# Mapping the HDL proteome in Metabolic Syndrome through label-free quantification.

Elvira Márquez Paradas

2023-06-05

## Summaryze

This script has been performed after the differential expression analysis of the HDL proteome of healthy patients and patients with metabolic syndrome and has several objectives: - Generate correlation heat maps between differentially expressed proteins and some variables of interest. - To apply logistic regression to test whether cholesterol efflux can be a significant predictor of metabolic syndrome (MetS). - To generate a linear regression model that explains cholesterol efflux levels %(ChE) from the expression levels of some proteins of interest".

The packages required for the execution of this script can be loaded below:

```
install_and_load_packages <- function(packages) {
  missing_packages <- packages[!sapply(packages, function(p) requireNamespace(p, quietly = TRUE))]

  if (length(missing_packages) > 0) {
    install.packages(missing_packages, dependencies = TRUE)
  }

  lapply(packages, library, character.only = TRUE)

  if (length(missing_packages) > 0) {
    cat("The following packages were installed:", paste(missing_packages, collapse = ", "), "\n")
  }
  cat("Packages were loaded:", paste(packages, collapse = ", "), "\n")
}

install_and_load_packages(c("reshape2", "ggplot2", "readxl", "ggsignif", "vcd"))
```

```
## Warning: package 'reshape2' was built under R version 4.2.2

## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'readxl' was built under R version 4.2.3

## Warning: package 'ggsignif' was built under R version 4.2.2

## Warning: package 'vcd' was built under R version 4.2.3

## Loading required package: grid

## Packages were loaded: reshape2, ggplot2, readxl, ggsignif, vcd
```

## CORRELATION HEATMAPS

In this study we have data on some parameters of clinical interest in healthy patients and metabolic syndrome. To find if there is any correlation between the differentially expressed proteins in HDL of patients with metabolic syndrome and biochemical values, we will perform correlation heatmaps from Pearson's linear correlation coefficients. Next, we will load the expression data and metadata of interest and rearrange them to calculate Pearson's correlation coefficients.

```
# Uploading and organizing preprocessed protein data
#####
genes=read.table("./data/preprocessed_data_mean_imputed.txt", header = T, sep = "\t", na.strings = "NaN
gene_names=genes[,1]
gene_names[62]= "IGKV3D-20" # modify to avoid having repeated variables
gene_names[43]= "APOM_APOM"
genes=genes[,-1]
rownames(genes)=gene_names

# Load list of overexpressed proteins
#####
activated=read.table("./data/activated_genes.txt", header = T, sep = "\t",
                     na.strings = "NaN")
activated=activated[,1]
activated.genes=t(genes[activated,])

# Load clinical parameter data
#####
metadata= read_excel("./data/metadata.xlsx",sheet = 1)
patients=as.vector(metadata[,1])
patients=patients$Sujeto
metadata=metadata[,-1]
rownames(metadata)=patients
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
# Linkage of clinical data and overespressed proteins
#####
rows_order <- rownames(metadata)
activated.genes <- activated.genes[rows_order, ] # It is important to join the tables taking into accou
data <- cbind(metadata, activated.genes)
data=data[,-c(1:4)]

# Convert data to numerical
#####
for (col in names(data)) {
  data[[col]] <- as.numeric(data[[col]])
}
```

Once the table has been created, we calculate the correlation coefficients:

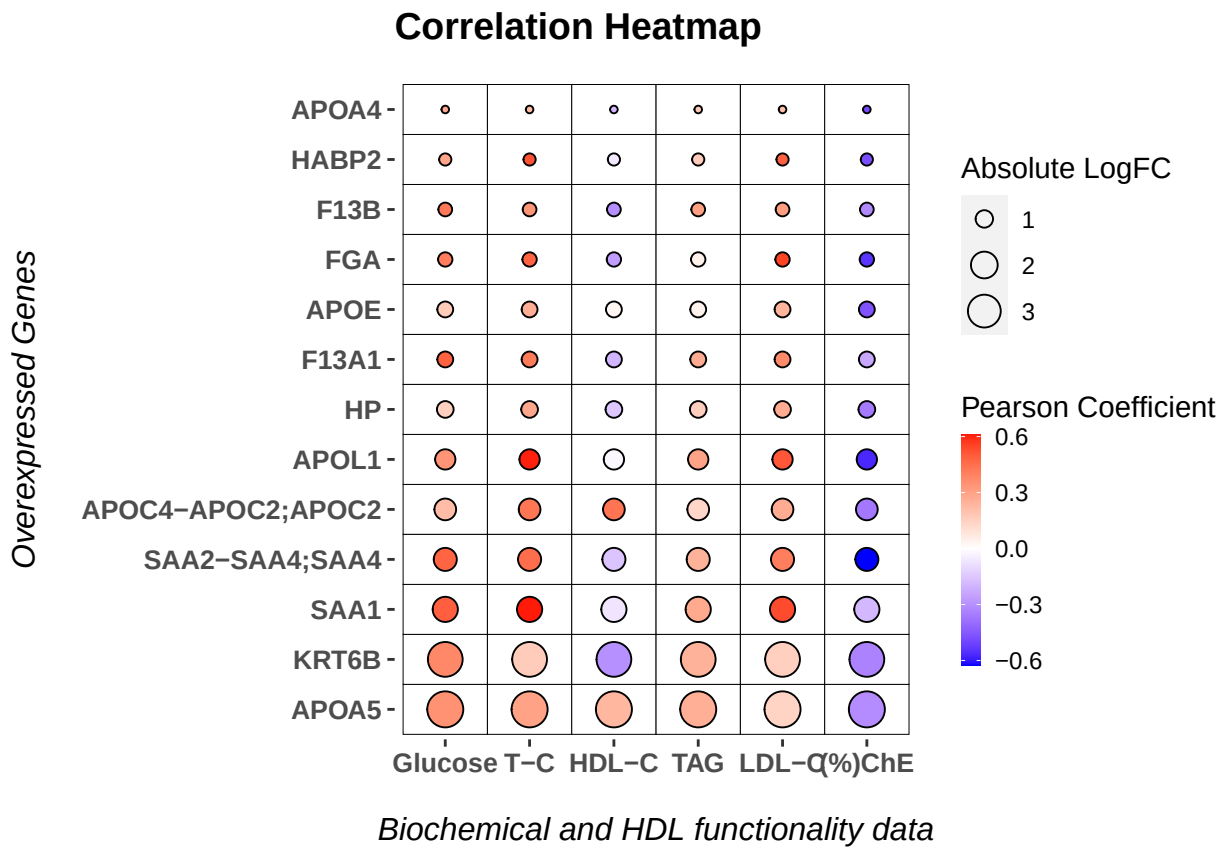
```
cor_pearson <- cor(data, method = "pearson")

# The correlation matrix will have by rows the clinical parameters and by columns the overexpressed pro
cor_pearson=cor_pearson[-c(7:19),-c(1:6)]
```

*CORRELATION HEATMAP OF OVER-EXPRESSED PROTEINS*

```
# Create the object required for the ggplot function
#####
mat_cor_melted <- melt(cor_pearson)
dim(mat_cor_melted)
```

```
## [1] 78  3
```

```
# The size of the circle will be dependent on the logFC value of the protein.
#####
fc_activated=read.table("./data/fc_activated.txt", header = T, sep = "\t",
                        na.strings = "NaN")

fc_activated=as.matrix(fc_activated)

# Merge logfc data
#####
merged_matrix <- merge(mat_cor_melted, fc_activated, by.x = "Var2", by.y = "Genes", all.x = TRUE)
merged_matrix$logFC=as.numeric(merged_matrix$logFC)
colnames(merged_matrix)[colnames(merged_matrix) == "value"] <- "Pearson.coefficient"

heatmap <- ggplot(merged_matrix, aes(x = Var1, y = Var2)) +
  geom_tile(color = "black", fill = "white") +
  geom_point(aes(size = abs(logFC), fill = Pearson.coefficient), shape = 21, color = "black") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  labs(x = "Biochemical and HDL functionality data", y = "Overexpressed Genes") +
  ggtitle("Correlation Heatmap") +
  theme(
    plot.title = element_text(face = "bold", size = 14, margin = margin(b = 12)),
    axis.text = element_text(face = "italic", size = 12),
    axis.title.x = element_text(face = "italic", size = 12, margin = margin(t = 15)),
    axis.title.y = element_text(face = "italic", size = 12, margin = margin(r = 15)),
    axis.text.x = element_text(face = "bold", size = 10),
    axis.text.y = element_text(face = "bold", size = 10),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    plot.title.position = "panel"
  ) +
  guides(fill = guide_colorbar(title = "Pearson Coefficient", barwidth = 0.5),
         size = guide_legend(title = "Absolute LogFC"))

heatmap
```




We will repeat the same steps to make the correlations between the biochemical parameters and the under-expressed proteins.

```
# Load list of under-expressed proteins
#####
repressed=read.table("./data/repressed_genes.txt", header = T, sep = "\t",
                     na.strings = "NaN")
repressed=repressed[,1]
repressed.genes=t(genes[repressed,])

# Linkage of clinical data and differentially expressed proteins
#####
rows_order <- rownames(metadata)
repressed.genes <- repressed.genes[rows_order, ]
data <- cbind(metadata, repressed.genes)
data=data[,-c(1:4)]

# Convert data to numerical
#####
for (col in names(data)) {
  data[[col]] <- as.numeric(data[[col]])
}
```

