Biomedical Data Science: Assignment 2

Agnieszka Słowik

2/25/2018

Problem 1

1.a)

```r
# Reading the results file and lipid class names
lipid.classes <- read.table("data_assignment2/lipid-classes.txt", sep = "\t",
    col.names = c("tag", "full.class.name"))
results <- read.table("data_assignment2/lipids.txt", sep = "\t", header = T)

# Extract the abbrevated lipid names
results$tag <- toupper(str_extract(results$lipid.species, "[aA-zZ]+"))
# Capitalise for the consistency
lipid.classes$tag <- toupper(lipid.classes$tag)

# Annotate each lipid with its corresponding full class name
results$full.class.name <- lipid.classes$full.class.name[match(results$tag,
    lipid.classes$tag)]

# The number of lipids in each class
table(results$full.class.name)
```

```
##
##                      Ceramides            Cholesterol esters
##                             13                             9
##                Diacylglycerols      Lysophosphatidylcholines
##                             16                            13
## Lysophosphatidylethanolamines         Phosphatidylcholines
##                              3                            42
##     Phosphatidylethanolamines          Phosphatidylserines
##                             25                             8
##                Sphingomyelins              Triacylglycerols
##                              0                           147
```

1.b)

```r
# Wald test statistics: Coefficients / Standard Errors
wald.statistic <- log(results$oddsratio) / results$se

N <- 288 # Number of patients
P <- 4 # Number of predictors

DF <- N - P - 1 # Degrees of freedom

# P-Values derived using t-distribution
p.value <- 2*pt(abs(wald.statistic), DF, lower=FALSE)
```

```r
results$p.value <- p.value

# P-Values derived using normal distribution
p.value.norm <- 2*pnorm(abs(wald.statistic), lower=FALSE)
```

The Wald test statistic follows the t-distribution with 283 degrees of freedom. From this we can derive a p-value, which is the probability of getting a more extreme result that what was observed. Since $t_\infty = \mathcal{N}(0,1)$ for the number of of degrees of freedom that is "high enough" the standard normal distribution can be used. In statistical tables , this critical number is usually around 100 degrees of freedom (sometimes 200). We are using a t-distribution of 283 degrees of freedom so the normal approximation is acceptable.

1.c)

```r
holm.bonferroni <- function(results, alpha){
  # Order the results by pval
  results.sorted <- results[order(p.value), ]
  results.sorted$k <- seq.int(nrow(results.sorted))
  k <- NULL # Index k
  for(i in 1:nrow(results.sorted)){
    if (results.sorted[i,]$p.value < alpha/(nrow(results)+1-results.sorted[i,]$k)){
      k <-i
    }
  }
  # The significant subset of results according to Holm-Bonferroni
  return(results.sorted[c(1:k),])
}
```

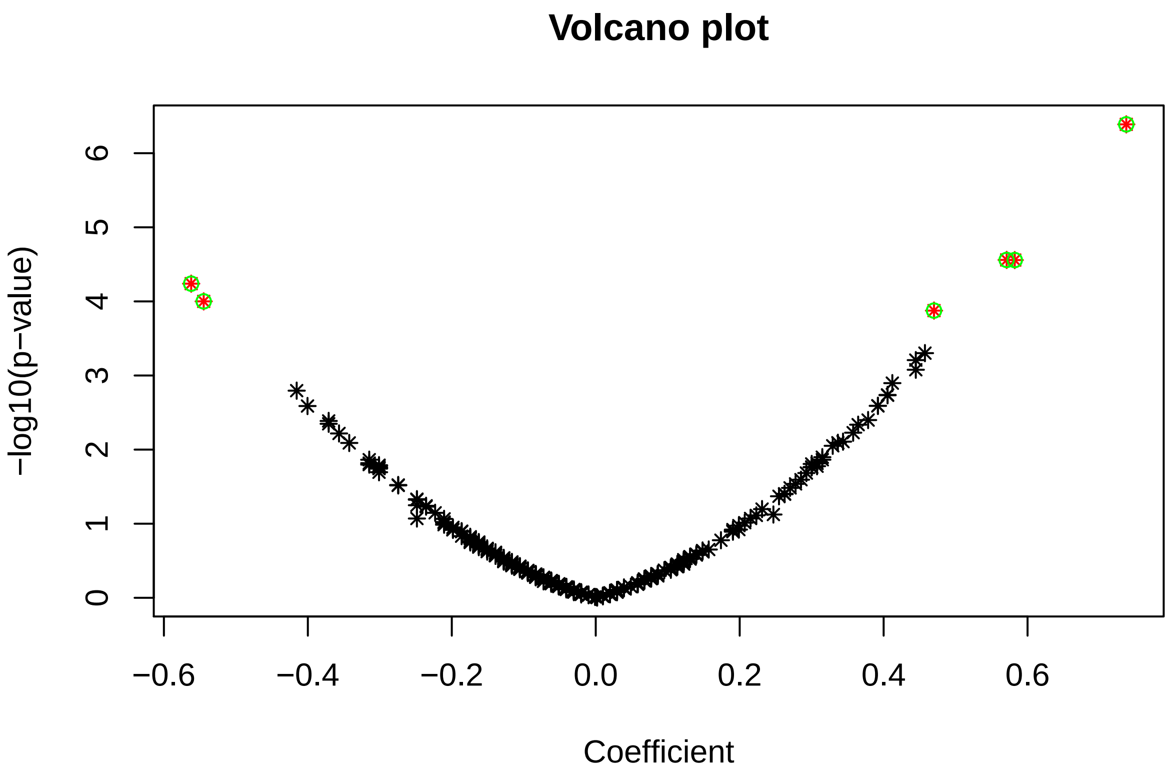
1.d)

```r
benjamini.hochberg <- function(results, q){
  # Order the results by pval
  results.sorted <- results[order(p.value), ]
  results.sorted$k <- seq.int(nrow(results.sorted))
  k <- NULL # Index k
  for(i in 1:nrow(results.sorted)){
    if (results.sorted[i,]$p.value <= (results.sorted[i,]$k/nrow(results))*q){
      k <-i
    }
  }
  # The significant subset of results according to Benjamini-Hochberg
  return(results.sorted[c(1:k),])
}
```

