Biomedical Data Science - Assignment 2

Yile Shi (s2168022)

2022/3/26

Assignment 2

Biomedical Data Science

Due on Tuesday 5th April 2022, 5:00pm

The assignment is marked out of 100 points, and will contribute to 30% of your final mark. Please knit this document in PDF format and submit using the gradescope link on Learn. If you can't knit to PDF directly, knit it to word and you should be able to either convert to PDF or print it and scan to PDF using a scanning app on your phone. If you have any code that doesn't run you won't be able to knit the document so comment it as you might still get some grades for partial code. Clear and reusable code will be rewarded so pay attention to indentation, choice of variable identifiers, comments, error checking, etc. An initial code chunk is provided after each subquestion but create as many chunks as you feel is necessary to make a clear report. Add plain text explanations in between the chunks as and when required and any comments necessary within code chunks to make it easier to follow your code/reasoning.

Problem 1 (27 points)

File wdbc2.csv (available from the accompanying zip folder on Learn) refers to a study of breast cancer where the outcome of interest is the type of the tumour (benign or malignant, recorded in column "diagnosis"). The study collected 30 imaging biomarkers on 569 patients.

Problem 1.a (7 points)

Using package caret, create a data partition so that the training set contains 70% of the observations (set the random seed to 984065 beforehand). Fit both a ridge regression model and a lasso model which uses cross-validation on the training set to diagnose the type of tumour from the 30 biomarkers. Then use a plot to help identify the penalty parameter λ that maximizes the AUC. Note: There is no need to use the prepare.glmnet() function from lab 4, using as.matrix() with the required columns is sufficient.

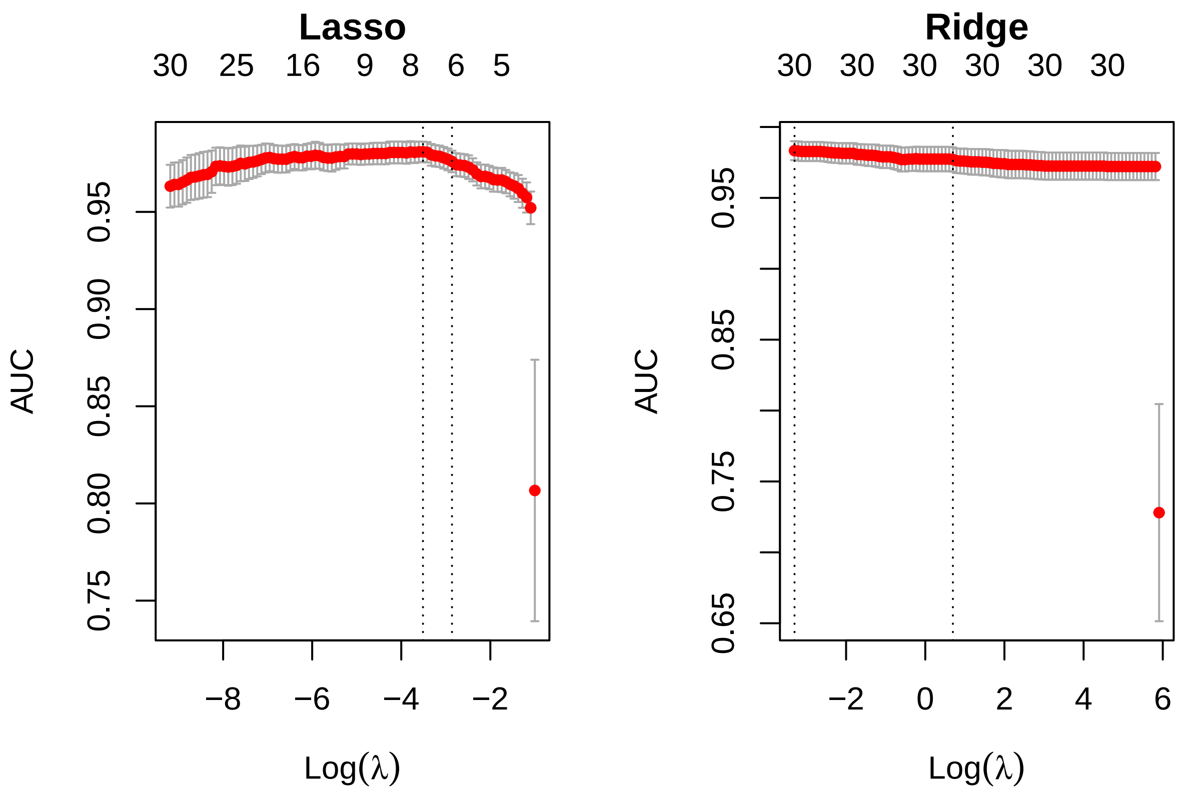
Solution:

```r
# Read dataset into R
wdbc <- fread("assignment2/wdbc2.csv")

# convert strings in 'diagnosis' to numeric
wdbc$diagnosis <- ifelse(wdbc$diagnosis == 'malignant', 1, 0)
```

```r
# set random seed beforehand
set.seed(984065)

# create training and testing sets
train.idx <- createDataPartition(y = wdbc$diagnosis, p = 0.7)$Resample1
train.wdbc <- wdbc[train.idx, -c(1)]
test.wbdc <- wdbc[-train.idx, -c(1)]
```

```r
# fit a Ridge regression and a Lasso regression using CV on training set
# first, we need matrices for x and y in the regression function
train.x <- train.wdbc[, -c(1)]
test.x <- test.wbdc[, -c(1)]
train.x.mat <- as.matrix(train.x)
test.x.mat <- as.matrix(test.x)

train.y.mat <- train.wdbc$diagnosis
test.y.mat <- test.wbdc$diagnosis

# now we fit regressions using CV
set.seed(984065)
fit.cv.lasso <- cv.glmnet(train.x.mat, train.y.mat,
                          family = "binomial", type.measure = "auc")
fit.cv.ridge <- cv.glmnet(train.x.mat, train.y.mat, alpha = 0,
                          family = "binomial", type.measure = "auc")
```

```r
# make plots to find the best lambdas that maximise the AUC
par(mfrow = c(1,2), mar = c(4,4,5,2))
plot(fit.cv.lasso, main = "Lasso")
plot(fit.cv.ridge, main = "Ridge")
```



```
# the values of best lambdas
cat("The best lambda in Lasso regression: ", fit.cv.lasso$lambda.min, "\n")
```

```
## The best lambda in Lasso regression:  0.02982835
```

```
cat("The best lambda in Ridge regression: ", fit.cv.ridge$lambda.min)
```

```
## The best lambda in Ridge regression:  0.03677378
```