Once the table has been created, we calculate the correlation coefficients:

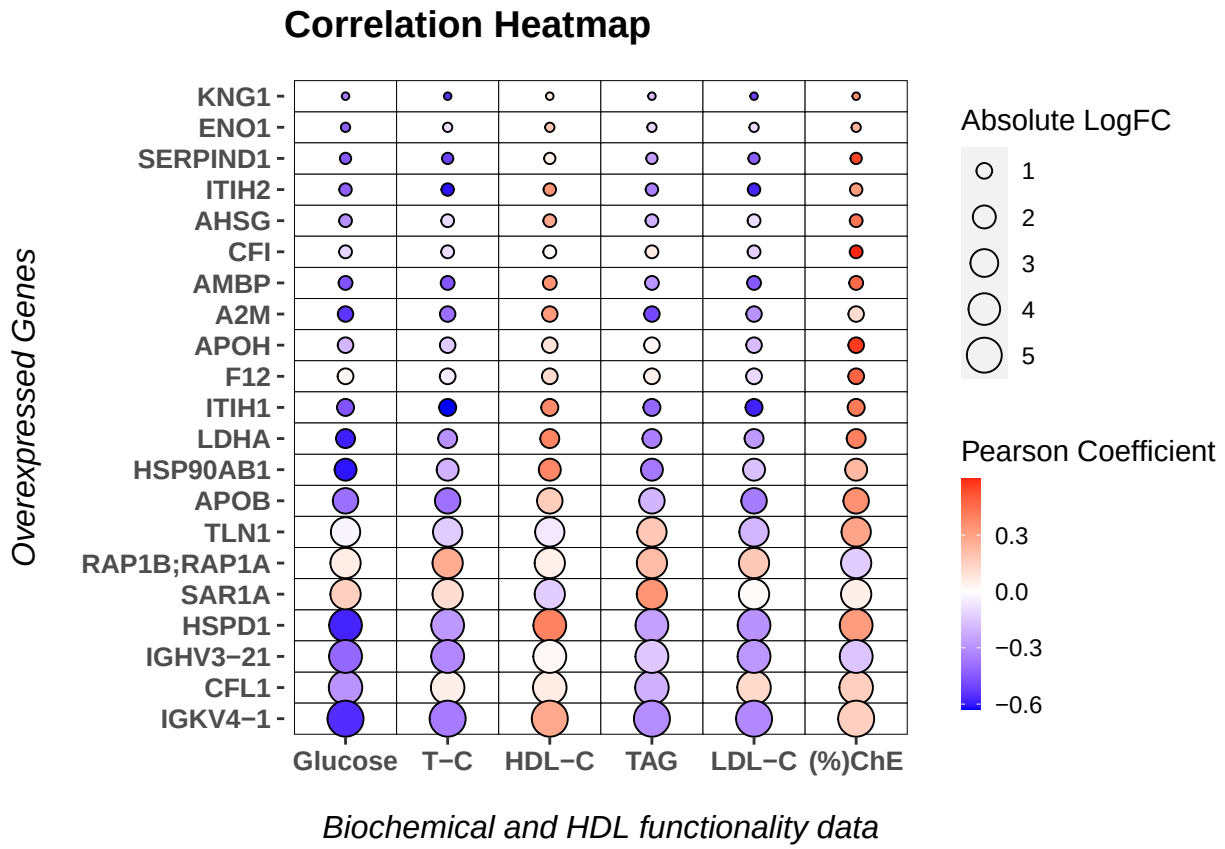
```
cor_pearson <- cor(data, method = "pearson")

# The correlation matrix will have by rows the clinical parameters and by columns the differentially
cor_pearson=cor_pearson[-c(7:27),-c(1:6)]
```

