Heart Failure Cohort Transcription Factor Enchriment

Aaron Troy

2023-05-01

Transcription Factor Enrichment for Individual Patients Using VIPER and prior knowledge of transcription factor regulatory relationships The purpose of this notebook is to leverage prior knowledge on transcription factor –> target gene relatioships to rank which TFs are most likley implicated in observed expression profiles. Here, this is done in a groupwise fashion, rather than sample-by-sample.

To do this, we require the following:

- Expression signatures associated with one or more conditions of interest. These can be simply normalized counts or the results of DEGs analysis, as in the case here. We will use the DESeq results with 3 paired contrasts between control, HFpEF, and HFrEF
- A prior knowledge network of TF–>gene relationships. These should be signed and included some confidence score. May databases exist for this. Here, we will use Dorothea (https://saezlab.github.io/dorothea/) with a threshold applied to select only high confidence interactions]
- A statistical method for computing an enrichment score. For this, we use VIPER (https://bioconductor.org/packages/release/bioc/html/viper.html)

Start by importing the required libraries

```
knitr::opts_chunk$set(echo = TRUE)

options(warn=-1)

suppressMessages(library(dplyr))
suppressMessages(library(ggplot2))
suppressMessages(library(tidyverse))
suppressMessages(library(biomaRt))
suppressMessages(library(modelr))
suppressMessages(library(poolr))
suppressMessages(library(pheatmap))
suppressMessages(library(lsa))
suppressMessages(library(DESeq2))
suppressMessages(library(GGally))

#Dorothea - transcription factor network
library(dorothea)

#VIPER - for inferring active TFs
library(viper)

#Human Protein Atlas
library(hpar)
```

```
## This is hpar version 1.38.0,
## based on the Human Protein Atlas
##   Version: 21.0
##   Release data: 2021.11.18
##   Ensembl build: 103.38
## See '?hpar' or 'vignette('hpar')' for details.
```

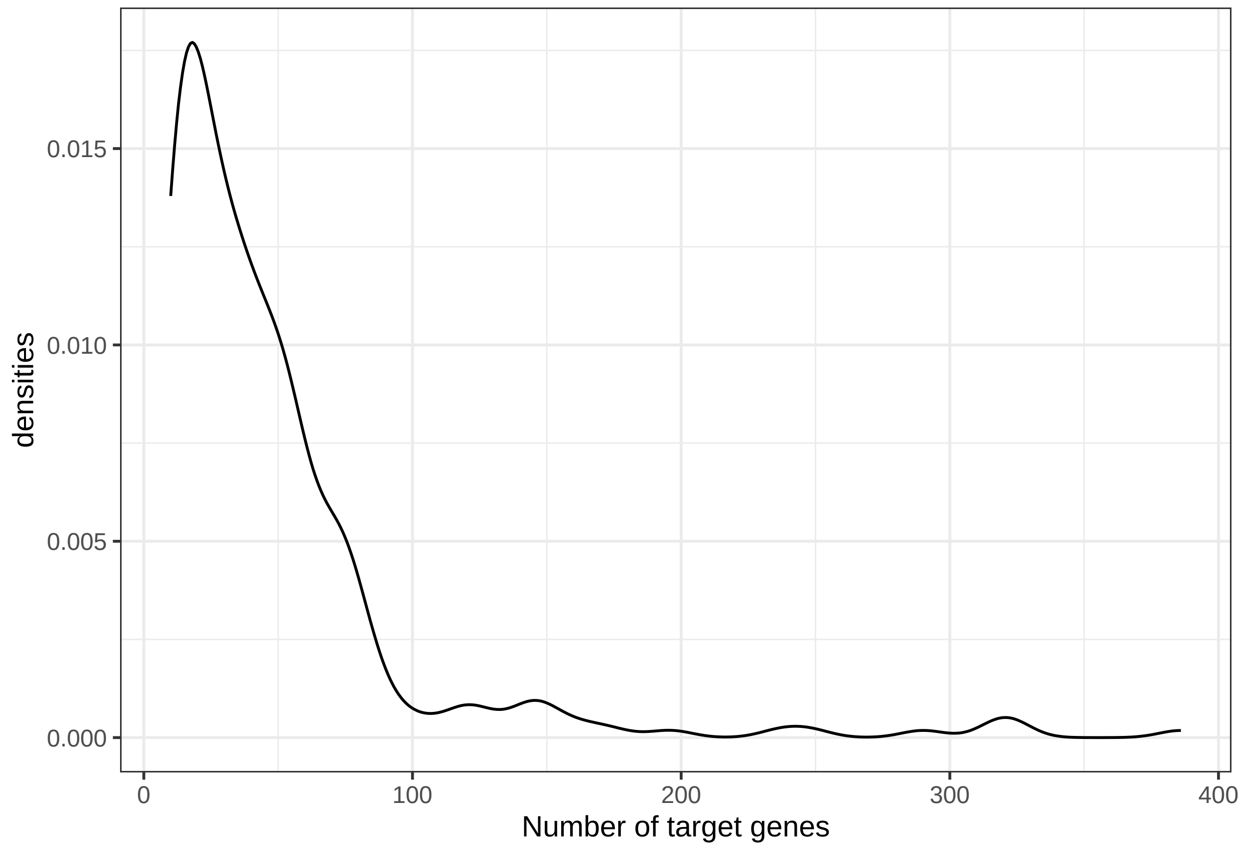
```
#Metric to perform enrichment on. {'z_score', 'log2FoldChange', 'stat'}
metric = 'z_score'
```

Retrieve Transcription Factor Network

Here we select only links with A, B, or C grade confidence. See doRothea documentation to see how these categories are defined.