Problem 1.b (2 points)

Create a data table that for each value of 'lambda.min' and 'lambda.1se' for each model fitted in problem 1.a reports: * the corresponding AUC, * the corresponding model size. Use 3 significant digits for floating point values and comment on these results. Hint: The AUC values are stored in the field called 'cvm'.

Solution:

```r
# now we create the required data table
rep.dt <- data.table(model = c("Lasso(lambda.min)", "Lasso(lambda.1se)",
                               "Ridge(lambda.min)", "Ridge(lambda.1se)"),
                     lambda = c(round(fit.cv.lasso$lambda.min, 3),
                                round(fit.cv.lasso$lambda.1se, 3),
                                round(fit.cv.ridge$lambda.min, 3),
                                round(fit.cv.ridge$lambda.1se, 3)),
                     AUC = c(round(fit.cv.lasso$cvm[fit.cv.lasso$index[1]], 3),
                             round(fit.cv.lasso$cvm[fit.cv.lasso$index[2]], 3),
                             round(fit.cv.ridge$cvm[fit.cv.ridge$index[1]], 3),
                             round(fit.cv.ridge$cvm[fit.cv.ridge$index[2]], 3)),
                     modelsize = c(round(fit.cv.lasso$nzero[fit.cv.lasso$index[1]], 3),
                                   round(fit.cv.lasso$nzero[fit.cv.lasso$index[2]], 3),
                                   round(fit.cv.ridge$nzero[fit.cv.ridge$index[1]], 3),
                                   round(fit.cv.ridge$nzero[fit.cv.ridge$index[2]], 3)))
rep.dt
```

```
##                 model lambda   AUC modelsize
## 1: Lasso(lambda.min)  0.030 0.981         8
## 2: Lasso(lambda.1se)  0.057 0.976         6
## 3: Ridge(lambda.min)  0.037 0.983        30
## 4: Ridge(lambda.1se)  2.009 0.977        30
```

Problem 1.c (7 points)

Perform both backward (we'll later refer to this as model B) and forward (model S) stepwise selection on the same training set derived in problem 1.a. Report the variables selected and their standardized regression coefficients in decreasing order of the absolute value of their standardized regression coefficient. Discuss the results and how the different variables entering or leaving the model influenced the final result.