*CORRELATION HEATMAP OF UNDER-EXPRESSED PROTEINS*

```
# Create the object required for the ggplot function
#####
mat_cor_melted <- melt(cor_pearson)
dim(mat_cor_melted)
```

```
## [1] 126   3
```

```
# The size of the circle will be dependent on the logFC value of the protein.
#####
fc_repressed=read.table("./data/fc_repressed.txt", header = T, sep = "\t",
                        na.strings = "NaN")
fc_repressed=as.matrix(fc_repressed)

# Merge logfc data
#####
merged_matrix <- merge(mat_cor_melted, fc_repressed, by.x = "Var2", by.y = "Genes", all.x = TRUE)
merged_matrix$logFC=as.numeric(merged_matrix$logFC)
colnames(merged_matrix)[colnames(merged_matrix) == "value"] <- "Pearson.coefficient"

heatmap <- ggplot(merged_matrix, aes(x = Var1, y = Var2)) +
  geom_tile(color = "black", fill = "white") +
  geom_point(aes(size = abs(logFC), fill = Pearson.coefficient), shape = 21, color = "black") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  labs(x = "Biochemical and HDL functionality data", y = "Overexpressed Genes") +
  ggtitle("Correlation Heatmap") +
  theme(
    plot.title = element_text(face = "bold", size = 14, margin = margin(b = 12)),
    axis.text = element_text(face = "italic", size = 12),
    axis.title.x = element_text(face = "italic", size = 12, margin = margin(t = 15)),
    axis.title.y = element_text(face = "italic", size = 12, margin = margin(r = 15)),
    axis.text.x = element_text(face = "bold", size = 10),
    axis.text.y = element_text(face = "bold", size = 10),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    plot.title.position = "panel"
  ) +
  guides(fill = guide_colorbar(title = "Pearson Coefficient", barwidth = 0.5),
         size = guide_legend(title = "Absolute LogFC"))

heatmap
```




## SIMPLE LOGISTIC REGRESSION

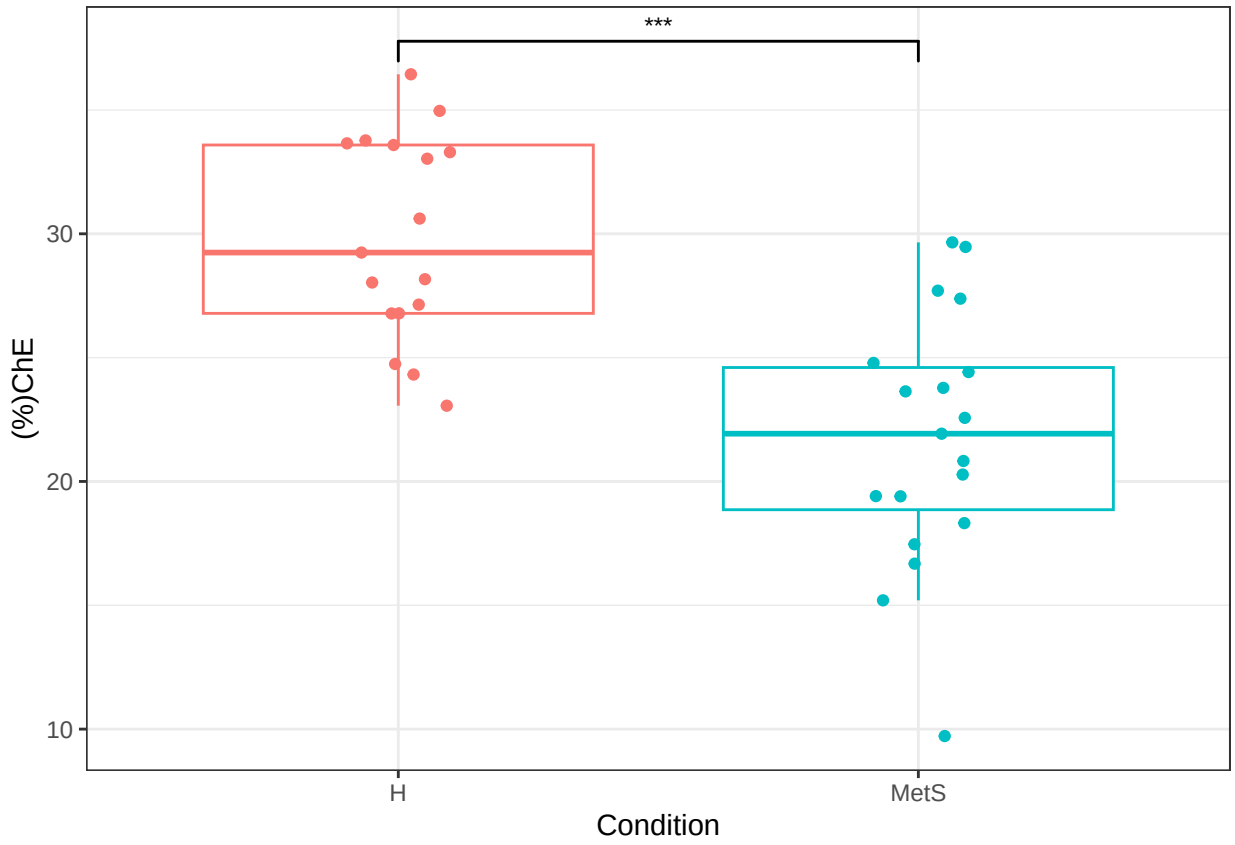
Simple Logistic Regression, (David Cox, 1958,) is a regression method that allows the estimation of the probability of a binary qualitative variable as a function of a quantitative variable. One of the main applications is binary classification, in which observations are classified in one group or another depending on the value of the variable used as predictor. We will use the cholesterol efflux data to see if it is significantly different between the two study cohorts and to see if it could be a significant predictor of MetS.

```
# Load cholesterol efflux data
#####
efflux= read_excel("./data/efflux.xlsx",sheet = 1)
efflux=efflux[,-1]
Conditions= c(rep("H", 17), rep("MetS", 19))
data=cbind(Conditions,efflux)
colnames(data)=c("Conditions", "ChE")
table(data$Conditions)
```

```
##
##    H MetS
##   17   19
```

```
# Distribution of cholesterol efflux levels in both groups
#####
```

```
ggplot(data = data, aes(x = Conditions, y = ChE, color = Conditions)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.1) +
  theme_bw() +
  theme(legend.position = "NULL") +
  labs(x = "Condition", y = "(%)ChE") +
  geom_signif(comparisons = list(c("H", "MetS")),
              test = "t.test",
              textsize = 3,
              map_signif_level = TRUE,
              colour="black")
```