1.e)

```r
# Indices of species that are significant according to HB
significant.species.hb <- as.numeric(rownames(holm.bonferroni(results, 0.05)))
# Indices of species that are significant according to BH
significant.species.bh <- as.numeric(rownames(benjamini.hochberg(results, 0.01)))
```

```r
# Additional column to make the plotting easier
results$color <- rep(0, length(results[,1]))

results$color[significant.species.hb] <- 2 # Red: HB significant
results$color[-significant.species.hb] <- 1 # Black: the rest

plot(log(results$oddsratio), -log10(results$p.value), main="Volcano plot",
     xlab="Coefficient", ylab="-log10(p-value)", pch=8, cex=0.8, col = results$color)

par(new=TRUE)
# Emphasize that the same values are significant by BH and HB
x <- as.numeric(log(results$oddsratio[significant.species.bh]))
y <- as.numeric(-log10(results$p.value[significant.species.bh]))
points(x, y, col = 'green')
```



1.f)

```r
# Significant lipid species (HB, alpha = 0.05)
as.character(holm.bonferroni(results, 0.05)$lipid.species)
```

```
## [1] "PS 36:4"  "PS 38:5"  "PC 34:6"  "LPC 14:0" "LPC 22:6" "PC 34:2"
```

```r
# Significant lipid species (BH, alpha = 0.05)
as.character(benjamini.hochberg(results, 0.05)$lipid.species)
```

```
##  [1] "PS 36:4"  "PS 38:5"  "PC 34:6"  "LPC 14:0" "LPC 22:6" "PC 34:2"
##  [7] "PC 36:2"  "PS 38:6"  "PS 38:4"  "PS 40:6"  "LPC 17:0" "PC 42:1"
## [13] "PC 34:0"  "PC 36:3"  "PS 34:4"  "LPC 18:2"
```

Problem 2

```
prepare.glmnet <- function(data, formula=~ .) {
  ## create the design matrix to deal correctly with factor variables,
  ## without losing rows containing NAs
  old.opts <- options(na.action='na.pass')
  x <- model.matrix(formula, data)
  4
  options(old.opts)
  ## remove the intercept column, as glmnet will add one by default
  x <- x[, -match("(Intercept)", colnames(x))]
  return(x)
}
```

```
# Read the data
breast.cancer <- read.csv("data_assignment2/wdbc2.csv")
```

2.a)

```
set.seed(1)
train.idx <- createDataPartition(breast.cancer$diagnosis, p=0.7)$Resample1
breast.cancer.train <- breast.cancer[train.idx,]

y <- breast.cancer.train$diagnosis
# Exclude the target variable and IDs
x <- prepare.glmnet(breast.cancer.train, ~ . - diagnosis - id)

set.seed(1)
fit.ridge <- cv.glmnet(x, y, family='binomial', type.measure = 'auc', alpha=0)
set.seed(1)
fit.lasso <- cv.glmnet(x, y, family='binomial', type.measure = 'auc')
```

2.b)

```
# Rounding doesn't change the choice of AUC

opt.lasso.auc <- fit.lasso$cvm[which(fit.lasso$lambda == fit.lasso$lambda.min)]
opt.ridge.auc <- fit.ridge$cvm[which(fit.ridge$lambda == fit.ridge$lambda.min)]
opt.lasso.1se <- fit.lasso$cvm[which(fit.lasso$lambda == fit.lasso$lambda.1se)]
opt.ridge.1se <- fit.ridge$cvm[which(fit.ridge$lambda == fit.ridge$lambda.1se)]


paste0("AUC correspoding to the optimal lambda in Lasso regression: ",
       round(opt.lasso.auc, 3))
```

```
## [1] "AUC correspoding to the optimal lambda in Lasso regression: 0.978"
```

```
paste0("AUC correspoding to the optimal lambda in Ridge regression: ",
       round(opt.ridge.auc, 3))
```

```
## [1] "AUC correspoding to the optimal lambda in Ridge regression: 0.976"
```

```r
paste0("AUC correspoding to the lambda st. AUC is within 1se of the max in Lasso regression: ",
round(opt.lasso.1se, 3))
```

```
## [1] "AUC correspoding to the lambda st. AUC is within 1se of the max in Lasso regression: 0.969"
```

```r
paste0("AUC correspoding to the lambda st. AUC is within 1se of the max in Ridge regression: ",
round(opt.ridge.1se, 3))
```

```
## [1] "AUC correspoding to the lambda st. AUC is within 1se of the max in Ridge regression: 0.967"
```