Also take a peak at the distribution of regulon sizes after threshold. You can see most TFs have less than 100 targets

```
dorotheaPKN <- dorothea::dorothea_hs %>%
  dplyr::filter(confidence %in% c("A", "B", "C")) %>%
  dplyr::select(target, tf, mor) %>%
  as.data.frame()

n_genes <- dorotheaPKN %>%
  group_by(tf) %>%
  summarize(n = n())

ggplot(data=n_genes, aes(x=n)) +
  geom_density() +
  theme(text = element_text(size=12)) +
  xlab('Number of target genes') +
  ylab('densities') +
  theme_bw() +
  theme(legend.position = "none")
```



```
#VIPER requires the PKN as a regulon object. Make this using dorothea
regulon <- df2regulon(dorotheaPKN)
```

Filter TFs using the Human Protein Atlas

```
data(hpaNormalTissue)

# Querying the HPA requires ensembl IDs. Retrieve these for the TFs in the PKN using biomaRt
mart <- useEnsembl(biomart = 'genes', dataset = 'hsapiens_gene_ensembl', version = 92)
map <- getBM(filters= "external_gene_name", attributes= c("ensembl_gene_id", "external_gene_name", "gene

# Filter for presense in the heart
heartTFs <- getHpa(map$ensembl_gene_id) %>%
  filter(Tissue == "heart muscle") %>%
  filter(Level != "Not detected")
dorotheaPKN <- filter(dorotheaPKN, tf %in% heartTFs$Gene.name)
```

Read in Data

Read the DE results from DESeq2

```r
setwd('~/Bioinformatics/Network Analysis of HFpEF/Hahn_FGN_Analysis/data')

# Sources for DEG and counts data for TF enrichement
DEG_src = "cov_adj_DEGs_deseq.csv"
cnts_src = "cov_adj_normCounts_deseq.csv"

# DEGs
degs <- read.csv(DEG_src, check.names = FALSE, sep = ',', row.names = 1)

# Counts
cnts <- read.csv(cnts_src, check.names = FALSE, sep = ',', row.names = 1)%>%
  as.tibble()%>%
  column_to_rownames('external_gene_name') %>%
  dplyr::select(!c('Row.names'))

#Phenotype data
pData <- read.csv("clinicalData.csv", check.names=FALSE) %>%
  filter(id %in% colnames(cnts)) %>%
  arrange(match(id, (colnames(cnts)))) %>%
  column_to_rownames("id")

head(pData)
```

```
##      tissue disease age    sex t2dm      bmi eGFR
## 7468    RVS  Normal  34   Male   No 30.80000   65
## 7469    RVS  Normal  72   Male   No 40.00000   55
## 7470    RVS  Normal  58   Male   No 30.80000   83
## 7472    RVS  Normal  55   Male  Yes 32.98792  113
## 7473    RVS  Normal  28   Male   No 24.02381    5
## 7475    RVS  Normal  27 Female   No 29.01745   36
```

```r
#Assemble as expression set object for sample_by_sample analysis
exp_dset <- ExpressionSet(assayData = as.matrix(cnts),
                          phenoData = AnnotatedDataFrame(pData))
```

msVIPER for TF Enrichment

msVIPER works by computing a three tailed enrichment score. The routine can be summarized through the following steps:

- Genes are ranked by the absolute value of the input statistic. Many metrics are acceptable, including z-score, $\log_2$ fold change, or t-statistic. The resulting rankings are then quantile transformed
- A one tail test is applied to compute the first enrichment score ES_k^1 for each regulator k:

$$ES_k^1 = \frac{1}{N_k}\sum_{i=1}^{N_k}(1-|s_i|)l_i R(|x_i|)$$

Here, N_k is the number of target gene for regulator k, s_i the mode of interaction (activation or inhibition), l_i is the likelihood of the interaction, and $R(|x_i|)$ is the quantile transformed rank of the absolute value of statistic x_i. For us, this term always goes to zero, as $s_i \in \{-1, 1\}$ for the dorothea network

- A second enrichment score ES_k^2 accounting for the mode of the interaction is then computed using:

$$ES_k^2 = \frac{1}{N_k}\sum_{i=1}^{N_k} l_i R(x_i)$$

- Enrichment scores are then combined:

$$ES_k = (ES_k^1 + |ES_k^2|) * \text{sign}(ES_k^2)$$

- A final normalized enrichment score (NES) is then computed:

$$NES_k = ES\sqrt{\sum_{i=1}^{N_k} l_i^2}$$

By default, VIPER assumes a uniform distribution of expression values as a prior. In some ways this is poor choice, as genes are heavily co-regulated in any circumstance and thus this choice will result in p-value under estimation. An alternative you might consider is permuting samples randomly, but in this case we only have 3 groups.

TODO: think more about an appropriate prior. For the time being, let's use the null model based on permuting the gene rather than the samples.

```
#Build an expression matrix using results from DESeq
expMatrix <- dplyr::select(degs, c(paste0(metric, c('_PEF', '_REF', '_PEFREF'))))
row.names(expMatrix) <- degs$external_gene_name

#Generate a null model
perms <- permute(as.data.frame(t(expMatrix)),  1000)$perm
nullModel <- as.data.frame(t(perms))

#Minimum number of target genes required to consider a regulon
minRegSize = 10

#Perform msVIPER for all three contrast using the desired metric (l2FC, z-score, or Wald statistic) A f
# - minsize = minimum number of gene in a regulon for it to be included in the analysis
# - pleiotropy = if true, the NES for each regulon is penalized for a regulon's overlap with others

# Follow msVIPER with shawdow and leading edge anlysis

resPEF <- msviper(as.matrix(expMatrix[paste0(metric, '_PEF')]), regulon, minsize = minRegSize, pleiotrop
  shadow() %>% ledge()
```

```
##
## Performing shadow analysis on 14 regulon pairs by permutation testing.

## Process started at Mon May  1 11:46:14 2023

##   |                                                                      |

## Computing regulon enrichment with aREA algorithm

##   |                                                                      |
resREF <-  msviper(as.matrix(expMatrix[paste0(metric, '_REF')]), regulon, minsize = minRegSize, pleiotr
  shadow() %>% ledge()

resPEFREF <-  msviper(as.matrix(expMatrix[paste0(metric, '_PEFREF')]), regulon, minsize = minRegSize, p
  shadow() %>% ledge()
```

```
##
## Performing shadow analysis on 40 regulon pairs by permutation testing.

## Process started at Mon May  1 11:46:17 2023

##   |                                                                                |
```

```
#Build summary dataframes
resREF_sum <- data.frame('score' = resREF$es$nes, 'name' = names(resREF$es$nes), 'p_value' = resREF$es$
resPEF_sum <- data.frame('score' = resPEF$es$nes, 'name' = names(resPEF$es$nes), 'p_value' = resPEF$es$
resPEFREF_sum <- data.frame('score' = resPEFREF$es$nes, 'name' = names(resPEFREF$es$nes), 'p_value' = r

#Build a summary table
resAll_sum <-  inner_join(resPEF_sum, resREF_sum, by = 'name', suffix = c('_PEF', '_REF')) %>%
  inner_join(resPEFREF_sum, by = 'name') %>%
  dplyr::rename(score_PEFREF = score, p_value_PEFREF = p_value) %>%
  as.tibble() %>%
  add_column('PEF_cos_sim' = NA, 'REF_cos_sim' = NA, 'nes_composite_PEF' = NA, 'nes_composite_REF' = NA


regMat <- filter(dorotheaPKN, tf %in% resAll_sum$name) %>%
  dplyr::rename(name = target) %>%
  inner_join(rownames_to_column(expMatrix, 'name'), by = 'name')
```