Efflux levels significantly lower in patients with MetS. Next, we will generate the model.

```
# Model generation
#####
data$Conditions = ifelse(data$Conditions == "H", 0, 1)
model = glm(Conditions ~ ChE, data = data, family = "binomial")
summary(model)
```

```
##
## Call:
## glm(formula = Conditions ~ ChE, family = "binomial", data = data)
##
## Deviance Residuals:
```

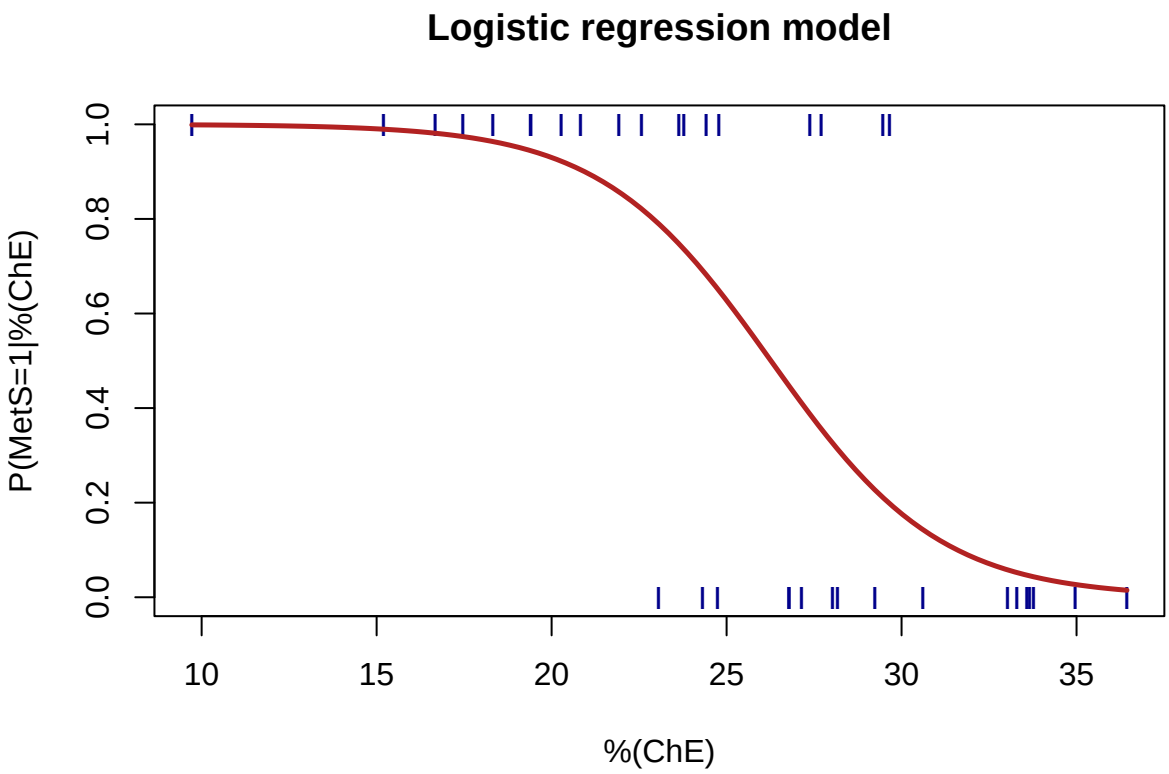
```
##      Min        1Q    Median        3Q       Max
## -1.76594  -0.59538   0.09521   0.57357   1.79882
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  10.8362     3.5903   3.018  0.00254 **
## ChE          -0.4125     0.1361  -3.032  0.00243 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 49.795  on 35  degrees of freedom
## Residual deviance: 28.575  on 34  degrees of freedom
## AIC: 32.575
##
## Number of Fisher Scoring iterations: 5
```

```
confint(object = model, level = 0.95 )
```

```
## Waiting for profiling to be done...
```

```
##                 2.5 %     97.5 %
## (Intercept)  5.103437 19.6432818
## ChE         -0.746364 -0.1954779
```

```
# Graph of the model
#####

# 0,1 coding of the response variable
data$Conditions = as.character(data$Conditions)
data$Conditions =  as.numeric(data$Conditions)

plot(Conditions ~ ChE, data, col = "darkblue",
     main = "Logistic regression model",
     ylab = "P(MetS=1|%(ChE)",
     xlab = "%(ChE)", pch = "I")

curve(predict(model, data.frame(ChE = x), type = "response"), #type = "response" returns predictions in
      col = "firebrick", lwd = 2.5, add = TRUE)
```




