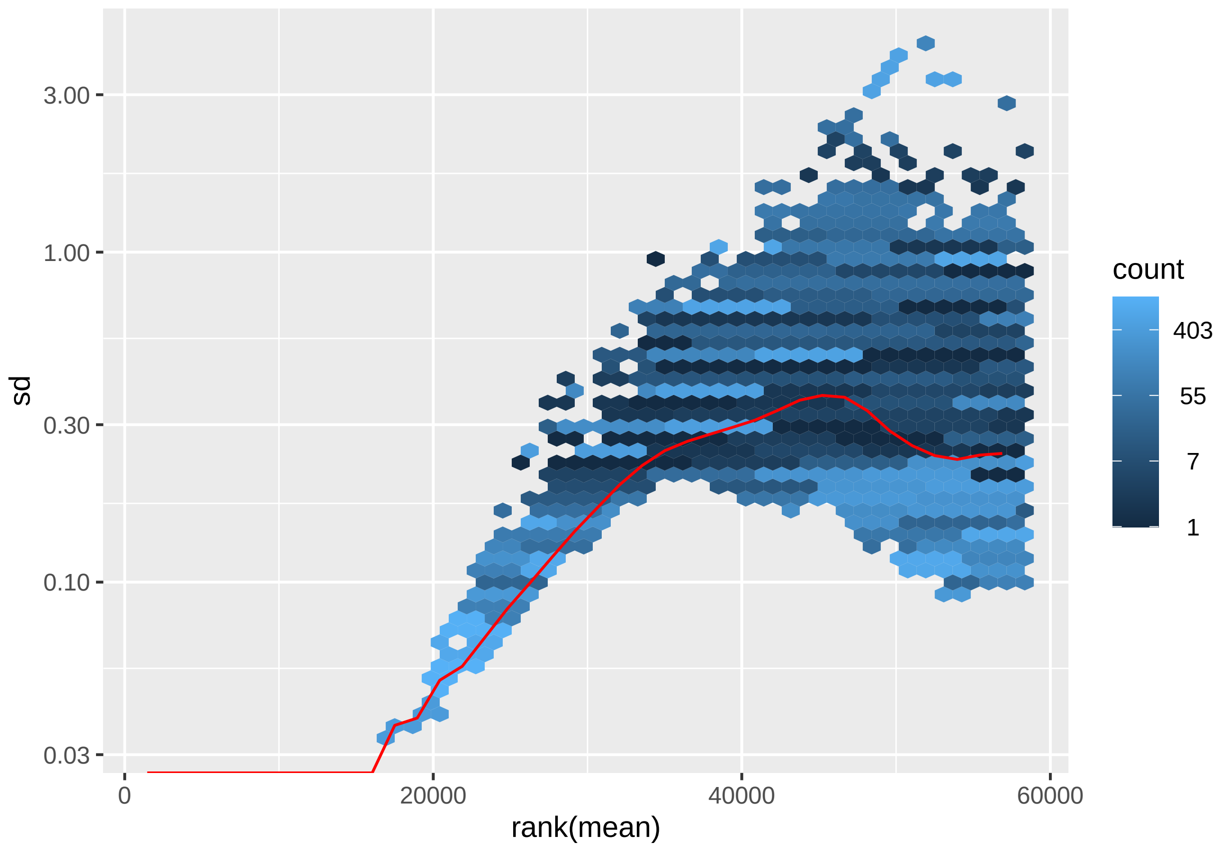
```r
if (rgCovs & !isEmpty(covs)){
  design <-  paste("~ disease", paste(covs, collapse = ' + '), sep = ' + ')
} else{
  design <-  "~ disease"
}

# Build sample info for DESeq. We do a couple things here:
# - Take only samples that we have RNAseq data for
# - Take only the clinical identifier, disease, and any covariates includes
# - Remove any patients for which there is clinical info missing
# - Zero center and scale any included convariates that are flagged as continuous.
sampInfCln <- sampInfo[as.character(sampInfo$id) %in% colnames(clnCnts), colnames(sampInfo) %in% c('id'
  na.omit() %>%
  mutate(across(.cols = contCovs, .fns = scale))

# Remove RNAseq data for patients that didn't meet the criteria just applied.
clnCnts <- clnCnts[, (colnames(clnCnts) %in% c(as.character(sampInfCln$id)))]

#Ensure things are arranged in the same order. Maybe this isn't necessary...
sampInfCln <- arrange(sampInfCln, sapply(id, function(y) which(y == colnames(clnCnts))))

# Make the DESeq object'
dds <- DESeqDataSetFromMatrix(clnCnts, sampInfCln, design = as.formula(design))

#Now let's complete normalization with size factors followed by variance stabilization
dds <- estimateSizeFactors(dds)

# Check the variance-mean dependence before stabilization
meanSdPlot(counts(dds, normalized = TRUE))$gg + scale_y_log10()
```


6e+05
4e+05
2e+05
0e+00
sd
0
20000
40000
60000
rank(mean)
count
403
55
7
1



```r
if (vsMeth == 'vst'){
  #Perform VST
  trnsCnts <- varianceStabilizingTransformation(dds, blind = FALSE)
  meanSdPlot(assay(trnsCnts))$gg + scale_y_log10()
} else if(vsMeth == 'rlog'){
  #Perform regularized log transform. Takes a while
  trnsCnts <- rlogTransformation(dds, blind = FALSE)
  meanSdPlot(assay(trnsCnts))$gg + scale_y_log10()
} else {
  trnsCnts <- normTransform(dds)
  meanSdPlot(assay(trnsCnts))$gg + scale_y_log10()
}
```