Solution:

```r
# define the full model and null model
full.model <- glm(diagnosis ~ ., data = train.wdbc, family = "binomial")
null.model <- glm(diagnosis ~ 1, data = train.wdbc, family = "binomial")
```

```r
# backward stepwise selection
model.B <- stepAIC(full.model, scope = list(lower = null.model),
                   direction = "back", trace = FALSE)

# forward stepwise selection
model.S <- stepAIC(null.model, scope = list(upper = full.model),
                   direction = "forward", trace = FALSE)
```

```r
# selected variables with standardized regression coefficients in model.B
# get the standardized coefficients
std.coef.B <- lm.beta(model.B)$standardized.coefficients

# order the coefficients in decreasing order of their absolute values
std.coef.B <- std.coef.B[order(abs(std.coef.B), decreasing = TRUE)]

cat("selected variables in model.B: \n")
```

```
## selected variables in model.B:
```

```r
round(std.coef.B, 3)
```

```
##        radius.worst          area.worst           perimeter     concavity.worst
##              53.047             -40.610             -18.501               8.473
##       concavepoints              radius       radius.stderr       texture.worst
##               8.399               7.378               7.372               5.605
##    compactness.worst concavepoints.worst      texture.stderr    smoothness.worst
##              -5.321              -3.832              -3.723               2.008
##         (Intercept)
##               0.000
```

```r
# selected variables with standardized regression coefficients in model.S
# get the standardized coefficients
std.coef.S <- lm.beta(model.S)$standardized.coefficients

# order the coefficients in decreasing order of their absolute values
std.coef.S <- std.coef.S[order(abs(std.coef.S), decreasing = TRUE)]

cat("selected variables in model.S: \n")
```

```
## selected variables in model.S:
```

```
std.coef.S
```

```
##        area.worst      radius.worst   perimeter.worst         perimeter
##        -44.347062         39.117707         16.396414        -13.329464
##     radius.stderr compactness.worst            radius         concavity
##         10.235291         -5.930017          5.544839          5.364296
##     texture.worst  perimeter.stderr   concavity.worst    texture.stderr
##          4.932050         -4.761290          4.299930         -3.028769
##  smoothness.worst       area.stderr       (Intercept)
##          2.661371          2.278462          0.000000
```

Problem 1.d (3 points)

Compare the goodness of fit of model B and model S in an appropriate way.

Solution:

Here we consider using AIC as an appropriate criterion to compare the goodness-of-fit to the training data of two models.

```
data.frame(model.B = round(model.B$aic, 3),
           model.S = round(model.S$aic, 3),
           row.names = "AIC")
```

```
##     model.B model.S
## AIC  99.471 105.295
```

```
# model.B
signif(pchisq(model.B$null.deviance - model.B$deviance,
              df = model.B$df.null - model.B$df.residual,
              lower.tail = FALSE), 3)
```

```
## [1] 1.46e-89
```

```
# model.S
signif(pchisq(model.S$null.deviance - model.S$deviance,
              df = model.S$df.null - model.S$df.residual,
              lower.tail = FALSE), 3)
```

```
## [1] 1.35e-87
```

Problem 1.e (2 points)

Compute the training AUC for model B and model S.

Solution:

```r
# predict values on training set
pred.B <- predict(model.B, newdata = train.wdbc, type = "response")
pred.S <- predict(model.S, newdata = train.wdbc, type = "response")
```

```r
# auc
auc.B <- roc(train.y.mat ~ pred.B)$auc
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
auc.S <- roc(train.y.mat ~ pred.S)$auc
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
cat("AUC of Model.B: ", round(auc.B, 3), "\n")
```

```
## AUC of Model.B:  0.994
```

```r
cat("AUC of Model.S: ", round(auc.S, 3))
```

```
## AUC of Model.S:  0.993
```

Problem 1.f (6 points)

Use the four models to predict the outcome for the observations in the test set (use the lambda at 1 standard error for the penalised models). Plot the ROC curves of these models (on the sameplot, using different colours) and report their test AUCs. Compare the training AUCs obtained in problems 1.b and 1.e with the test AUCs and discuss the fit of the different models.