Once the model has been created, we will evaluate its quality by checking the significance of the difference in residuals between our model and a null model using the Likelihood ratio test.

```
# Residue difference
#####
dif = model$null.deviance - model$deviance

# Degrees of freedom
#####
df = model$df.null - model$df.residual

# p-value
####
p_value = pchisq(q = dif, df = df, lower.tail = FALSE)

paste("Residue difference:", round(dif, 4))
```

```
## [1] "Residue difference: 21.22"
```

```
paste("p-value:", p_value)
```

```
## [1] "p-value: 4.09459292174328e-06"
```

Due to the low p.value, we confirm that the generated model is significant.

For this study, we will use a threshold of 0.5, so that if the probability of the variable acquiring the value 1 (belonging to the MetS group) is higher than 0.5, it is assigned to this level; if it is lower, it is assigned to 0 (belonging to the healthy group):

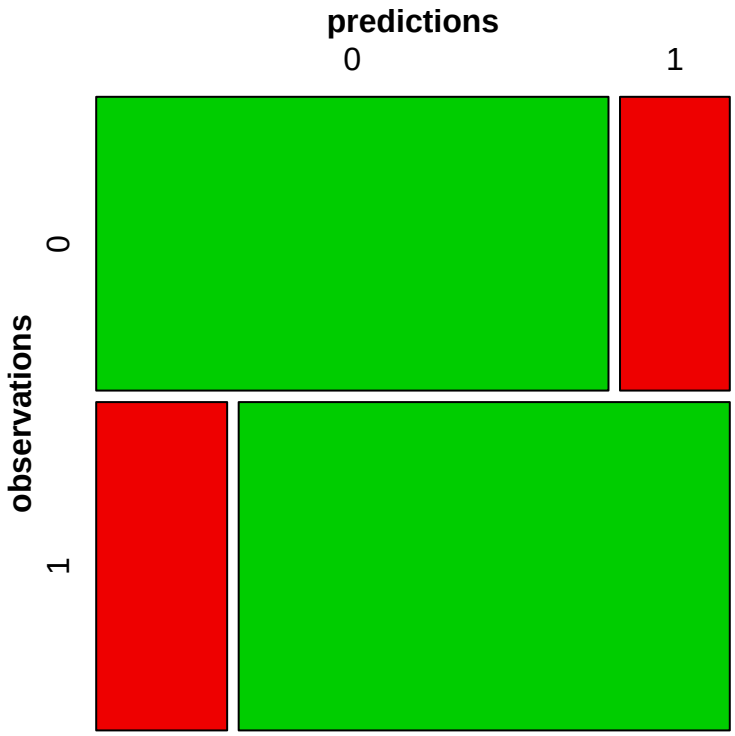
```
predictions = ifelse(test = model$fitted.values > 0.5, yes = 1, no = 0)
matrix_confusion = table(model$model$Conditions, predictions,
                         dnn = c("observations", "predictions"))
matrix_confusion
```

```
##             predictions
## observations  0  1
##            0 14  3
##            1  4 15
```

```
accuracy=(14+15)/(14+15+4+3)
accuracy
```

```
## [1] 0.8055556
```

```
mosaic(matrix_confusion, shade = T, colorize = T,
       gp = gpar(fill = matrix(c("green3", "red2", "red2", "green3"), 2, 2)))
```