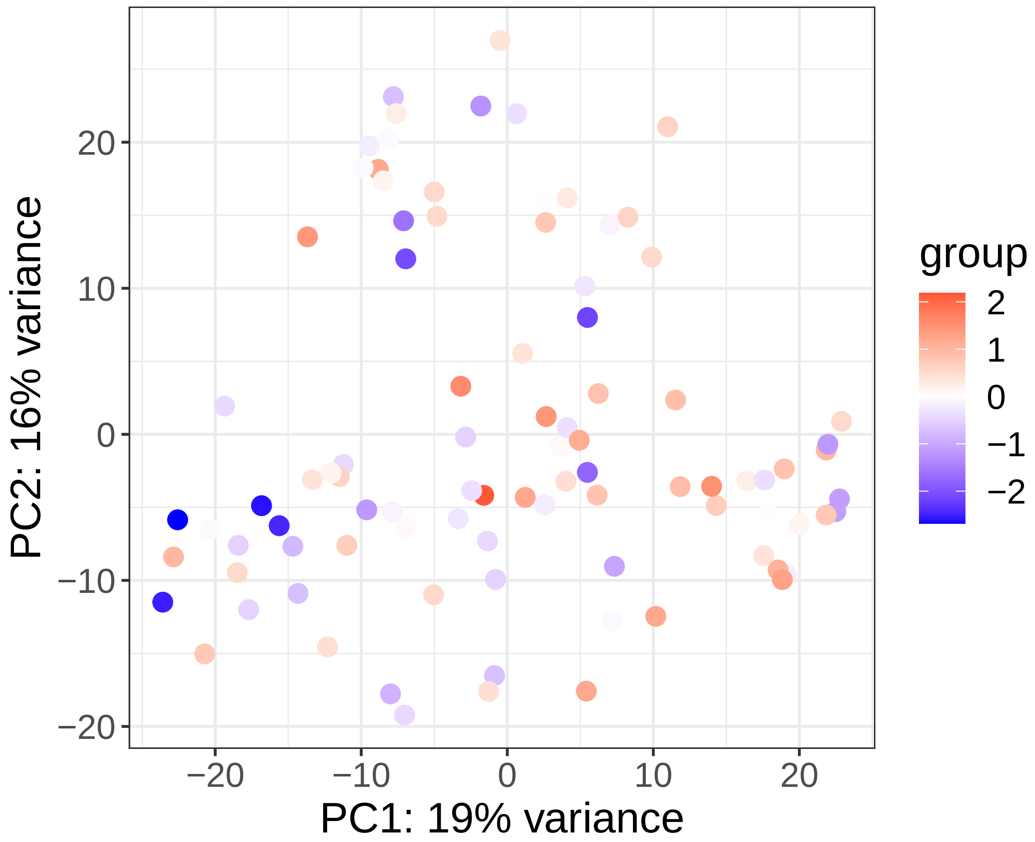

4
3
2
1
0
sd
0
20000
40000
60000
rank(mean)
count
403
55
7
1

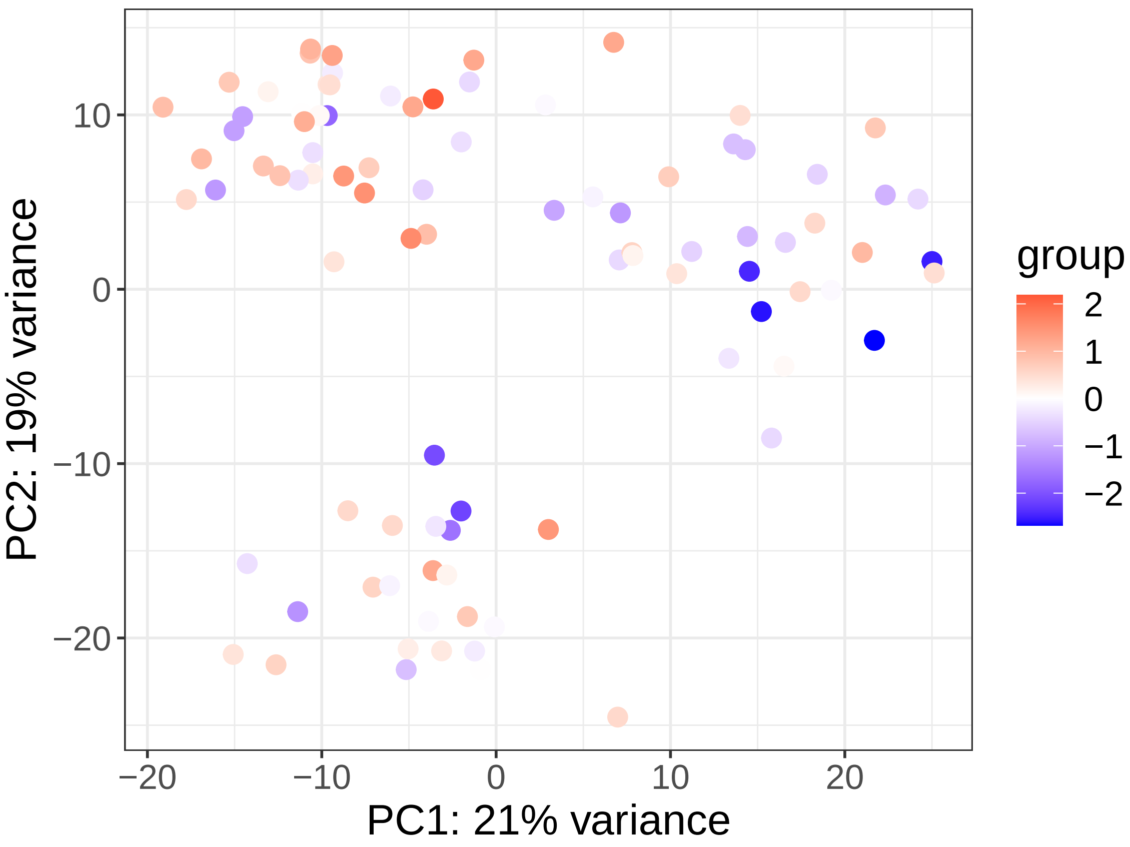


```
#Regress out the effects of the selected covariates. For now, sex is included as a discrete covariate b
if (rgCovs) {

  mat <- assay(trnsCnts)
  desMat <- model.matrix(~disease, colData(trnsCnts))
  adjCnts <- trnsCnts

  if (!isEmpty(contCovs)) {
    contCovs <- paste('~ ', paste(contCovs, collapse = ' + '))
    covMat <- model.matrix(as.formula(paste('~ ', paste(contCovs, collapse = ' + '))), colData(trnsCnts
    assay(adjCnts) <- (limma::removeBatchEffect(mat, batch = trnsCnts$sex, covariates = covMat, design
  } else {
    assay(adjCnts) <- limma::removeBatchEffect(mat, batch = trnsCnts$sex)
  }
} else {
   adjCnts <- trnsCnts
}
```