Compute Cosine Composite Enrichment Score

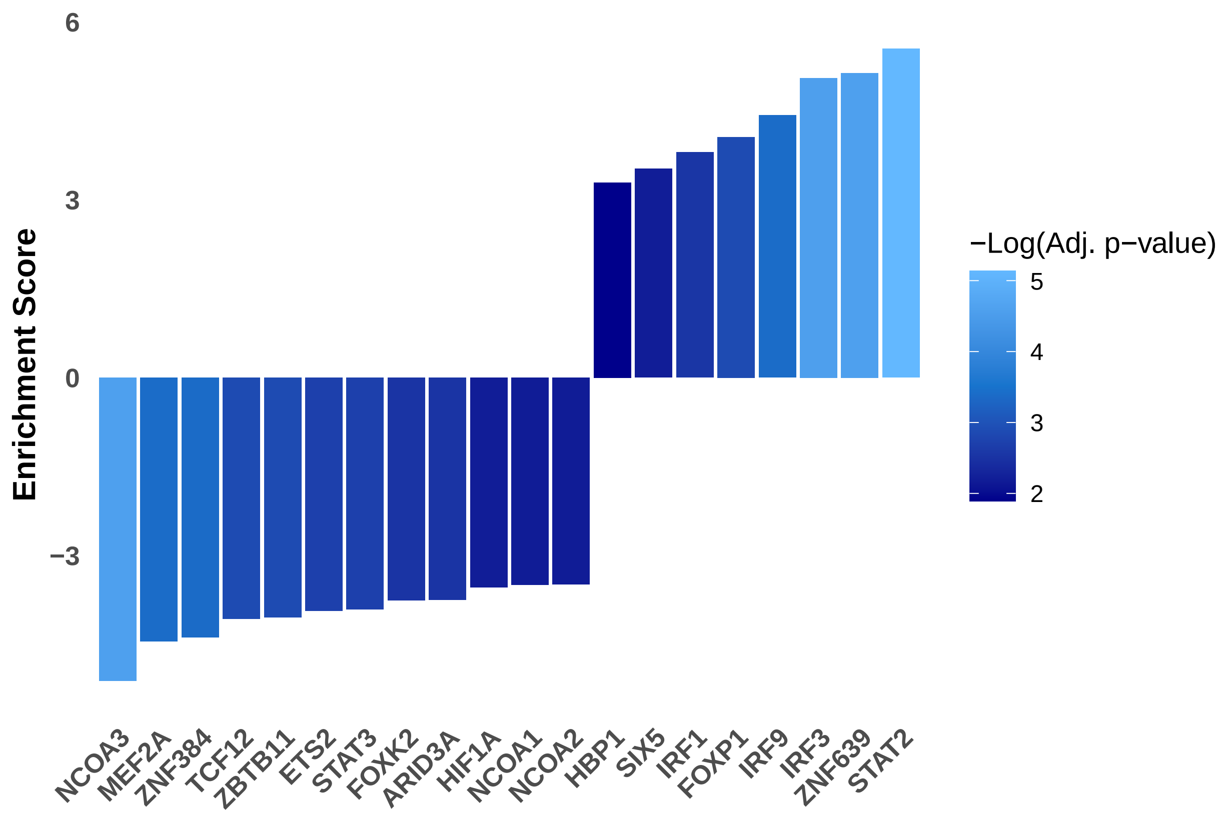
To combine information between different contrasts and produce a composite enrichement score (CES), we will use the cosine similarity. Essentially we are accounting for how similar / different the enrichment of a given regulon is as opposed to adding the ES naively.

$$CES_i = \left| \frac{\vec{r}_{(i,PvC)} \cdot \vec{r}_{(i,PvR)}}{||\vec{r}_{(i,PvC)}|| \; ||\vec{r}_{(i,PvR)}||} \right| \left(\frac{NES_{(i,PvC)} + NES_{(i,PvR)}}{2} \right)$$

```
for (i in (1:length(resAll_sum$name))){

  reg = resAll_sum$name[i]

  Ep = resAll_sum$score_PEF[i]
  Er = resAll_sum$score_REF[i]
  Epr = resAll_sum$score_PEFREF[i]

  pEF_id = paste0(metric, '_PEF')
  rEF_id = paste0(metric, '_REF')
  pEF_rEF_id = paste0(metric, '_PEFREF')
  cos_sim <- filter(regMat, tf == reg) %>%
    dplyr::select(pEF_id, rEF_id, pEF_rEF_id) %>%
    as.matrix() %>%
    cosine() %>%
    abs()

  resAll_sum$PEF_cos_sim[i] = cos_sim[pEF_rEF_id, pEF_id]
  resAll_sum$REF_cos_sim[i] = -cos_sim[pEF_rEF_id, rEF_id]

  resAll_sum$nes_composite_PEF[i] <- cos_sim[pEF_rEF_id, pEF_id] * ((Ep) + (Epr)) / 2
  resAll_sum$nes_composite_REF[i]  <- cos_sim[pEF_rEF_id, rEF_id] * ((Er) - (Epr)) / 2

}

#Build unique metric dataframes
resPEF_nesc <- data.frame('score' =  resAll_sum$nes_composite_PEF, 'name' = resAll_sum$name, 'p_adj' =
resREF_nesc <- data.frame('score' =  resAll_sum$nes_composite_REF, 'name' = resAll_sum$name)
```