2.c)

```r
paste0("The optimal lambda in Lasso regression: ", signif(fit.lasso$lambda.min, 3))
```

```
## [1] "The optimal lambda in Lasso regression: 0.0117"
```

```r
paste0("The optimal lambda in Ridge regression: ", signif(fit.ridge$lambda.min, 3))
```

```
## [1] "The optimal lambda in Ridge regression: 0.0402"
```

```r
paste0("The lambda for AUC within 1se of the max in Lasso regression: ",
       signif(fit.lasso$lambda.1se, 3))
```

```
## [1] "The lambda for AUC within 1se of the max in Lasso regression: 0.0827"
```

```r
paste0("The lambda for AUC within 1se of the max in Ridge regression: ",
       signif(fit.lasso$lambda.1se, 3))
```

```
## [1] "The lambda for AUC within 1se of the max in Ridge regression: 0.0827"
```

```r
paste0("Lasso model size for the optimal lambda: ",
       signif(fit.lasso$nzero[fit.lasso$lambda == fit.lasso$lambda.min], 3))
```

```
## [1] "Lasso model size for the optimal lambda: 8"
```

```r
paste0("Ridge model size for the optimal lambda: ",
       signif(fit.ridge$nzero[fit.ridge$lambda == fit.ridge$lambda.min], 3))
```

```
## [1] "Ridge model size for the optimal lambda: 30"
```

```r
paste0("Lasso model size for the lambda within 1se of the max: ",
       signif(fit.lasso$nzero[fit.lasso$lambda == fit.lasso$lambda.1se], 3))
```

```
## [1] "Lasso model size for the lambda within 1se of the max: 5"
```

```r
paste0("Ridge model size for the lambda within 1se of the max: ",
       signif(fit.ridge$nzero[fit.ridge$lambda == fit.ridge$lambda.1se], 3))
```

```
## [1] "Ridge model size for the lambda within 1se of the max: 30"
```

```r
paste0("AUC correspoding to the optimal lambda in Lasso regression: ",
       round(opt.lasso.auc, 3))
```

```
## [1] "AUC correspoding to the optimal lambda in Lasso regression: 0.978"
```

```r
paste0("AUC correspoding to the optimal lambda in Ridge regression: ",
       round(opt.ridge.auc, 3))
```

```
## [1] "AUC correspoding to the optimal lambda in Ridge regression: 0.976"
```

```
paste0("AUC correspoding to the lambda st. AUC is within 1se of the max in Lasso regression: ",
round(opt.lasso.1se, 3))
```

```
## [1] "AUC correspoding to the lambda st. AUC is within 1se of the max in Lasso regression: 0.969"
```

```
paste0("AUC correspoding to the lambda st. AUC is within 1se of the max in Ridge regression: ",
round(opt.ridge.1se, 3))
```

```
## [1] "AUC correspoding to the lambda st. AUC is within 1se of the max in Ridge regression: 0.967"
```

2.d)

```
# Full model
full.model <- glm(diagnosis ~ ., data=breast.cancer.train, family="binomial")
# Backward elimination
model.b <- stepAIC(full.model, direction="back", trace = FALSE)

# Calculate standardized regression coefficient
std.coef.b <- lm.beta(model.b)

# Order by the absolute value of s.r.c
std.coef.b <- std.coef.b[order(abs(std.coef.b), decreasing = TRUE)]

# Selected variables and their std. reg. coefs.
std.coef.b
```

```
##     radius.worst        area.worst    concavepoints              area
##        54.788742        -40.256307        11.310374         -9.914996
##        concavity  concavity.worst      compactness     radius.stderr
##        -8.143271          7.478898        -7.170570          5.055362
##          texture fractaldimension smoothness.worst   symmetry.stderr
##         4.140394          3.001371         1.875849          1.755983
```

2.e)

```
# Null model
null.model <- glm(diagnosis ~ 1, data=breast.cancer.train, family="binomial")

# Stepwise selection
model.s <- stepAIC(null.model, scope=list(upper=full.model),direction="both", trace = FALSE)

# Calculate standardized regression coefficient
std.coef.s <- lm.beta(model.s)

# Order by the absolute value of s.r.c
std.coef.s <- std.coef.s[order(abs(std.coef.s), decreasing = TRUE)]

# Selected variables and their std. reg. coefs.
std.coef.s
```

```
##        radius.worst          area.worst concavepoints.worst
##           40.996636          -33.552163            5.929745
##       radius.stderr         compactness             texture
##            5.827213          -4.863382           3.692405
##                area    fractaldimension    smoothness.worst
##           -2.863976           2.822798           1.487387
##      symmetry.worst
##            1.406498
```

The variables that entered the model and were later discarded:

- perimeter.worst
- area.stderr
- perimeter

2.f)

The models are not nested - the appropriate and available metric is AIC. Although there is no formal test that can be applied when comparing the AICs of two models, we can still use it to measure the goodness of fit to the training data for each of the models.

```
paste0("AIC of the stepwise selection model: ", signif(AIC(model.s), 3)) # 108.816
```

```
## [1] "AIC of the stepwise selection model: 109"
```

```
paste0("AIC of the backward elimination model: ", signif(AIC(model.b), 3)) # 104.691
```

```
## [1] "AIC of the backward elimination model: 105"
```

The lower the AIC, the better the fit. In this case, the backward elimination model fits the data better - we do not know yet which model generalizes better.