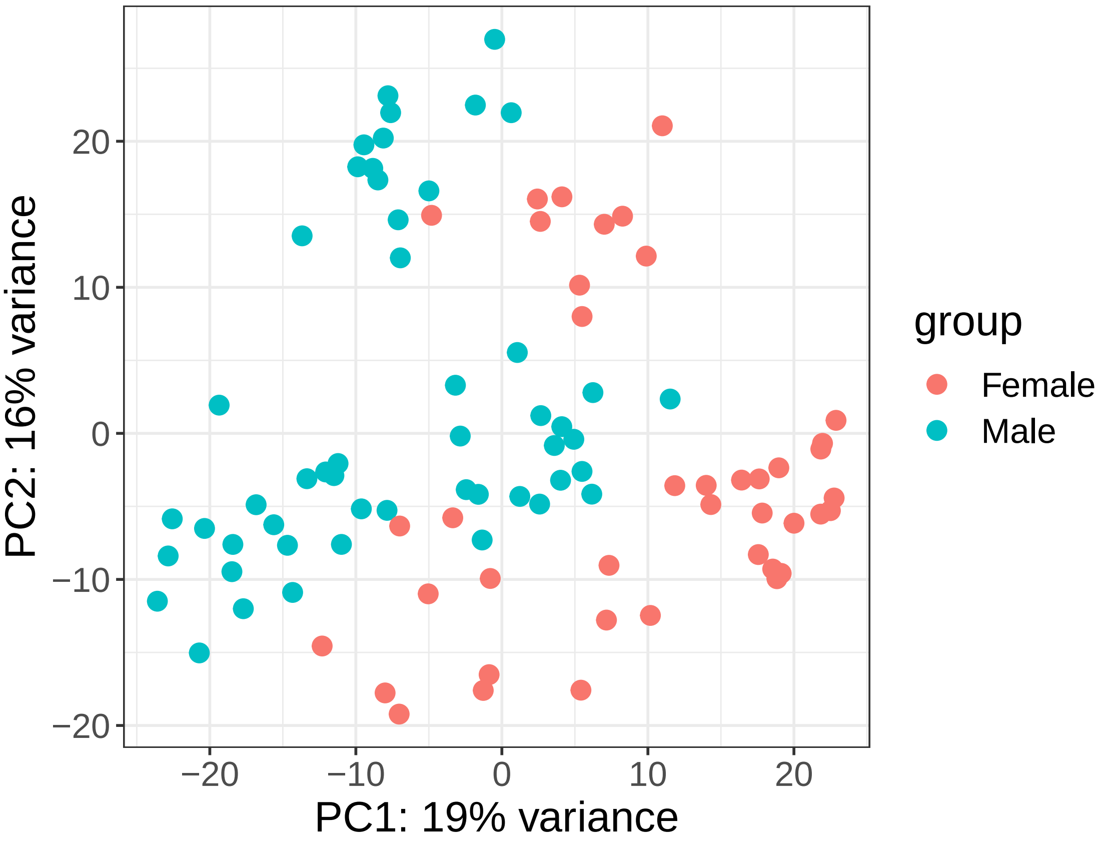
```
## Coefficients not estimable: (Intercept)
```

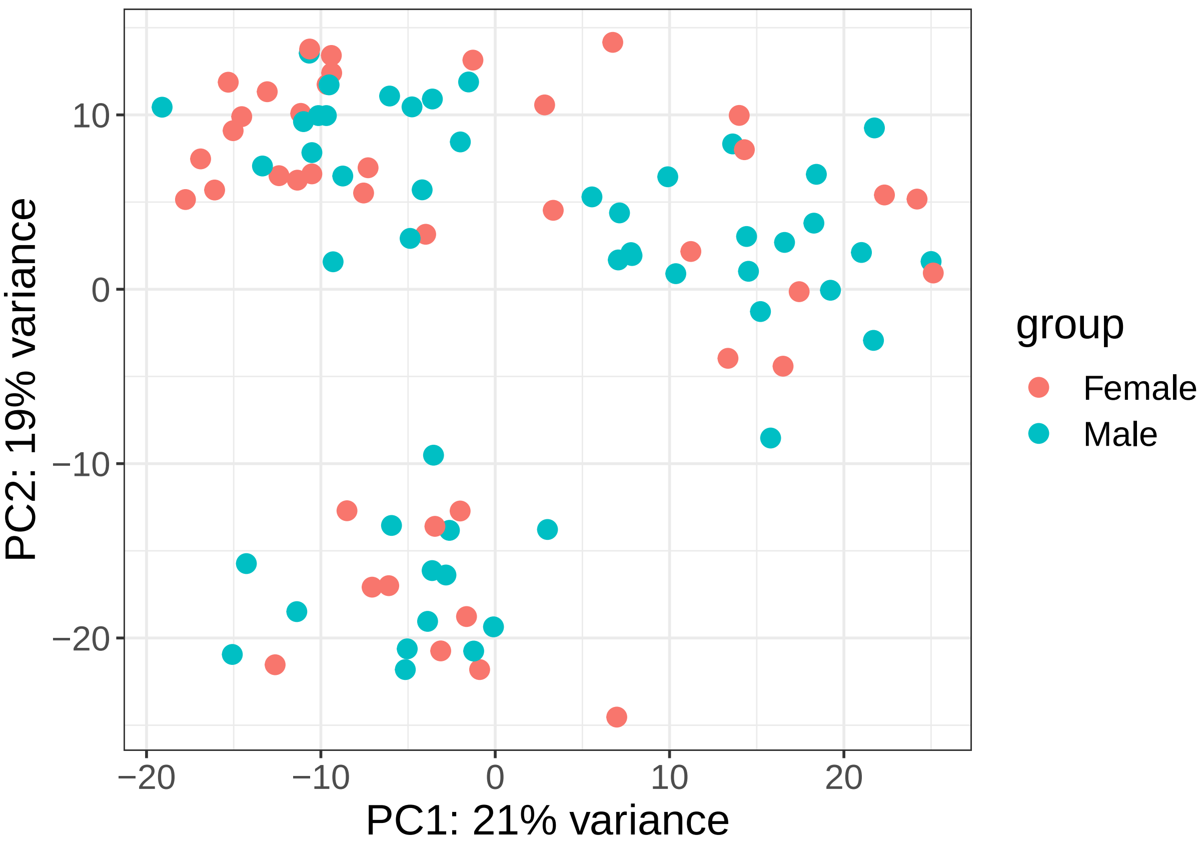
```
#Check PCA, before and after adjustment, for select covariates


if ('age' %in% covs) {
  print(plotPCA(trnsCnts, intgroup = 'age') + scale_color_gradient2(midpoint=0, low="blue", mid="white"
  print(plotPCA(adjCnts, intgroup = 'age') + scale_color_gradient2(midpoint=0, low="blue", mid="white",
}
```