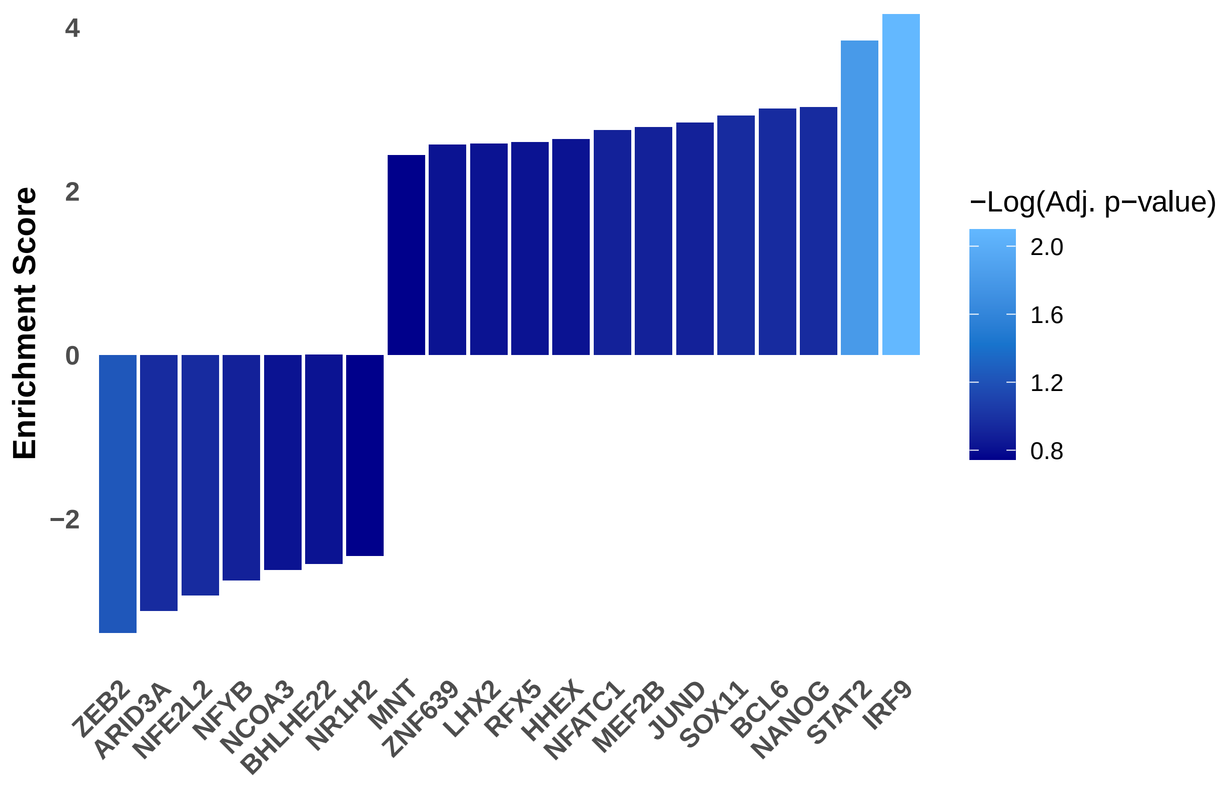
Results Plotting

```
# Make a helper function for the standard top TF plot

plotTopTFs <- function (res, topN = 20, title = NULL) {
    res <- arrange(res, desc(abs(score))) %>%
    dplyr::slice(1:topN)

   if (is.null(title)){
    title <- sprintf("Top %d Enriched Transcription Factors", topN)
   } else {
    title <- paste(sprintf("Top %d Enriched Transcription Factors -", topN), title)
  }

  ggplot(dplyr::select(res, c('name', 'score', 'p_adj'))) +
      geom_bar(aes(x = reorder(name, (score)), y = score, fill = -log10(p_adj)), stat = "identity") +
    scale_fill_gradient2(name="-Log(Adj. p-value)", low = "darkblue", high = "steelblue1",
        mid = "dodgerblue3", midpoint = (max(-log10(res$p_adj)) - min(-log10(res$p_adj))) / 2 + min(-lo
    theme_minimal() +
    theme(axis.title = element_text(face = "bold", size = 12),
    axis.text.x = element_text(angle = 45, hjust = 1, size =10, face= "bold"),
    axis.text.y = element_text(size =10, face= "bold"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()) +
    xlab(title) +
    ylab('Enrichment Score')
}

#Plot for each contrast
plotTopTFs(resPEF_sum, title = 'HFpEF vs Control')
```



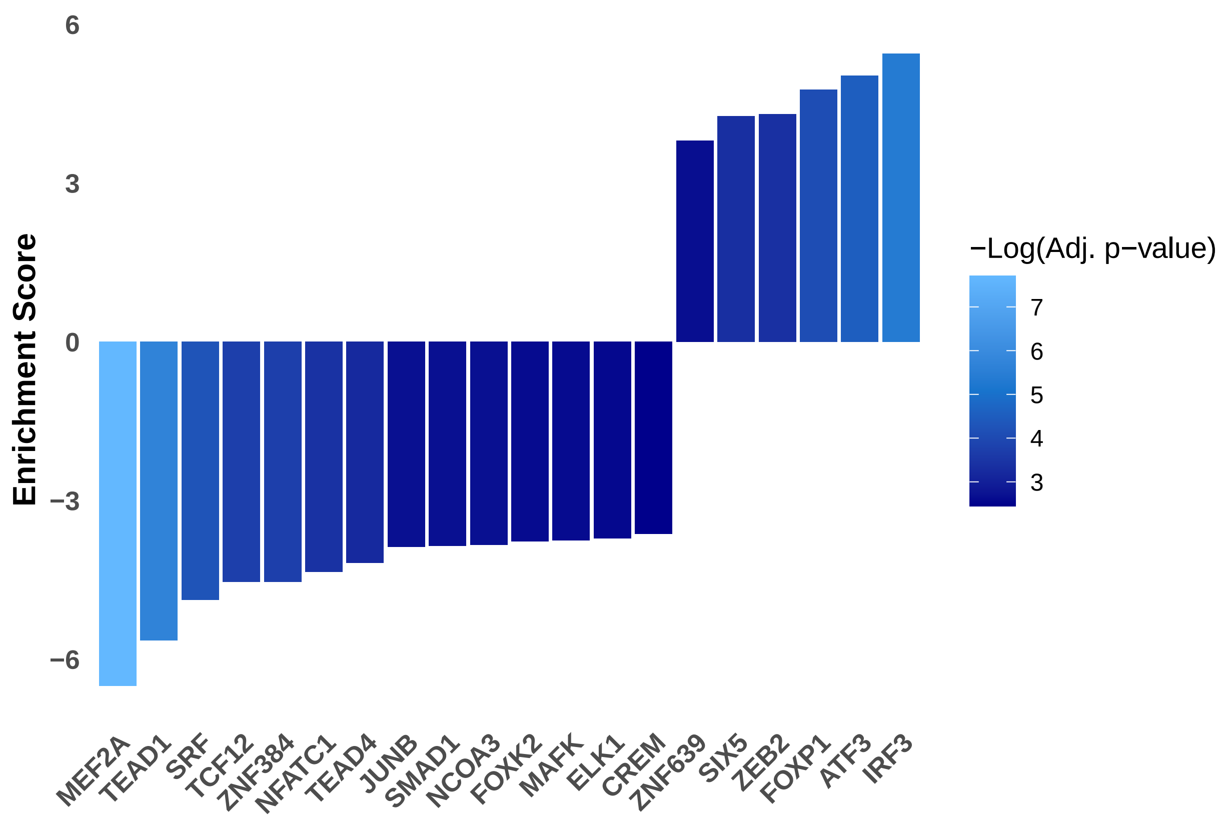
Top 20 Enriched Transcription Factors − HFpEF vs Control

```
plotTopTFs(resREF_sum, title = 'HFrEF vs Control')
```