2.g)

```
y.train.b <- predict(model.b, newdata=breast.cancer.train)
y.train.s <- predict(model.s, newdata=breast.cancer.train)

paste0("Training AUC of model B: ", signif(roc(y, y.train.b)$auc,3))
```

```
## [1] "Training AUC of model B: 0.993"
```

```
paste0("Training AUC of model S :", signif(roc(y, y.train.s)$auc,3))
```

```
## [1] "Training AUC of model S :0.992"
```

2.h)

```
breast.cancer.test <- breast.cancer[-train.idx, ]
y.test <- breast.cancer.test$diagnosis
x.test <- prepare.glmnet(breast.cancer.test, ~ . - diagnosis - id)

# Test predictions: Model S
y.test.s <-predict(model.s, newdata=breast.cancer.test)

# Test predictions: Model B
y.test.b <- predict(model.b, newdata = breast.cancer.test)
```

```r
# Test predictions: Lasso
y.test.lasso <- predict(fit.lasso, newx = x.test, s=fit.lasso$lambda.min)

# Test predictions: Ridge
y.test.ridge <- predict(fit.ridge, newx = x.test, s=fit.ridge$lambda.min)


roc(y.test, y.test.s, plot=TRUE, col='red', legacy.axes=TRUE)
```

```
##
## Call:
## roc.default(response = y.test, predictor = y.test.s, plot = TRUE,     col = "red", legacy.axes = TRU
##
## Data: y.test.s in 107 controls (y.test benign) < 63 cases (y.test malignant).
## Area under the curve: 0.9723
```

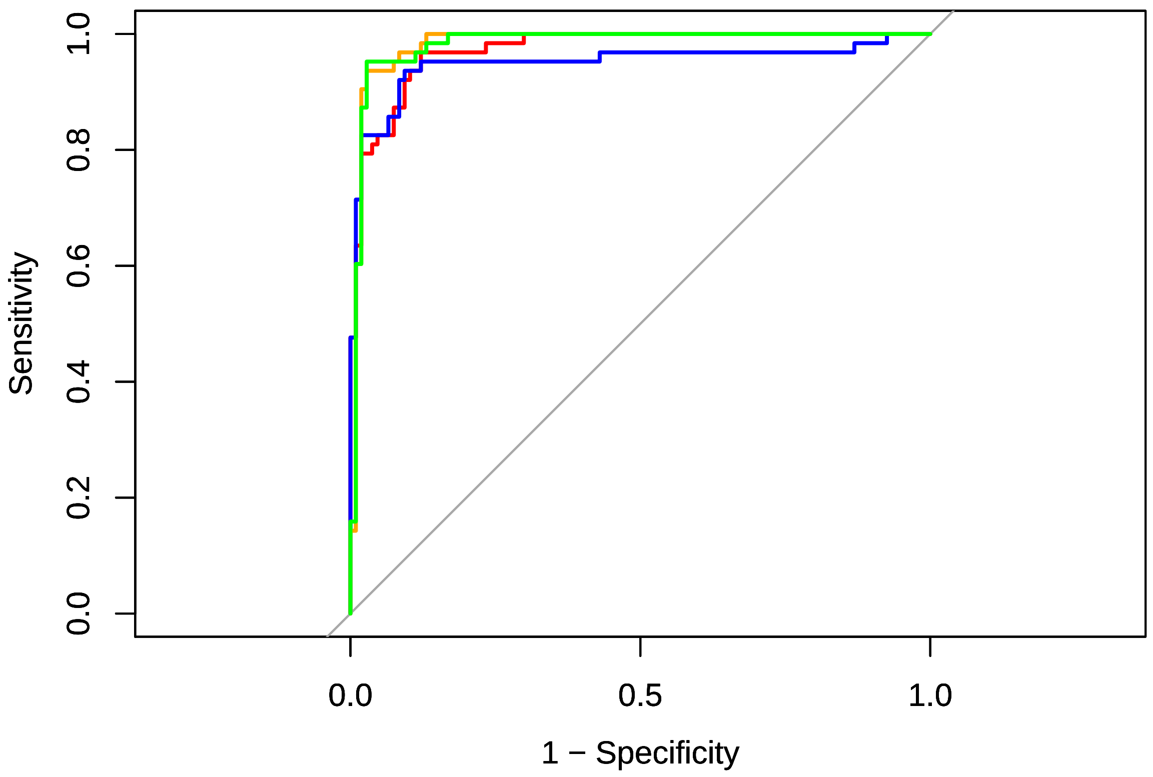
```r
par(new=TRUE)
roc(y.test, y.test.b, plot=TRUE, col='blue', legacy.axes=TRUE)
```

```
##
## Call:
## roc.default(response = y.test, predictor = y.test.b, plot = TRUE,     col = "blue", legacy.axes = TR
##
## Data: y.test.b in 107 controls (y.test benign) < 63 cases (y.test malignant).
## Area under the curve: 0.9496
```

```r
par(new=TRUE)
roc(y.test, y.test.lasso, newx=x, plot=TRUE, col='orange', legacy.axes=TRUE)
```

```
##
## Call:
## roc.default(response = y.test, predictor = y.test.lasso, plot = TRUE,     newx = x, col = "orange", 
##
## Data: y.test.lasso in 107 controls (y.test benign) < 63 cases (y.test malignant).
## Area under the curve: 0.9826
```

```r
par(new=TRUE)
roc(y.test, y.test.ridge, newx=x, plot=TRUE, col='green', legacy.axes=TRUE)
```



```
##
## Call:
## roc.default(response = y.test, predictor = y.test.ridge, plot = TRUE,     newx = x, col = "green", l
##
## Data: y.test.ridge in 107 controls (y.test benign) < 63 cases (y.test malignant).
## Area under the curve: 0.9821
```

```
# Test AUC

paste0("Test AUC of model B: ", signif(roc(y.test, y.test.b)$auc,3))
```

```
## [1] "Test AUC of model B: 0.95"
```

```
paste0("Test AUC of model S :", signif(roc(y.test, y.test.s)$auc,3))
```

```
## [1] "Test AUC of model S :0.972"
```

```
paste0("Test AUC of the Lasso model: ", signif(roc(y.test, y.test.lasso)$auc,3))
```

```
## [1] "Test AUC of the Lasso model: 0.983"
```

```
paste0("Test AUC of the Ridge model:", signif(roc(y.test, y.test.ridge)$auc,3))
```

```
## [1] "Test AUC of the Ridge model:0.982"
```