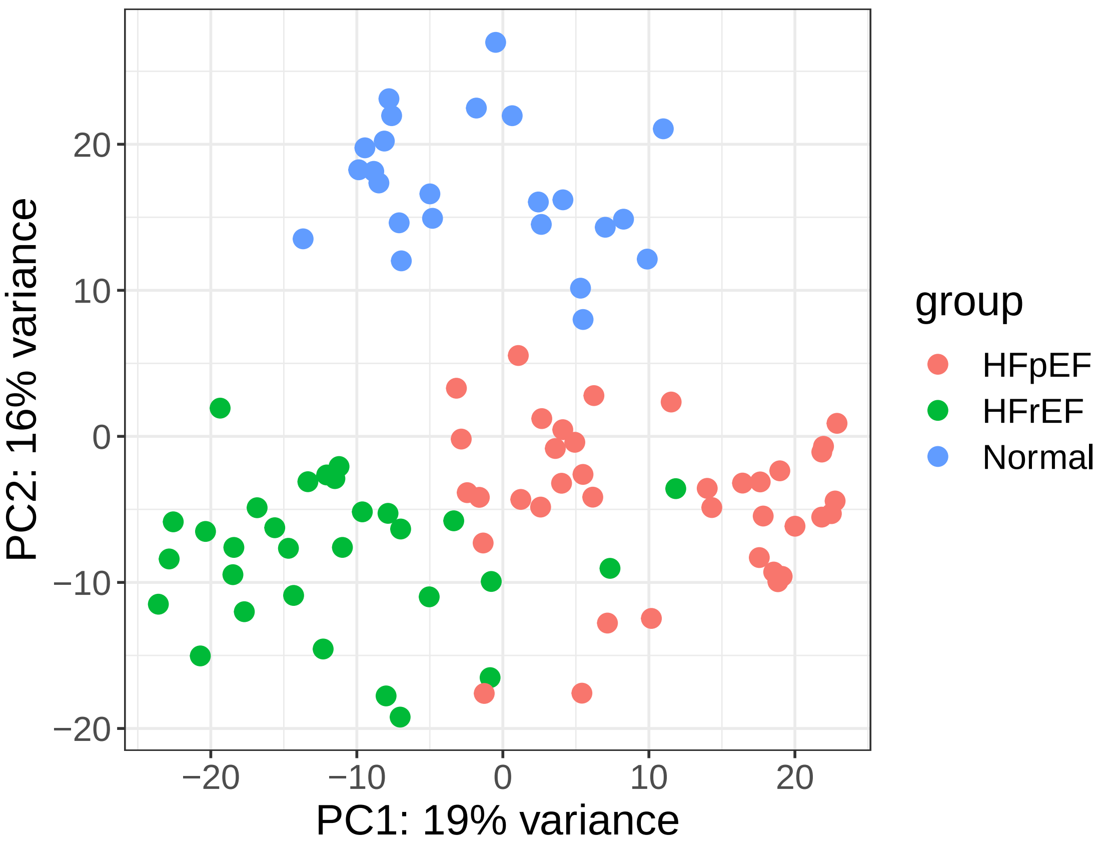


```
if ('sex' %in% covs){
  print(plotPCA(trnsCnts, intgroup = 'sex') + theme_bw() + theme(text = element_text(size = 16)))
  print(plotPCA(adjCnts, intgroup = 'sex') + theme_bw() + theme(text = element_text(size = 16)) )
}
```