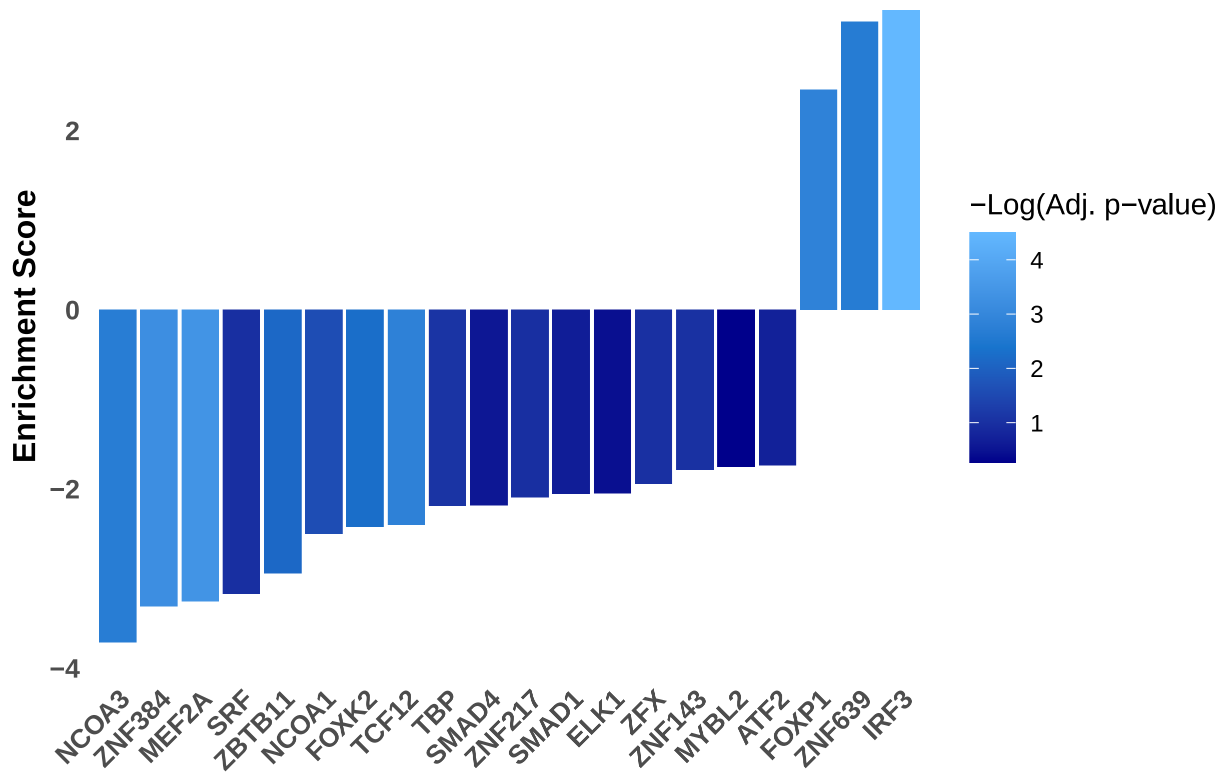
Top 20 Enriched Transcription Factors − HFrEF vs Control

```
plotTopTFs(resPEFREF_sum, title = 'HFpEF vs HFrEF')
```



Top 20 Enriched Transcription Factors − HFpEF vs HFrEF

```
#Plot for "unique" metrics
plotTopTFs(resPEF_nesc, title = 'HFpEF - Cosine Composite NES')
```



ɔ 20 Enriched Transcription Factors − HFpEF − Cosine Composite NES

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

#Update the names of the output, add z_scores
names(resAll_sum)[names(resAll_sum) == 'name'] <- 'external_gene_name'
resAll_sum$z_score_PEF = abs(qnorm(resAll_sum$p_value_PEF)) * sign(resAll_sum$score_PEF)
resAll_sum$z_score_REF = abs(qnorm(resAll_sum$p_value_REF)) * sign(resAll_sum$score_REF)
resAll_sum$z_score_PEFREF = abs(qnorm(resAll_sum$p_value_PEFREF)) * sign(resAll_sum$score_PEFREF)

#Save the results
write.csv(resAll_sum, paste0(metric, 'TF_enrichment.csv'), row.names = FALSE)
```

Samplewise Analysis

This allows us to extract correlations between TF enrichment and clinical characteristics.