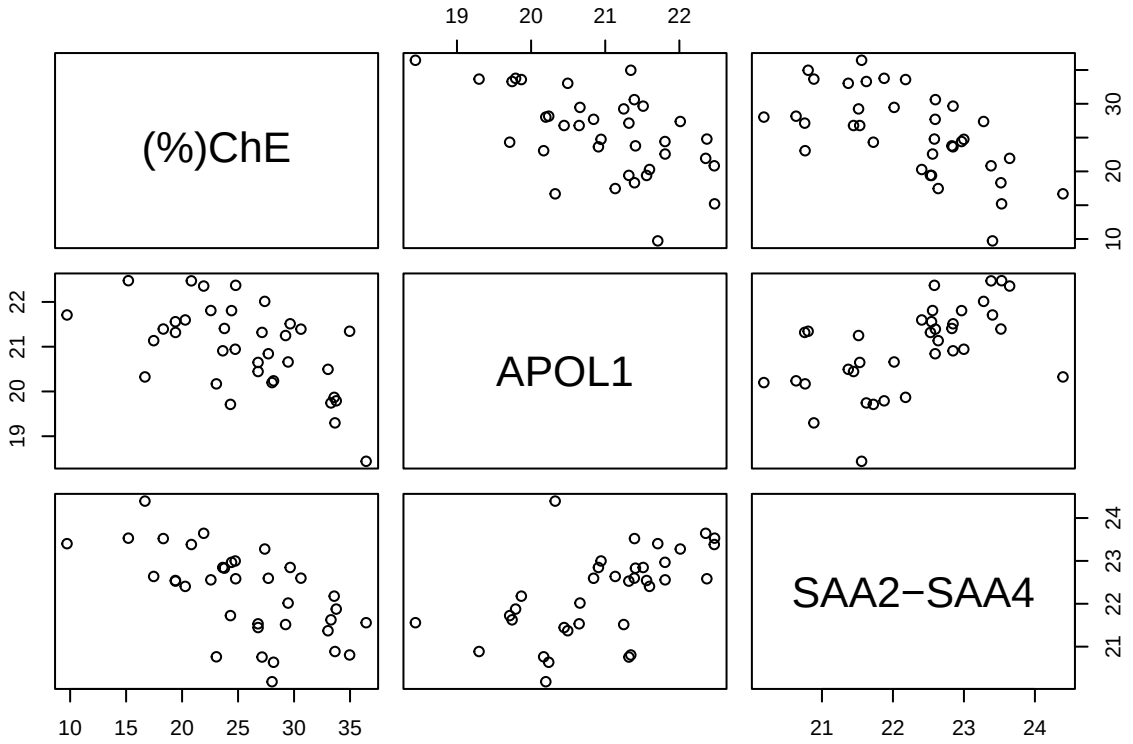

The model is able to classify 81% of the data used for training correctly. It is important to note that this is the hit rate calculated with the training data, so it is not generalisable to new observations.

## LINEAR REGRESSION MODEL

To test whether some variables could be good predictors of %(ChE) levels, i.e. could act as significant explanatory variables, the most relevant correlations observed were fitted by linear regression models. The proteins with the highest Pearson correlation coefficient with %(ChE) were SAA2- SAA4 and ApoL1.

```
# Load and reorder expression data
#####
efflux <- read_excel("./data/efflux.xlsx")
genes=read.table("./data/preprocessed_data_mean_imputed.txt", header = T, sep = "\t",
                 na.strings = "NaN")
genes=t(genes)
gene_names=genes[1,]
colnames(genes)=gene_names
genes=as.data.frame(genes[-1,])
patients=rownames(genes)
genes=cbind(patients,genes)

# Merging of data of interest in a table
#####
data <- merge(efflux, genes, by.x = "Subject", by.y = "patients")
data=cbind(data$`(%)ChE`,data$APOL1, data$`SAA2-SAA4;SAA4`)
colnames(data)=c("(%)ChE","APOL1","SAA2-SAA4")
data=as.data.frame(data)

# Convert data to numerical values
#####
for (col in names(data)) {
  data[[col]] <- as.numeric(data[[col]])
}
```

```
pairs(data)
```




```
cor_pearson <- cor(data, method = "pearson")
```