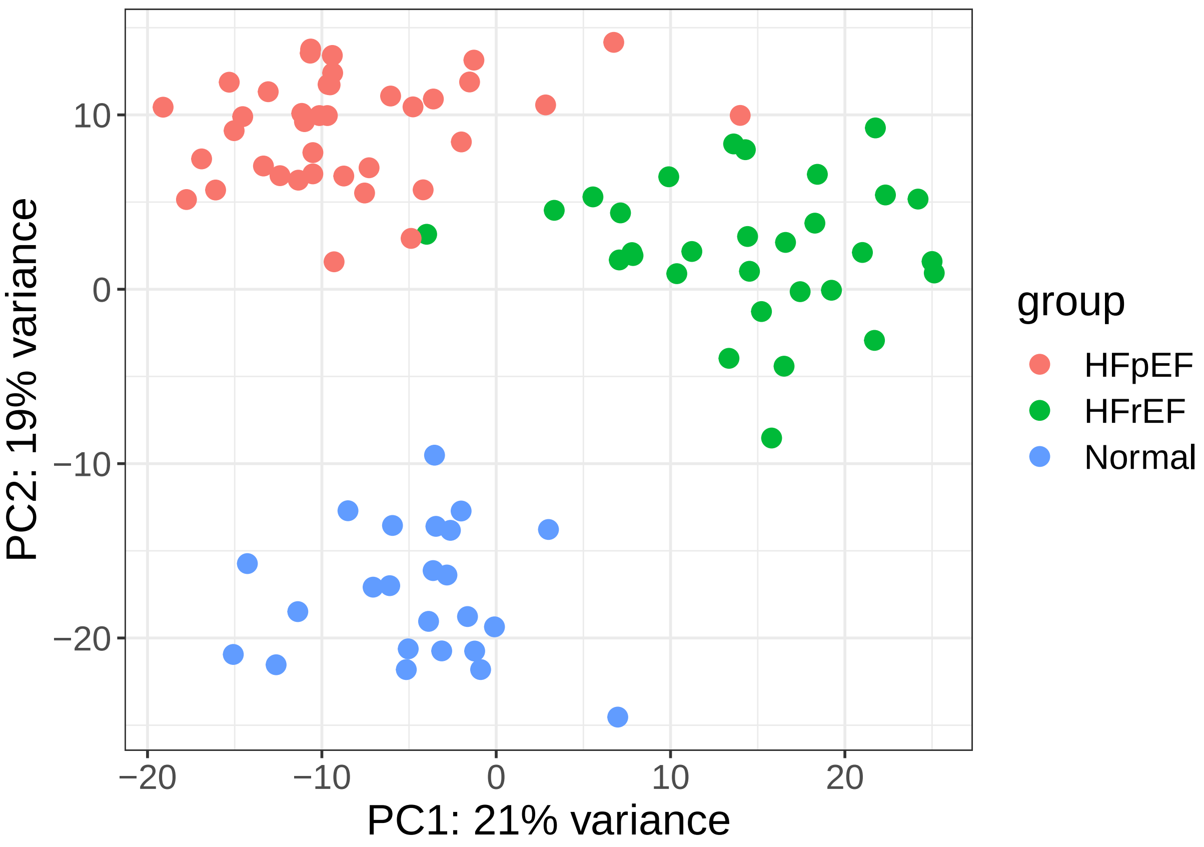

PC2: 16% variance
20
10
0
−10
−20
−20
−10
0
10
20
PC1: 19% variance
group
● Female
● Male



```
# Also check separation by disease
plotPCA(trnsCnts, intgroup = 'disease')+ theme_bw() + theme(text = element_text(size = 16))
```



```
plotPCA(adjCnts, intgroup = 'disease') + theme_bw() + theme(text = element_text(size = 16))
```



```
adjCnts <- assay(adjCnts)
```

Perform DESeq to detect DEGs Set the transformed counts assigned for the time being to compute the DEGs group by group. This will done using DESeq using a negative bionomial distribution as a prior. A few operations that are applied that you should be aware of:

- The Cook's distance (see here) is applied by default to remove outlier genes, using a threshold of the 0.99 quantile if the F distribution. These genes will be flagged in the results by a p-value and adjusted p-value set to NA
- Similarly, genes will low expression will be removed automatically and result in an adjusted p-value of NA. Not that the p-value will be retained, provided the Cook's distance passes the threshold.

- Adjusted p-values are computed using the Benjamini-Hochberg procedure by default, although alternatives can be specified.
- An α of 0.1 is used by default as the false discovery rate cutoff. Change this if desired

```
alphaFDR <- 0.05

#Perform DESeq
dds <- DESeq(dds)
```

```
## using pre-existing size factors
```

```
## estimating dispersions
## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```r
#Get the results and genes flagged (padj set to NA, be default) for low expression of high Cook's dista
resPEF <- results(dds,tidy = TRUE, contrast = c("disease","HFpEF","Normal"), alpha=alphaFDR) %>%
  rename_with(function(x) paste0(x, "_PEF"))
resREF <- results(dds,tidy = TRUE, contrast = c("disease","HFrEF","Normal"), alpha=alphaFDR) %>%
  rename_with(function(x) paste0(x, "_REF"))
resPEFREF <- results(dds,tidy = TRUE, contrast = c("disease","HFpEF","HFrEF"), alpha=alphaFDR) %>%
  rename_with(function(x) paste0(x, "_PEFREF"))

# Compile genes marked as outliers oR lowly expressed
badGenes <-  c(resPEF[is.na(resPEF$padj_PEF), "row_PEF"], resREF[is.na(resREF$padj_REF), "row_REF"], res

# Assemble the results and discard flagged genes. Add z-scores for each contrast
resDES <- merge(resPEF, resREF, by.x = 'row_PEF', by.y = "row_REF") %>%
  merge(resPEFREF, by.x = 'row_PEF', by.y = "row_PEFREF") %>%
  filter(!(row_PEF %in% badGenes)) %>%
  dplyr::rename(geneID = row_PEF) %>%
  as.data.table()

#Add z-scores. DESeq returns Wald statistics which can also be used
resDES <- resDES[, z_score_PEF := abs(qnorm(pvalue_PEF/2)) * sign(log2FoldChange_PEF)]
resDES <- resDES[, z_score_REF := abs(qnorm(pvalue_REF/2)) * sign(log2FoldChange_REF)]
resDES <- resDES[, z_score_PEFREF := abs(qnorm(pvalue_PEFREF/2)) * sign(log2FoldChange_PEFREF)]

head(resDES)
```

```
##              geneID baseMean_PEF log2FoldChange_PEF  lfcSE_PEF   stat_PEF
## 1: ENSG00000000003    241.24688         0.12583151 0.07109513  1.7699035
## 2: ENSG00000000005     10.98864         0.71611098 0.38575611  1.8563827
## 3: ENSG00000000419   1065.30018         0.38755067 0.04211131  9.2030079
## 4: ENSG00000000457    279.70059        -0.03321329 0.04870828 -0.6818818
## 5: ENSG00000000460     48.21061        -0.04267566 0.07181400 -0.5942526
## 6: ENSG00000000938     89.52089        -0.35975151 0.19220596 -1.8716980
##       pvalue_PEF     padj_PEF baseMean_REF log2FoldChange_REF  lfcSE_REF
## 1: 7.674322e-02 1.326837e-01    241.24688         -0.2242384 0.07435638
## 2: 6.339902e-02 1.128980e-01     10.98864          2.8338193 0.38841833
## 3: 3.480674e-20 1.350076e-18   1065.30018          0.3425154 0.04373767
## 4: 4.953137e-01 6.011161e-01    279.70059         -0.1564778 0.05087563
## 5: 5.523432e-01 6.535351e-01     48.21061         -0.1885063 0.07592052
## 6: 6.124840e-02 1.095933e-01     89.52089         -1.1341427 0.20100609
##      stat_REF   pvalue_REF     padj_REF baseMean_PEFREF log2FoldChange_PEFREF
## 1: -3.015725 2.563659e-03 9.351259e-03       241.24688            0.35006988
## 2:  7.295792 2.969075e-13 7.979128e-12        10.98864           -2.11770832
## 3:  7.831131 4.834990e-15 1.701219e-13      1065.30018            0.04503526
## 4: -3.075693 2.100141e-03 7.865874e-03       279.70059            0.12326452
## 5: -2.482942 1.303023e-02 3.784343e-02        48.21061            0.14583060
## 6: -5.642330 1.677643e-08 2.035611e-07        89.52089            0.77439117
##     lfcSE_PEFREF stat_PEFREF pvalue_PEFREF  padj_PEFREF z_score_PEF z_score_REF
## 1:   0.07072377    4.949819  7.428250e-07 7.566041e-06   1.7699035   -3.015725
## 2:   0.35534250   -5.959626  2.528162e-09 4.895974e-08   1.8563827    7.295792
## 3:   0.04141766    1.087344  2.768847e-01 4.146759e-01   9.2030079    7.831131
## 4:   0.04856417    2.538178  1.114312e-02 3.119980e-02  -0.6818818   -3.075693
## 5:   0.07279773    2.003230  4.515260e-02 9.944230e-02  -0.5942526   -2.482942
## 6:   0.19204906    4.032257  5.524373e-05 3.235098e-04  -1.8716980   -5.642330
##    z_score_PEFREF
## 1:       4.949819
## 2:      -5.959626
## 3:       1.087344
## 4:       2.538178
## 5:       2.003230
## 6:       4.032257
```

```
#Remove flagged genes for low expression or high Cook's distance from the counts themsselves.
adjCnts <- adjCnts[!(row.names(adjCnts) %in% badGenes), ] %>% as.data.frame()
```