Solution:

```r
# predict values of testing set
pred.lasso.test <- predict(fit.cv.lasso, newx = test.x.mat,
                           s = fit.cv.lasso$lambda.1se, type = "response")
pred.ridge.test <- predict(fit.cv.ridge, newx = test.x.mat,
                           s = fit.cv.ridge$lambda.1se, type = "response")
pred.B.test <- predict(model.B, newdata = test.x, type = "response")
pred.S.test <- predict(model.S, newdata = test.x, type = "response")

# plot ROC curves of four models
roc.lasso <- roc(test.y.mat, pred.lasso.test,
                 plot = TRUE, col = "blue", xlim = c(0, 1),
                 main = "ROC curves (on testing data)")
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(test.y.mat, pred.lasso.test, plot = TRUE, col = "blue", :
## Deprecated use a matrix as predictor. Unexpected results may be produced, please
## pass a numeric vector.
```

```
## Setting direction: controls < cases
```

```r
roc.ridge <- roc(test.y.mat, pred.ridge.test,
                 plot = TRUE, col = "red", add = TRUE, xlim = c(0, 1))
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(test.y.mat, pred.ridge.test, plot = TRUE, col = "red", :
## Deprecated use a matrix as predictor. Unexpected results may be produced, please
## pass a numeric vector.
```

```
## Setting direction: controls < cases
```

```r
roc.B <- roc(test.y.mat, pred.B.test,
                 plot = TRUE, col = "green", add = TRUE, xlim = c(0, 1))
```

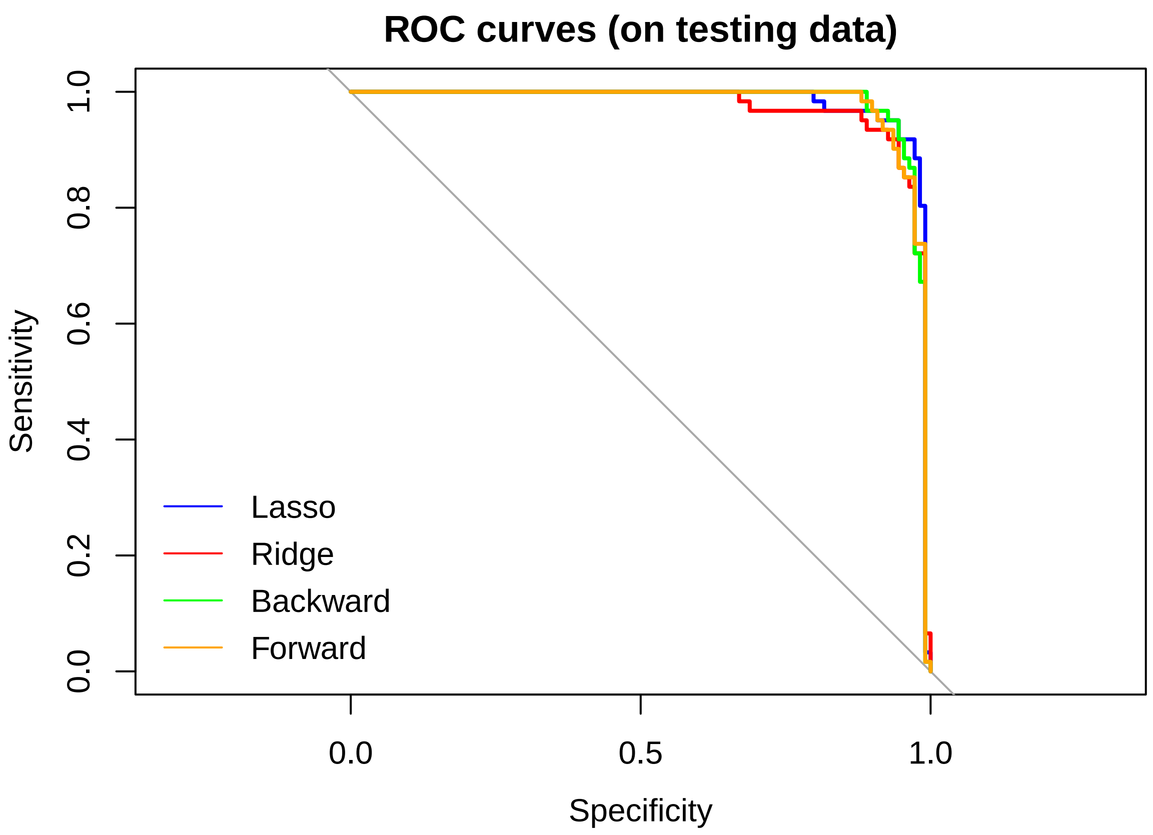
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
roc.S <- roc(test.y.mat, pred.S.test,
                 plot = TRUE, col = "orange",add = TRUE,  xlim = c(0, 1))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# add legend
legend("bottomleft", legend = c("Lasso", "Ridge", "Backward", "Forward"),
       col = c("blue", "red", "green", "orange"),
       lty = c(1, 1, 1, 1), bty = "n")
```