```
vpsig <- viperSignature(exp_dset, 'disease', 'Normal', per = 1000)
```

```
##   |                                                                              |
```

```
vpres <- viper(vpsig, regulon, verbose = TRUE, minsize = 10)
```

```
##
## Computing the association scores
## Computing regulons enrichment with aREA
```

```
##   |                                                                      |
```

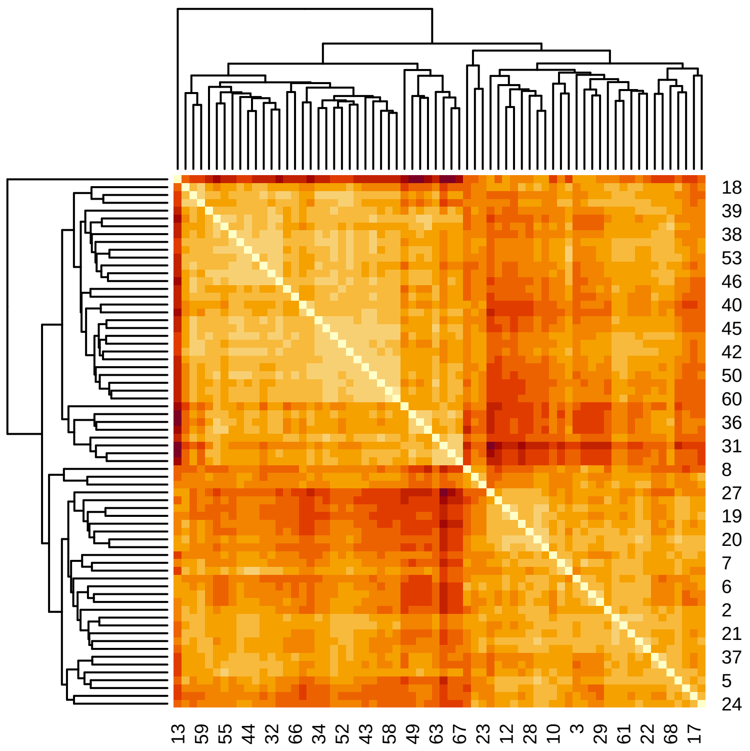
```
pos <- pData(vpres)[["disease"]] %in% c("HFpEF", "HFrEF", "Normal")
res_viper <- t(exprs(vpres)[, pos]) %>%
  as.data.frame()

#Pickup the clinical correlates to compare to TF enrichment
res_viper['Disease'] <- pData(vpres)[["disease"]]
res_viper['BMI'] <- pData(vpres)[["bmi"]]
res_viper['T2DM'] <- pData(vpres)[["diabetes"]]
res_viper['Age'] <- pData(vpres)[["age"]]
res_viper['Sex'] <- pData(vpres)[["sex"]]
res_viper['eGFR'] <- pData(vpres)[["eGFR"]]

write.csv(res_viper, 'samplewise_TF_enrichment.csv', row.names = FALSE)
```

IRF3 Activity Correlations with Clinical Characteristics

For IRF3 specifically, compute correlations for both the enrichment score and expression with clinical covariates

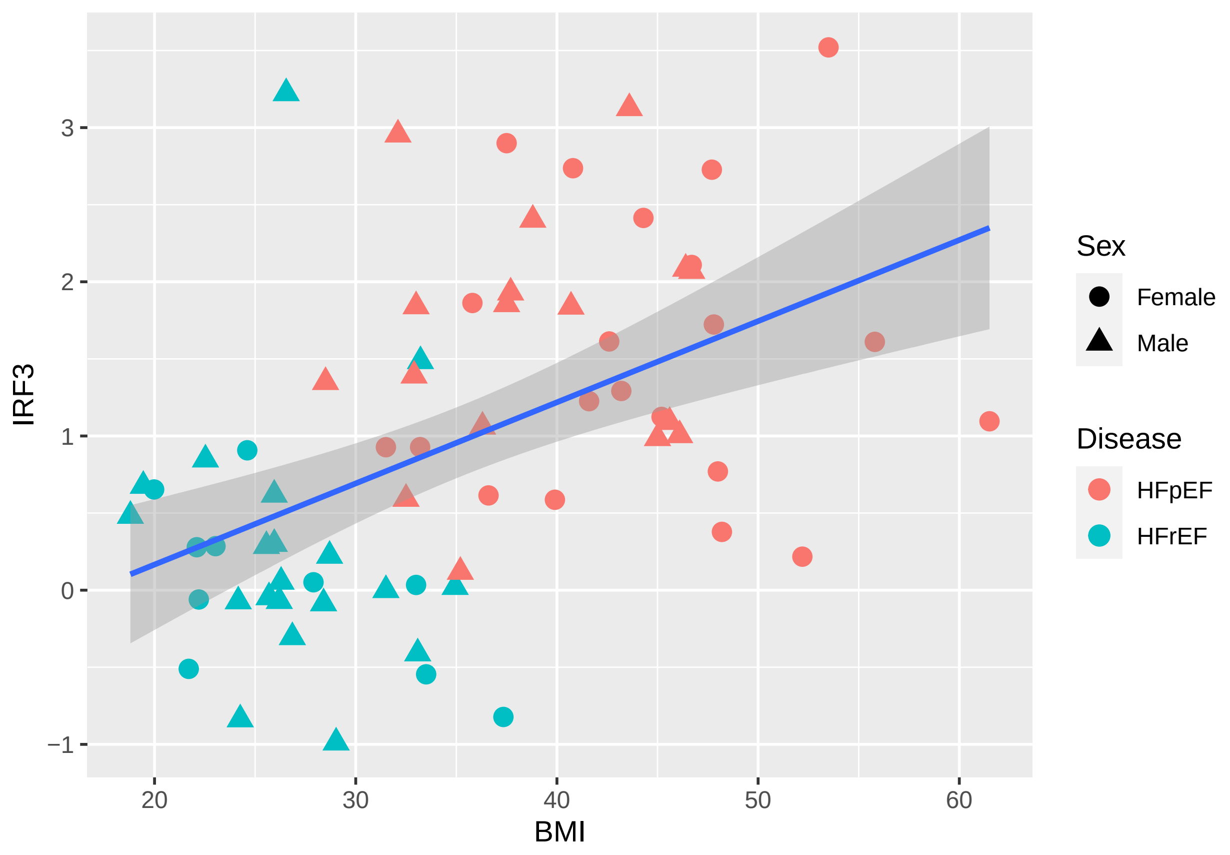
IRF3 is the top enriched TF for HFpEF using the cosine composite enrichment score. Let's examine this a bit closer by looking for correlations against given clinical covariates.