Almost done. Some post processing and making things more accessible. No one has ensembl IDs memorized, so add the external gene names using bioMart. IMPORTANT: you need to use the same version of ensembl that was used during the alignment.

Even with this, there will be a large number (100 or so) emsembl IDs that map to multiple gene names (multi-mapping, MM). What to do about this will depend on what you're interested in. Certainly, just merging the expression / count values of genes with MM is not principled. Some amount of manual inspection to determine to what to do in each case is often unavoidable if you want to avoid discarding data.

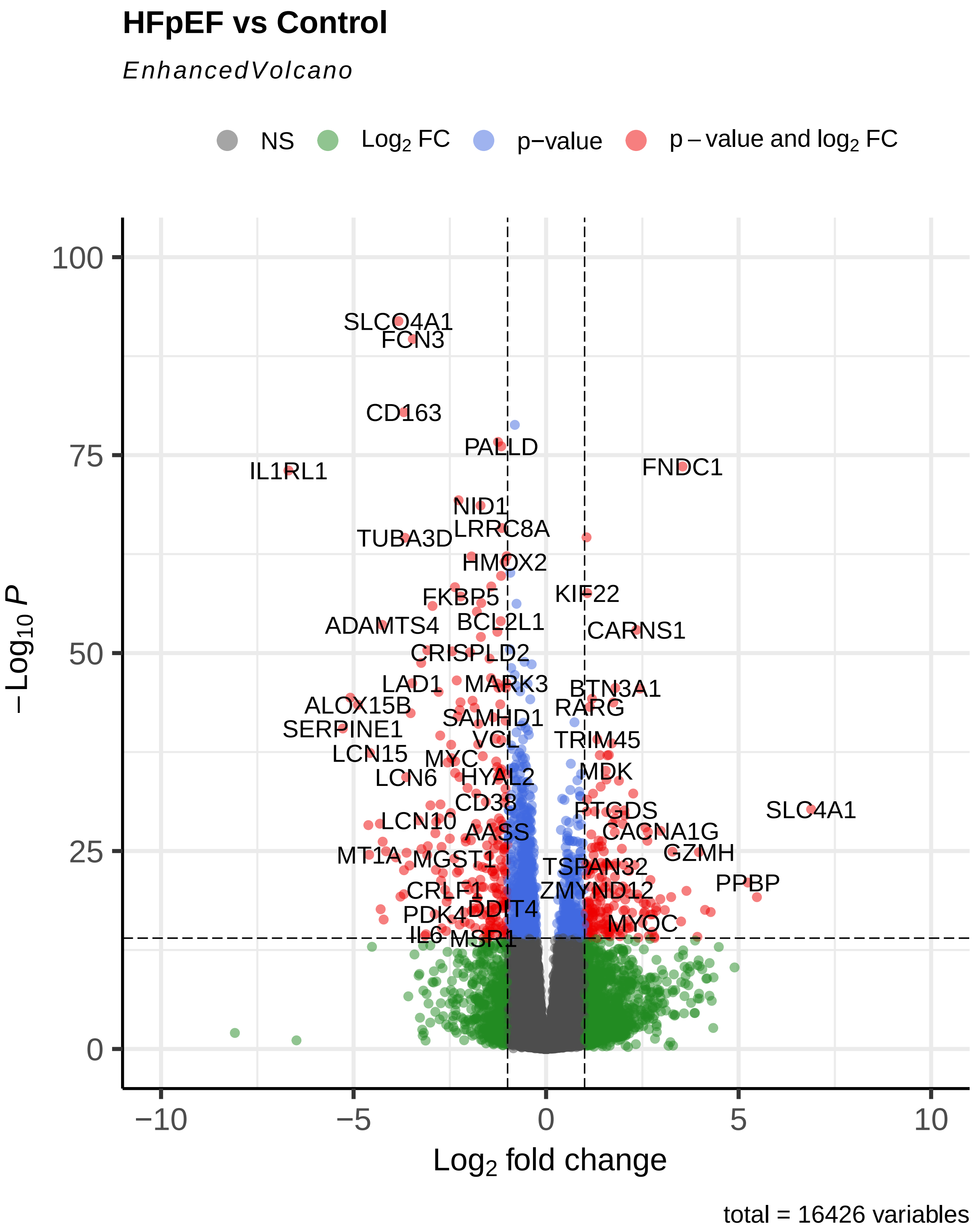
Further, if all you care about is protein coding genes (what even is short/long/medium/whatever non-coding RNA?), include "gene_biotype" as an attribute in your biomaRt query. You can use this to filter for stuff that actually becomes proteins.