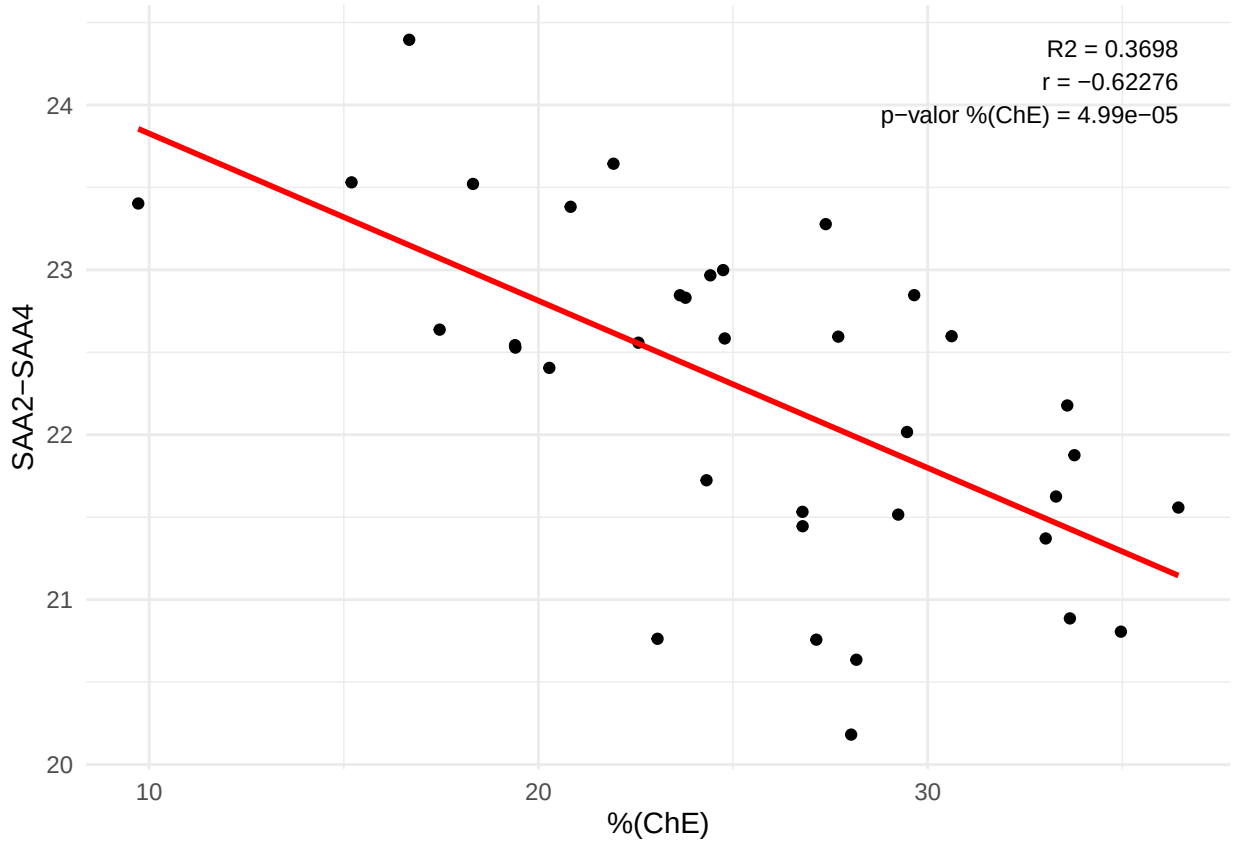
Next, we generate the model:

```
efflux=data$`(%)ChE`
apol1=data$APOL1
saa2=data$SAA2

# Model generation with SAA2-SAA4
#####
regression_1=lm(efflux ~ saa2)
summary(regression_1)
```

```
##
## Call:
## lm(formula = efflux ~ saa2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.4302  -3.7072   0.3991   3.5675   8.2370
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 110.6651    18.3547   6.029 7.89e-07 ***
## saa2         -3.8250     0.8241  -4.641 4.99e-05 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.908 on 34 degrees of freedom
## Multiple R-squared:  0.3878, Adjusted R-squared:  0.3698
## F-statistic: 21.54 on 1 and 34 DF,  p-value: 4.99e-05
```

```r
# Graphical visualization of the model
#####

ggplot(data, aes(x = efflux, y = saa2)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red") +
  labs(x = "%(ChE)", y = "SAA2-SAA4") +
  theme_minimal() +
  annotate("text", x = max(efflux), y = max(saa2),
           label = "R2 = 0.3698\nr = -0.62276\np-valor %(ChE) = 4.99e-05",
           hjust = 1, vjust = 1, color = "black", size = 3)
```




The predictor variable SAA2-SAA4 is significant for the model, although it only explains 31% of the variance of the data. To see if it improves the R2 of the model, we will add ApoL1 as an explanatory variable.

```r
# Model generation with SAA2-SAA4 and ApoL1
#####
regression_2=lm(efflux ~ saa2+apol1)
summary(regression_2)
```

```
##
## Call:
## lm(formula = efflux ~ saa2 + apol1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.170  -3.601   0.804   3.158   6.877
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 129.7298    19.8623   6.531 2.04e-07 ***
## saa2         -2.7185     0.9552  -2.846  0.00756 **
## apol1        -2.0831     1.0167  -2.049  0.04849 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.692 on 33 degrees of freedom
## Multiple R-squared:  0.4569, Adjusted R-squared:  0.424
## F-statistic: 13.88 on 2 and 33 DF,  p-value: 4.219e-05
```

R2 increases to 42.4%. this indicates that there must be more factors involved in determining cholesterol efflux levels.