```r
# data frame to report the AUC of each model on both training and testing sets
data.frame(train.AUC = c(round(fit.cv.lasso$cvm[fit.cv.lasso$index[2]], 3),
                         round(fit.cv.ridge$cvm[fit.cv.ridge$index[2]], 3),
                         round(auc.B, 3), round(auc.S, 3)),
           test.AUC = c(round(roc.lasso$auc, 3), round(roc.ridge$auc, 3),
                        round(roc.B$auc, 3), round(roc.S$auc, 3)),
           row.names = c("Lasso", "Ridge", "Backward", "Forward"))
```

```
##          train.AUC test.AUC
## Lasso        0.976    0.981
## Ridge        0.977    0.971
## Backward     0.994    0.980
## Forward      0.993    0.979
```

Problem 2 (40 points)

File GDM.raw.txt (available from the accompanying zip folder on Learn) contains 176 SNPs to be studied for association with incidence of gestational diabetes (a form of diabetes that is specific to pregnant women). SNP names are given in the form "rs1234_X" where "rs1234" is the official identifier (rsID), and "X" (one of A, C, G, T) is the reference allele.

Problem 2.a (3 points)

Read file GDM.raw.txt into a data table named gdm.dt. Impute missing values in gdm.dt according to SNP-wise median allele count.

Solution:

```r
# read data into R
gdm.dt <- data.table(read.table("assignment2/GDM.raw.txt", header = TRUE))
```

```r
# define a function to impute missing values in each column with median
impute.to.median <- function(x){
  na.idx <- is.na(x)
  x[na.idx] <- median(x, na.rm = TRUE)
  return(x)
}

# impute the missing values using the function
gdm.dt.imputed <- gdm.dt[, (colnames(gdm.dt)) := lapply(.SD, impute.to.median),
                          .SDcols = colnames(gdm.dt)]

sum(is.na(gdm.dt.imputed))
```

```
## [1] 0
```

Problem 2.b (8 points)

Write function univ.glm.test <- function(x, y, order = FALSE) where x is a data table of SNPs, y is a binary outcome vector, and order is a boolean. The function should fit a logistic regression model for each SNP in x, and return a data table containing SNP names, regression coefficients, odds ratios, standard errors and p-values. If order is set to TRUE, the output data table should be ordered by increasing p-value.

Solution:

```r
univ.glm.test <- function(x, y, order = FALSE){
  # initialize lists for output
  name <- NULL
  coef <- NULL
  OR <- NULL
  se <- NULL
  p.value <- NULL

  # for convenience, we convert x to data frame
  x <- as.data.frame(x)

  # get the names of columns in x
  x.cols <- colnames(x)

  # logistic regression for each SNP in x
  for (i in 1:length(x.cols)){
    logreg <- glm(y ~ x[, i], family = "binomial")
    reg.sum <- summary(logreg)
    name[i] <- x.cols[i]
    coef[i] <- reg.sum$coefficients[2, 1]
    OR[i] <- exp(coef[i])
    se[i] <- reg.sum$coefficients[2, 2]
    p.value[i] <- reg.sum$coefficients[2, 4]
  }

  # create a data table including the values
  dt <- data.table(name, coef, OR, se, p.value)

  # if order = TRUE ...
  if (order == TRUE){
    return(setorder(dt, p.value))
  }
  else{
    return(dt)
  }
}
```

Problem 2.c (5 points)

Using function univ.glm.test(), run an association study for all the SNPs in gdm.dt against having gestational diabetes (column "pheno"). For the SNP that is most strongly associated to increased risk of gestational diabetes and the one with most significant protective effect, report the summary statistics from the GWAS as well as the 95% and 99% confidence intervals on the odds ratio.

Solution:

```r
# Use pre-defined function on the imputed dataset
# construct function input
SNP.x <- gdm.dt.imputed[, -c(1, 2, 3)]
gd.y <- gdm.dt.imputed$pheno
res <- univ.glm.test(x = SNP.x, y = gd.y, order = TRUE)
head(res, 5)
```

```
##             name       coef        OR        se      p.value
## 1: rs12243326_A  0.6454198 1.9067873 0.1583787 4.598104e-05
## 2:  rs2237897_T -0.4394456 0.6443936 0.1126133 9.530178e-05
## 3:  rs2237892_C -0.4042888 0.6674513 0.1108494 2.651236e-04
## 4:  rs4506565_T  0.4865711 1.6267287 0.1346281 3.012927e-04
## 5:  rs7903146_C  0.4790441 1.6145304 0.1382068 5.279878e-04
```