Whatever is left over after this is usually manageable through manual inspection.

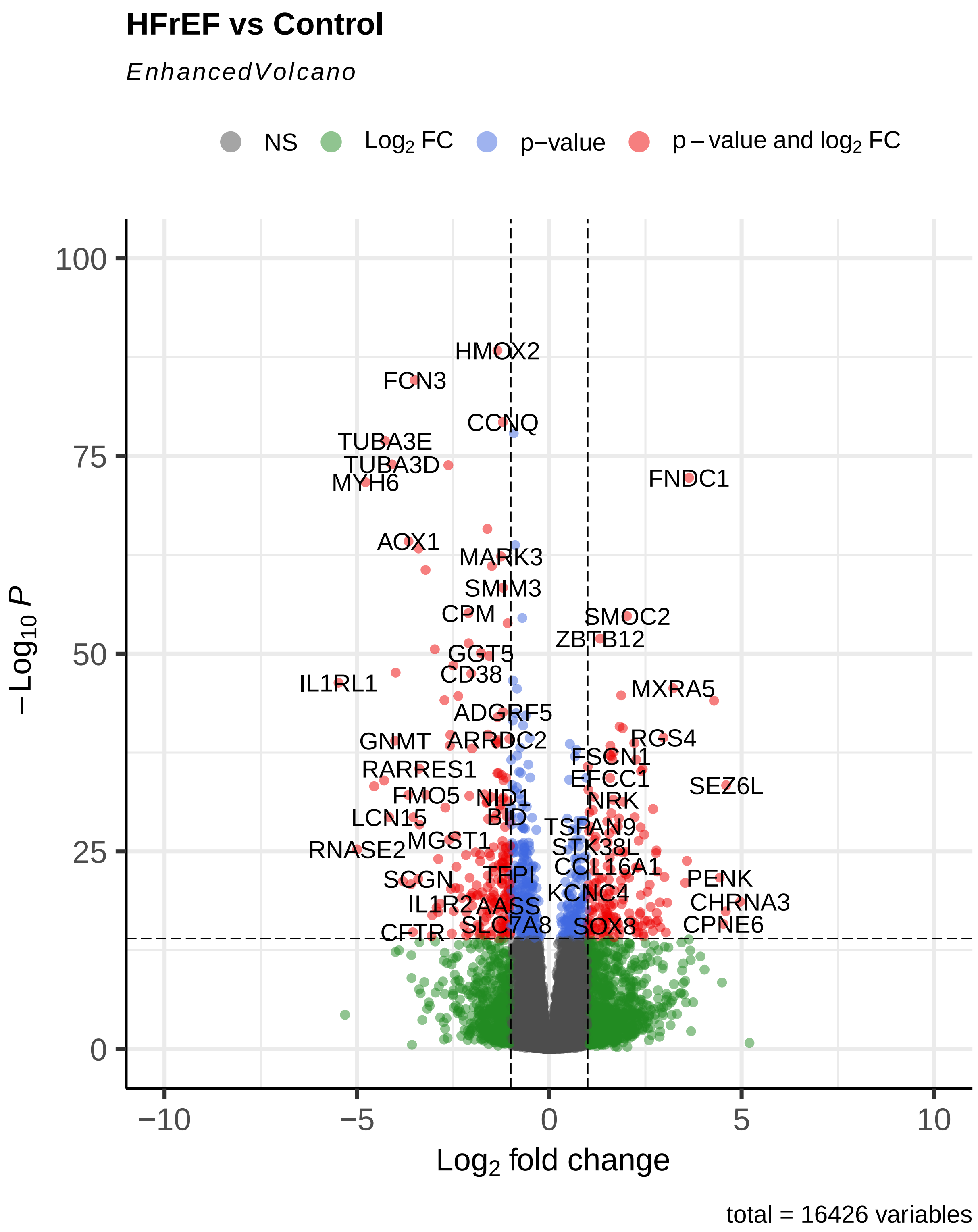
```
mart <- useEnsembl(biomart = 'genes', dataset = 'hsapiens_gene_ensembl', version = 92)
map <- getBM(filters= "ensembl_gene_id", attributes= c("ensembl_gene_id", "external_gene_name", "gene_bi

#Take only things that are protein coding and unique
map <- filter(map, gene_biotype %in% c('protein_coding'))
map <- map[isUnique(map$external_gene_name), c('external_gene_name', 'ensembl_gene_id')]



finalDEGs <- merge(resDES, map, by.x = 'geneID', by.y = "ensembl_gene_id")
finalCounts <- merge(adjCnts, map, by.x = 'row.names', by.y = "ensembl_gene_id")

#Save counts and DEGs, for use elsewhere
setwd("~/Bioinformatics/Network Analysis of HFpEF/Hahn_FGN_Analysis/")

if (rgCovs){
  write.csv(finalDEGs, "data/cov_adj_DEGs_deseq.csv")
  write.csv(finalCounts, "data/cov_adj_normCounts_deseq.csv")
} else {
  write.csv(finalDEGs, "data/DEGs_deseq.csv")
  write.csv(finalCounts, "data/normCounts_deseq.csv")
}
```