The penalised models probably do not overfit, their test AUCs are slightly better than the train AUCs. This observation is consistent with the idea of regularization - penalizing large coefficients to reduce overfitting. Models B and S have very high training AUCs (0.993 and 0.992) while their testing AUCs are lower (0.947 and 0.972). The difference is small, however, one interpretation of overfitting states that it takes place when it is possible to find a model with a higher test to train accuracy ratio so in this case we can say that models B and S overfit.

Problem 3

3.a)

```r
gdm <- read.table("data_assignment2/GDM.raw.txt", sep='\t',header=T,
                  stringsAsFactors=FALSE)

# Missing data imputation
column.means <- colMeans(gdm, na.rm=TRUE)

# Loop only over the columns with missing values
for (col in which(is.na(colSums(gdm)))){
    gdm[is.na(gdm[, col]), col] <- column.means[col]
}

# rsID and coded allele dataframe
rsID <- sapply(colnames(gdm)[-c(1:3)], function(x) strsplit(x, '_')[[1]][1])
allele <- sapply(colnames(gdm)[-c(1:3)], function(x) strsplit(x, '_')[[1]][2])

snp.allele <- data.frame(rsID, allele)
rownames(snp.allele) <- c()
```

3.b)

```r
univ.glm.test <- function(x, y, order=FALSE) {
  stopifnot(length(y) == length(x[, 1]))
  name <- NULL
  regr.coef <- NULL
  or <- NULL
  se <- NULL
  pval <- NULL
  for (i in 1:length(x)) {
    regr <- glm(y ~ x[, i], family="binomial")
    name[i] <- colnames(x)[i]
    regr.coef[i] <- coef(summary(regr))[2, 1]
    or[i] <- exp(coef(regr)[2])
    se[i] <- coef(summary(regr))[2, 2]
    pval[i] <- coef(summary(regr))[2, 4]
  }
  df <- data.frame(name, regr.coef, or, se, pval)
  # If order is set to TRUE order the data by increasing pval
  if (order) {
    return(df[order(df$pval),])
  }
  else{
    return(df)
  }

}
```

3.c)

```
univ.glm.test.results <- univ.glm.test(gdm[-c(1:3)], gdm$pheno)

snp.strongest <- univ.glm.test.results[which(univ.glm.test.results$pval
== min(univ.glm.test.results$pval)), ]

# Summary statistics of the most associated SNP
# (The smallest p-value)
snp.strongest
```

```
##            name regr.coef       or        se         pval
## 98 rs12243326_A 0.6489955 1.913618 0.1585863 4.269678e-05
```

```
# Normal approximation of 95% and 99% confidence
#intervals on the odds ratio for the most associated SNP
round(exp(snp.strongest$regr.coef + 1.96 * snp.strongest$se* c(-1, 1)), 3)
```

```
## [1] 1.402 2.611
```

```
round(exp(snp.strongest$regr.coef + 2.576 * snp.strongest$se* c(-1, 1)), 3)
```

```
## [1] 1.272 2.879
```

```
snp.most.protect <- univ.glm.test.results[which(univ.glm.test.results$regr.coef
== min(univ.glm.test.results$regr.coef)), ]

# Summary statistics of the SNP with the most significant protective effect
# (The most negative coef)
snp.most.protect
```

```
##            name  regr.coef        or        se      pval
## 60 rs11575839_C -0.6112123 0.5426926 0.3761926 0.1042196
```

```
# Normal approximation of 95% and 99% confidence
#intervals on the odds ratio for the SNP with most significant protective effect
round(exp(snp.most.protect$regr.coef + 1.96 * snp.most.protect$se* c(-1, 1)), 3)
```

```
## [1] 0.260 1.134
```

```
round(exp(snp.most.protect$regr.coef + 2.576 * snp.most.protect$se* c(-1, 1)), 3)
```

```
## [1] 0.206 1.430
```

3.d)-3.e)

The report part:

There are two hit snps (rs12243326, rs2237897). For rs12243326, the names that are within a 1Mb window: TCF7L2 For rs2237897: TH, KCNQ1, CACNA2D4, SMG6