```
# Patient difference heat map
d1 <- exprs(vpres)[, pos]
colnames(d1) <- pData(vpres)[["Disease"]][pos]
dd <- dist(t(d1), method = "euclidean")
heatmap(as.matrix(dd), Rowv = as.dendrogram(hclust(dd, method = "average")), symm = T)
```

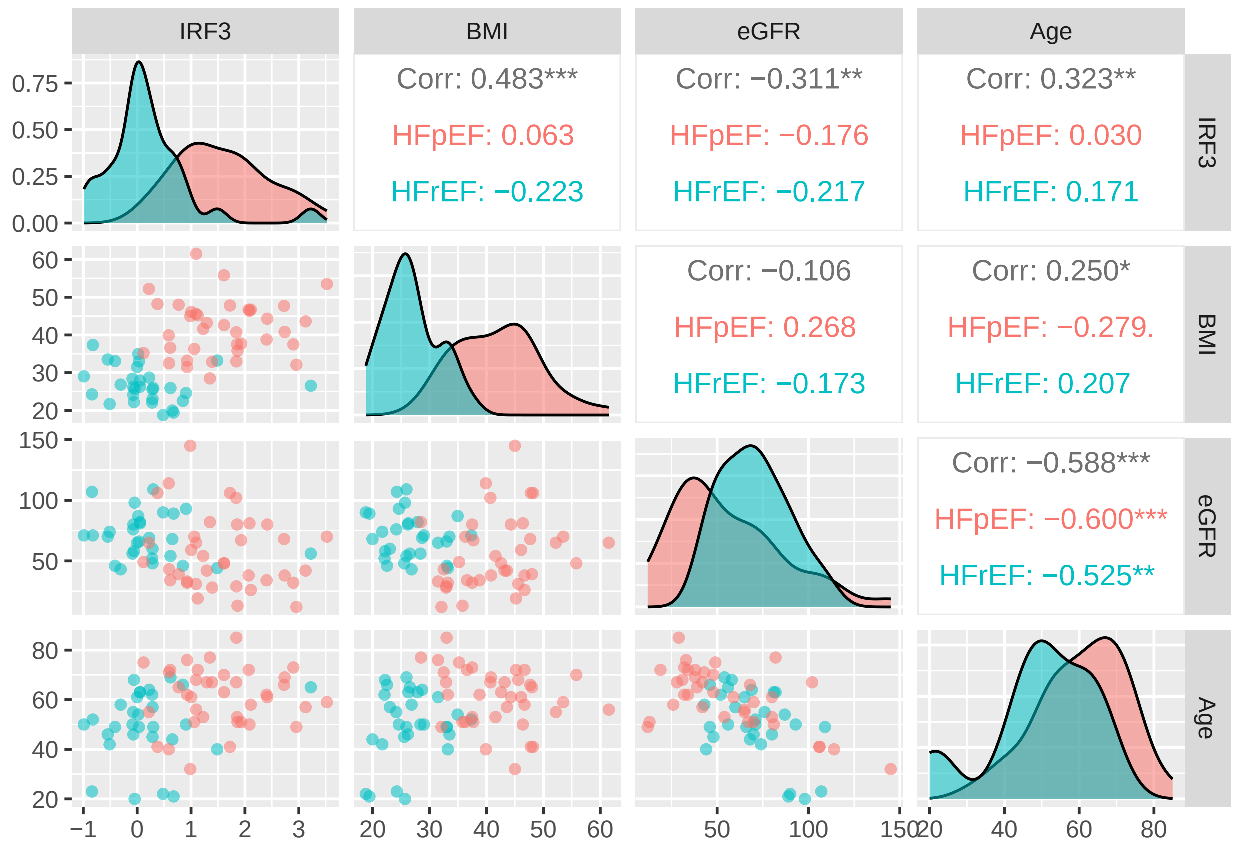


```
myColors <- c("#ffb3ba", "#ffffba")


# IRF3 - TF enrichment correlations
ggplot(res_viper, aes(x=BMI, y=IRF3)) +
  geom_point(aes(color = Disease, shape = Sex), size = 3.25) +
  geom_smooth(method= lm, fullrange = TRUE)
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

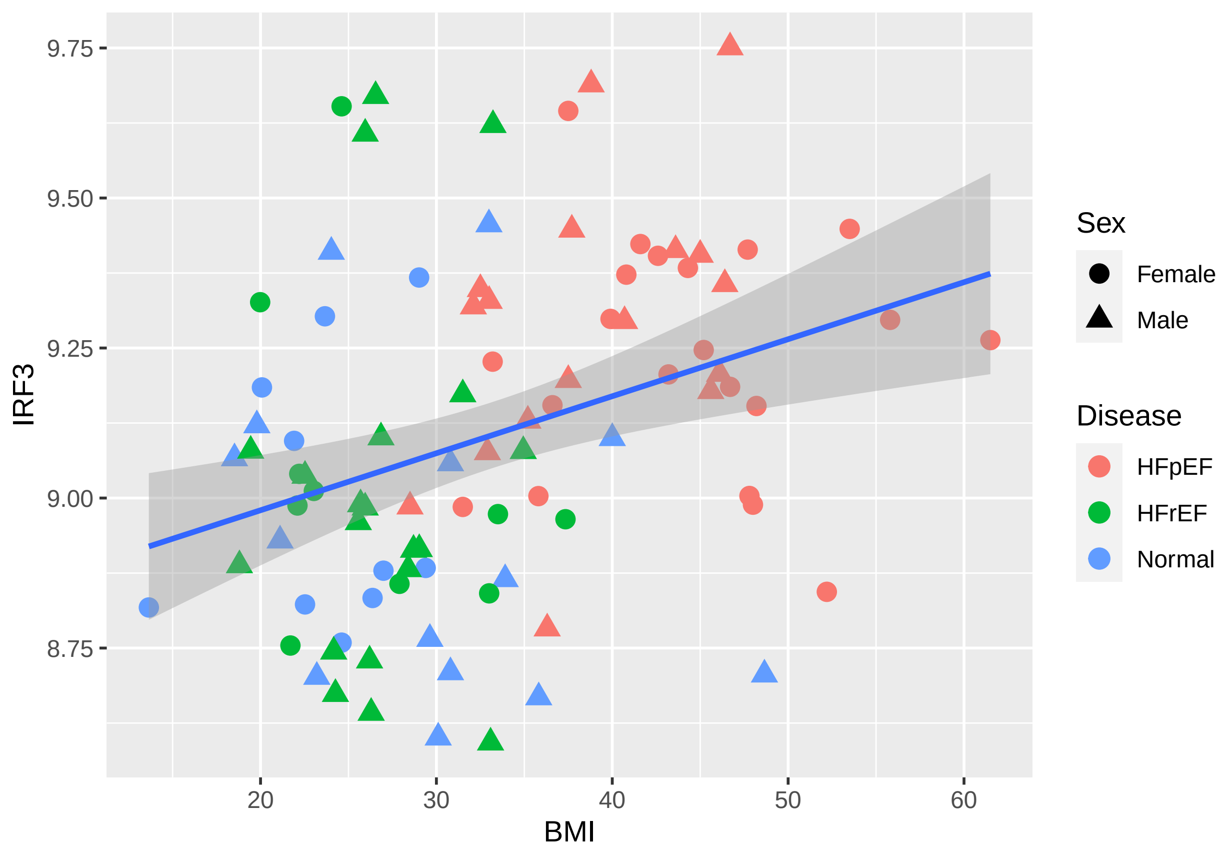


```
ggpairs(res_viper, columns = c('IRF3', 'BMI', 'eGFR', 'Age'), alpha = 1, ggplot2::aes(colour=Disease, al
```