Volcano plots to highlight highly differntially expressed genes

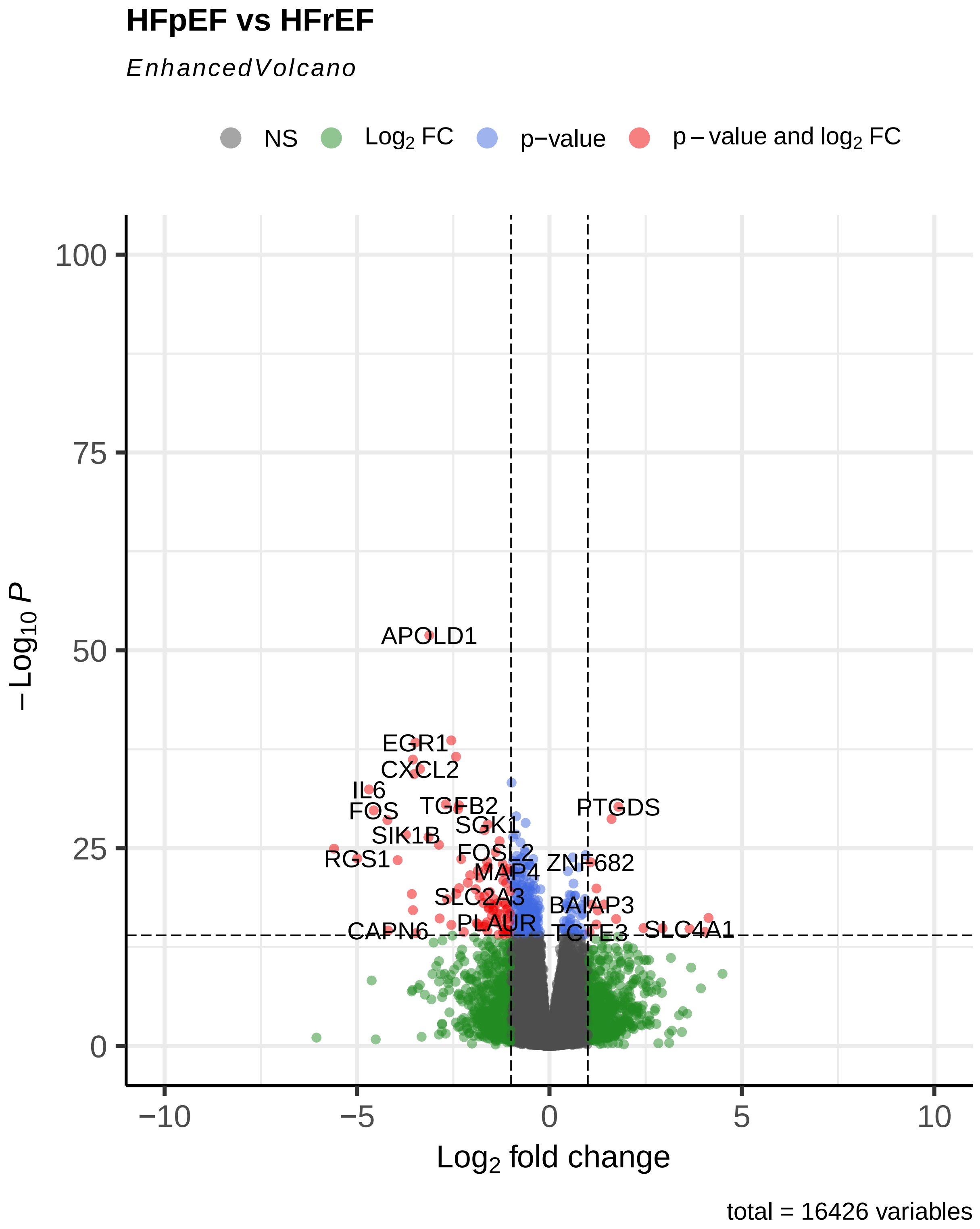
```
#HFpEF vs control
EnhancedVolcano(finalDEGs,
                     lab = finalDEGs$external_gene_name,
                     x = 'log2FoldChange_PEF',
                     y = 'pvalue_PEF',
                     title = 'HFpEF vs Control',
                     pCutoff = 10e-15,
                     ylim = c(0, 100),
                    xlim = c(-10, 10))
```



```
#HFrEF vs Control
EnhancedVolcano(finalDEGs,
                     lab = finalDEGs$external_gene_name,
                     x = 'log2FoldChange_REF',
                     y = 'pvalue_REF',
                     title = 'HFrEF vs Control',
                     pCutoff = 10e-15,
                     ylim = c(0, 100),
                    xlim = c(-10, 10))
```



```
#HFpEF vs HFrEF
EnhancedVolcano(finalDEGs,
                    lab = finalDEGs$external_gene_name,
                    x = 'log2FoldChange_PEFREF',
                    y = 'pvalue_PEFREF',
                    title = 'HFpEF vs HFrEF',
                    pCutoff = 10e-15,
                    ylim = c(0, 100),
                   xlim = c(-10, 10))
```


HFpEF vs HFrEF
EnhancedVolcano
NS
Log$_2$ FC
p−value
p – value and log$_2$ FC
100
75
50
25
0
−Log$_{10}$ P
−10
−5
0
5
10
Log$_2$ fold change
APOLD1
EGR1
CXCL2
IL6
FOS
TGFB2
PTGDS
SGK1
SIK1B
RGS1
FOSL2
ZNF682
MAP4
SLC2A3
BAIAP3
PLAUR
CAPN6
TCTE3
SLC4A1
total = 16426 variables