Model 1 (SNPs with pval < 1e-4): or 2.722, 95% ci (1.92, 3.89), pval 2.342e-08 Model 2 (SNPs with pval < 1e-3): or 1.45, 95% ci(1.28, 165), pval 8.231e-09 Model 3: FTO or 1.4, 95% ci (0.8, 2.43), pval 0.227

```
# Last minute problems with moving the code to Rmarkdown.....

# gdm.annot <- read.table("data_assignment2/GDM.annot.txt", sep='\t',header=T, stringsAsFactors=FALSE)
# univ.glm.test.results$snp <- sapply(univ.glm.test.results$names, function(x) strsplit(as.character(x)
```

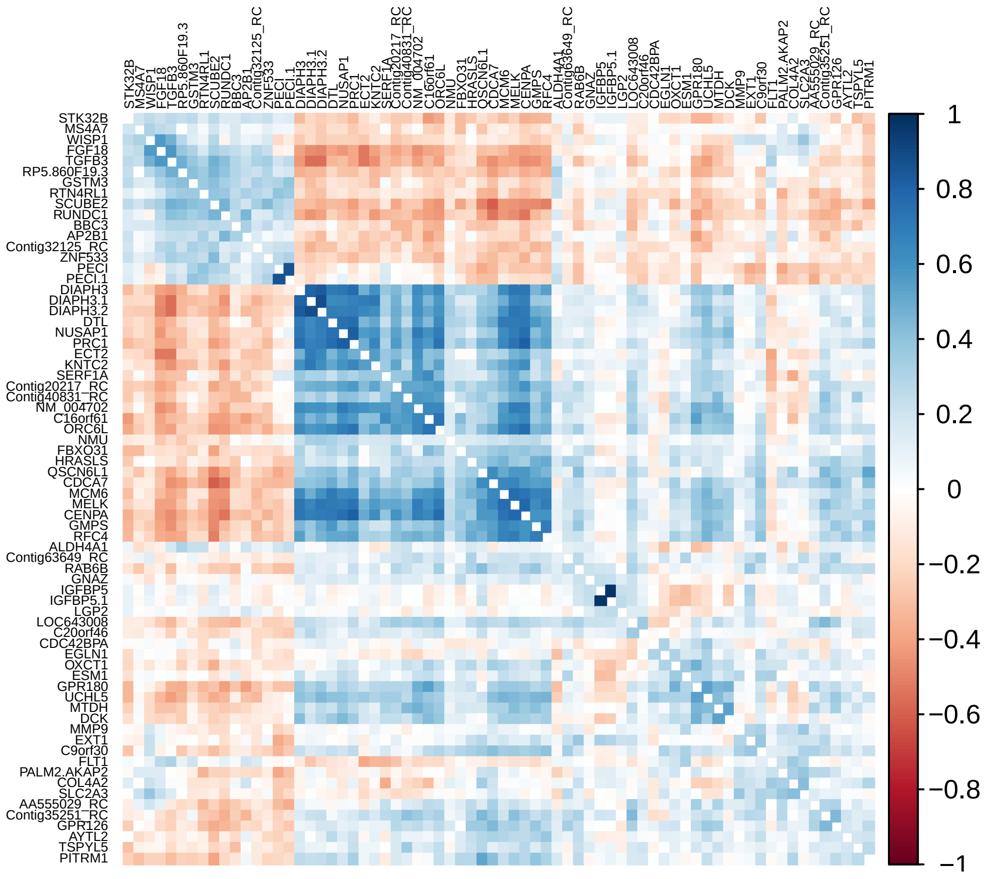
```r
# results.annotated <- merge(univ.glm.test.results, gdm.annot, by='snp') # Merge by SNP

# hit.snps <- subset(results.annotated, pval < 1e-4)

# SNP names, effect allele, chromosome number and corresponding gene name
# hit.snps

# pos1 <- hit.snps$pos[1]
# pos2 <- hit.snps$pos[2]

# Window for the first snp < 1e-4 (excluding the hit snp)
# paste(unique(results.annotated[abs(results.annotated$pos - pos1) <= 10^6 & results.annotated$pos != p
# Window for the second one
# paste(unique(results.annotated[abs(results.annotated$pos - pos2) <= 10^6 & results.annotated$pos != p

# # We can build a simple genetic risk score for each patient simply by summing the allele counts acros
# associated SNPs (unweighted score). To take the direction of the effect into consideration, we switch
# for the SNPs that have an inverse association: we use function sign() for this purpose, which returns
# positive elements, and -1 for negative elements

# snps.pval.3 <- subset(results.annotated, pval < 1e-3 )
# snps.fto <- subset(results.annotated, gene=="FTO")


# hit.snps.risk <- results.annotated[which (results.annotated$snp %in% hit.snps$snp), ]
# stopifnot(hit.snps.risk$snp == hit.snps$snp)
# it.snps.unweighted.score <- as.matrix(hit.snps.risk) %*% sign(hit.snps$reg.coef)
```

Problem 4

```r
breast.cancer <- read.csv("data_assignment2/nki.csv")
```

4.a)

```r
gene.expression <- breast.cancer[,(ncol(breast.cancer)-69):ncol(breast.cancer)]
gene.expression.corr <- cor(gene.expression[, sapply(gene.expression, is.numeric)],
                            use="pairwise.complete")
corrplot(gene.expression.corr, order="hclust", diag=FALSE, tl.col="black", tl.cex=0.4, method="color")
```