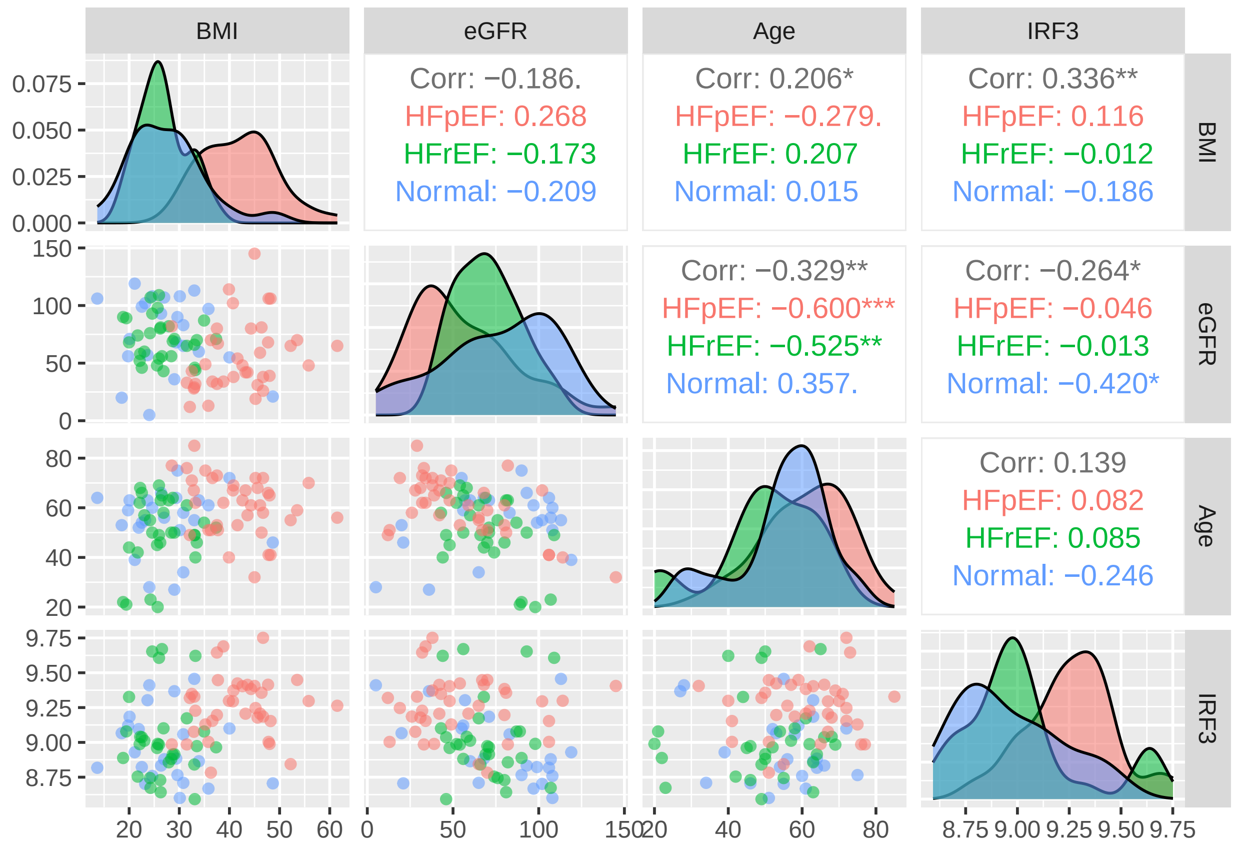


```
#IRF3 - expression correlations
cnts_plot <- t(exp_dset@assayData$exprs) %>%
  as.data.frame() %>%
  dplyr::select(IRF3)
cnts_plot['Disease'] <- exp_dset$disease
cnts_plot['BMI'] <- exp_dset$bmi
cnts_plot['T2DM'] <- exp_dset$diabetes
cnts_plot['Age'] <- exp_dset$age
cnts_plot['Sex'] <- exp_dset$sex
cnts_plot['eGFR'] <- exp_dset$eGFR

ggplot(cnts_plot, aes(x=BMI, y=IRF3)) +
  geom_point(aes(color = Disease, shape = Sex), size = 3.25) +
  geom_smooth(method= lm, fullrange = TRUE)
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



```
ggpairs(cnts_plot, columns = c('BMI', 'eGFR', 'Age','IRF3'), alpha = 1, ggplot2::aes(colour=Disease, al
```


BMI
eGFR
Age
IRF3
Corr: −0.186.
HFpEF: 0.268
HFrEF: −0.173
Normal: −0.209
Corr: 0.206*
HFpEF: −0.279.
HFrEF: 0.207
Normal: 0.015
Corr: 0.336**
HFpEF: 0.116
HFrEF: −0.012
Normal: −0.186
Corr: −0.329**
HFpEF: −0.600***
HFrEF: −0.525**
Normal: 0.357.
Corr: −0.264*
HFpEF: −0.046
HFrEF: −0.013
Normal: −0.420*
Corr: 0.139
HFpEF: 0.082
HFrEF: 0.085
Normal: −0.246