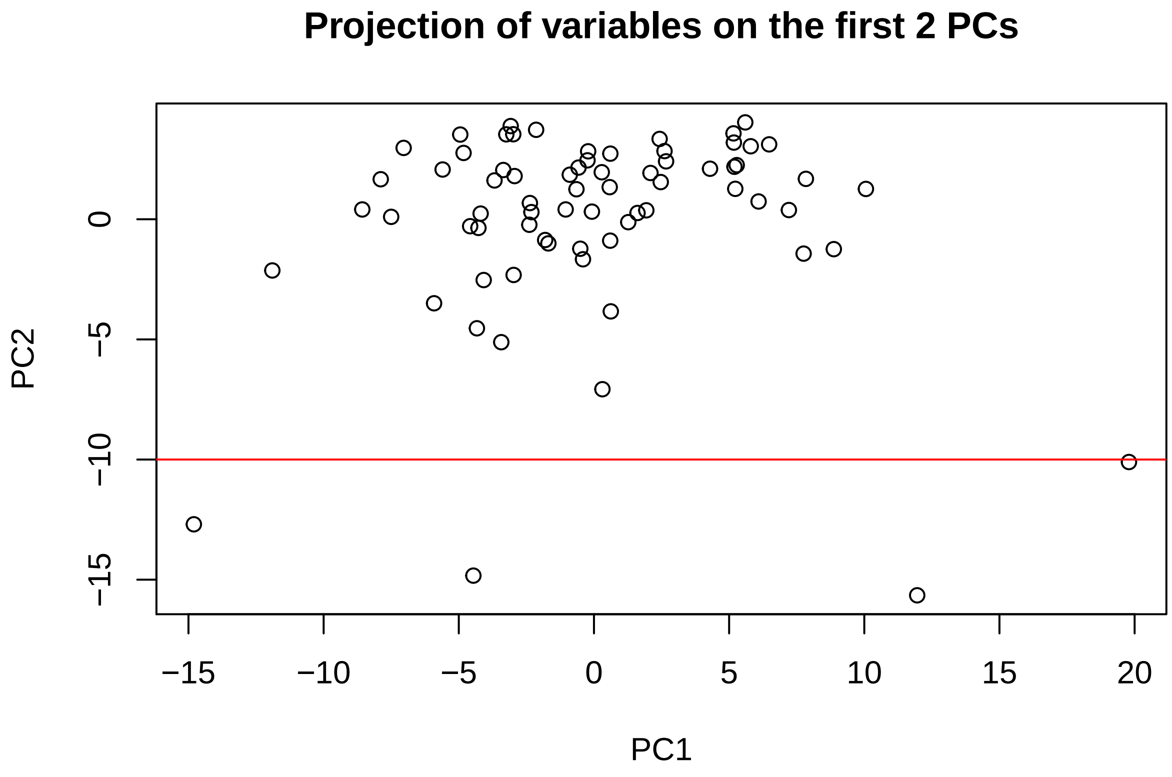


```r
# The unique pairs of distinct variables with the
# correlation coefficient greater than 0.8

k <- 1

for (i in 1:ncol(gene.expression.corr)){
  for (j in 1:i){
    gene1 <- colnames(gene.expression.corr)[i]
    gene2 <- rownames(gene.expression.corr)[j]
    corr <- gene.expression.corr[i, j]
    if (abs(corr) > 0.8 && (gene1 != gene2) ){
      cat(sprintf("The correlation between %s and %s is %f\n",
                  gene1, gene2, corr))
    }
  }
}
```

```
## The correlation between DIAPH3.1 and DIAPH3 is 0.803137
## The correlation between DIAPH3.2 and DIAPH3 is 0.833859
## The correlation between DIAPH3.2 and DIAPH3.1 is 0.886874
## The correlation between PECI.1 and PECI is 0.869784
## The correlation between IGFBP5.1 and IGFBP5 is 0.977503
## The correlation between PRC1 and NUSAP1 is 0.829836
## The correlation between CENPA and PRC1 is 0.817542
```

4.b)

```r
pca.vars <- prcomp(t(gene.expression), scale=TRUE)
plot(pca.vars$x[, 1:2], main="Projection of variables on the first 2 PCs")
abline(h=-10, col='red')
```



```r
perc.expl <- pca.vars$sdev^2 / sum(pca.vars$sdev^2)
print(sum(perc.expl[1:2]))
```

```
## [1] 0.3316167
```

```r
pca <- pca.vars$x[, 1:2]

# Simple rule based on the plot:
print(rownames(pca)[pca[,2] <= -10])
```

```
## [1] "SCUBE2" "MMP9"   "ZNF533" "CDCA7"
```

4.c)

```r
pca.patients <- prcomp((gene.expression), scale=TRUE)
pca.components <- pca.patients$x[, 1:3]

fit.m11 <- glm(breast.cancer$Event ~ pca.components[,1], family="binomial")
fit.m12 <- glm(breast.cancer$Event ~ pca.components[,2], family="binomial") # NOT SIGNIFICANT
fit.m13 <- glm(breast.cancer$Event ~ pca.components[,3], family="binomial") # The lowest pval

fit.m11.adj <- glm(breast.cancer$Event ~ pca.components[,1]+
                     breast.cancer$Age+breast.cancer$EstrogenReceptor, family="binomial")
```

```r
fit.m12.adj <- glm(breast.cancer$Event ~ pca.components[,2]+
                     breast.cancer$Age+breast.cancer$EstrogenReceptor, family="binomial")
fit.m13.adj <- glm(breast.cancer$Event ~ pca.components[,3]+
                     breast.cancer$Age+breast.cancer$EstrogenReceptor, family="binomial")


coef(summary(fit.m11))
```

```
##                        Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)          -0.7296809 0.18328720 -3.981080 6.860296e-05
## pca.components[, 1]   0.1176835 0.04532846  2.596239 9.425039e-03
```

```r
coef(summary(fit.m12))
```

```
##                        Estimate Std. Error    z value     Pr(>|z|)
## (Intercept)          -0.6971060 0.17747389 -3.9279356 8.567813e-05
## pca.components[, 2] -0.0671668 0.07789272 -0.8622989 3.885231e-01
```

```r
coef(summary(fit.m13))
```

```
##                        Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)          -0.7285011 0.18315080 -3.977603 6.961359e-05
## pca.components[, 3]   0.2435485 0.09273186  2.626374 8.630000e-03
```

```r
coef(summary(fit.m11.adj))
```

```
##                                            Estimate Std. Error     z value
## (Intercept)                              2.69711953 1.56661970  1.72161726
## pca.components[, 1]                      0.12407472 0.05574522  2.22574651
## breast.cancer$Age                       -0.07764297 0.03533212 -2.19751779
## breast.cancer$EstrogenReceptorPositive -0.01420363 0.53992349 -0.02630675
##                                          Pr(>|z|)
## (Intercept)                             0.08513887
## pca.components[, 1]                     0.02603117
## breast.cancer$Age                       0.02798349
## breast.cancer$EstrogenReceptorPositive 0.97901267
```

```r
coef(summary(fit.m12.adj))
```

```
##                                            Estimate Std. Error    z value
## (Intercept)                              2.92279664 1.58465761  1.8444342
## pca.components[, 2]                      0.02065952 0.09047753  0.2283387
## breast.cancer$Age                       -0.06813276 0.03455678 -1.9716177
## breast.cancer$EstrogenReceptorPositive -0.77229838 0.51537872 -1.4985065
##                                          Pr(>|z|)
## (Intercept)                             0.06511989
## pca.components[, 2]                     0.81938297
## breast.cancer$Age                       0.04865327
## breast.cancer$EstrogenReceptorPositive 0.13400170
```

```r
coef(summary(fit.m13.adj))
```

```
##                                            Estimate Std. Error   z value
## (Intercept)                              2.47437162 1.57066744  1.575363
## pca.components[, 3]                      0.23847167 0.09453064  2.522692
## breast.cancer$Age                       -0.05836367 0.03506933 -1.664237
## breast.cancer$EstrogenReceptorPositive -0.79560119 0.45416138 -1.751803
##                                              Pr(>|z|)
## (Intercept)                               0.11517264
## pca.components[, 3]                       0.01164604
## breast.cancer$Age                         0.09606515
## breast.cancer$EstrogenReceptorPositive 0.07980772
```

To measure the associations, we consider p-values first. Overall in the adjusted models the associations between the principal components and the outcome variables are lower than in the unadjusted models because other variables (Age, Estrogen Receptor) explain a part of the variance. For the significance level of alpha = 0.05, we can not say anything about any of the models that use the second principal component (pval is too big, null hypothesis cannot be rejected).

```
prepare.glmnet <- function(data, formula=~ .) {
  ## create the design matrix to deal correctly with factor variables,
  ## without losing rows containing NAs
  old.opts <- options(na.action='na.pass')
  x <- model.matrix(formula, data)
  4
  options(old.opts)
  ## remove the intercept column, as glmnet will add one by default
  x <- x[, -match("(Intercept)", colnames(x))]
  return(x)
}

set.seed(1)

y.nki <- breast.cancer$Event
x.nki <- prepare.glmnet(breast.cancer, ~ . - Event)

fit.lasso <- cv.glmnet(x.nki, y.nki, family='binomial', type.measure = 'auc')
fit.lasso.genes.only <- cv.glmnet(x.nki, y.nki, family='binomial',
                                  penalty.factor=c(rep(0,6),rep(1, 70)), type.measure = 'auc')

opt.lasso.auc <- fit.lasso$cvm[which(fit.lasso$lambda == fit.lasso$lambda.min)]
opt.lasso.genes.only.auc <- fit.lasso.genes.only$cvm[which(fit.lasso.genes.only$lambda ==
                                                           fit.lasso.genes.only$lambda.min)]

paste0("AUC of the model that penalizes all of the variables: ",
       round(opt.lasso.auc, 3))
```

```
## [1] "AUC of the model that penalizes all of the variables: 0.756"
```

```
paste0("AUC of the model that penalizes only gene expression variables: ",
       round(opt.lasso.genes.only.auc, 3))
```

```
## [1] "AUC of the model that penalizes only gene expression variables: 0.738"
```

```
paste0("Lasso model size (all vars penalized): ",
       signif(fit.lasso$nzero[fit.lasso$lambda == fit.lasso$lambda.min], 3))
```

```
## [1] "Lasso model size (all vars penalized): 37"
```

```
paste0("Lasso model size (gene expression vars penalized): ",
       signif(fit.lasso.genes.only$nzero[fit.lasso.genes.only$lambda
                                         == fit.lasso.genes.only$lambda.min], 3))
```

```
## [1] "Lasso model size (gene expression vars penalized): 45"
```

The model that does not penalize all of the coefficients might be more prone to overfitting (we would need a test dataset to assess this).