```r
#The SNP mostly strongly associated to increased risk of gestational diabetes is:
SNP.msa <- res[1, ]
print(SNP.msa)
```

```
##             name      coef       OR        se      p.value
## 1: rs12243326_A 0.6454198 1.906787 0.1583787 4.598104e-05
```

```r
# The 95% and 99% confidence intervals of corresponding odds ratio are:
CI.95.msa <- exp(SNP.msa$coef + qnorm(0.975) * SNP.msa$se * c(-1, 1))
CI.99.msa <- exp(SNP.msa$coef + qnorm(0.995) * SNP.msa$se * c(-1, 1))
data.frame(lower = c(round(CI.95.msa[1], 3), round(CI.99.msa[1], 3)),
           upper = c(round(CI.95.msa[2], 3), round(CI.99.msa[2], 3)),
           row.names = c("95%", "99%"))
```

```
##     lower upper
## 95% 1.398 2.601
## 99% 1.268 2.867
```

```r
#The SNP with most significant protective effect is:
SNP.msp <- res[which.min(res$coef)]
print(SNP.msp)
```

```
##             name       coef        OR        se   p.value
## 1: rs11575839_C -0.6022542 0.5475759 0.3758156 0.1090394
```

```
# The 95% and 99% confidence intervals of corresponding odds ratio are:
CI.95.msp <- exp(SNP.msp$coef + qnorm(0.975) * SNP.msp$se * c(-1, 1))
CI.99.msp <- exp(SNP.msp$coef + qnorm(0.995) * SNP.msp$se * c(-1, 1))
data.frame(lower = c(round(CI.95.msp[1], 3), round(CI.99.msp[1], 3)),
           upper = c(round(CI.95.msp[2], 3), round(CI.99.msp[2], 3)),
           row.names = c("95%", "99%"))
```

```
##     lower upper
## 95% 0.262 1.144
## 99% 0.208 1.442
```

Problem 2.d (4points)

Merge your GWAS results with the table of gene names provided in file GDM.annot.txt (available from the accompanying zip folder on Learn). For SNPs that have p-value $< 10^{-4}$ (hit SNPs) report SNP name, effect allele, chromosome number and corresponding gene name. Separately, report for each 'hit SNP' tahe names of the genes that are within a 1Mb window from the SNP position on the chromosome. Note: That's genes that fall within +/- 1,000,000 positions using the 'pos' column in the dataset.

Solution:

```r
# read dataset into R
gdm.annot.dt <- data.table(read.table("assignment2/GDM.annot.txt",
                                      sep = "\t", header = TRUE))

# merge previous results with gene names in gdm.annot.dt
# before merging, split 'name' in res to SNP names and effect allele
res$snp <- sapply(res$name, function(x) strsplit(x, split = "_")[[1]][1])
res$allele <- sapply(res$name, function(x) strsplit(x, split = "_")[[1]][2])

# now merge the datasets by 'snp'
res.merged <- merge(res, gdm.annot.dt, by = "snp")
```

```r
# find the hit SNPs and report
hit.SNP <- res.merged[which(res.merged$p.value < 1e-4)]
hit.SNP[, .(snp, allele, chrom, gene)]
```

```
##           snp allele chrom   gene
## 1: rs12243326      A    10 TCF7L2
## 2:  rs2237897      T    11  KCNQ1
```

```r
# 1Mb window for 1st SNP in hit.SNP
unique(res.merged[abs(res.merged$pos - hit.SNP$pos[1]) <= 10^6
                            & res.merged$pos != hit.SNP$pos[1], "gene"])
```

```
##      gene
## 1: TCF7L2
```

```r
# 1Mb window for 2nd SNP in hit.SNP
unique(res.merged[abs(res.merged$pos - hit.SNP$pos[2]) <= 10^6
                        & res.merged$pos != hit.SNP$pos[2], "gene"])
```

```
##        gene
## 1:       TH
## 2:    KCNQ1
## 3: CACNA2D4
## 4:     SMG6
```

Problem 2.e (8 points)

Build a weighted genetic risk score that includes all SNPs with p-value $< 10^{-4}$, a score with all SNPs with p-value $< 10^{-3}$, and a score that only includes SNPs on the FTO gene (hint: ensure that the ordering of SNPs is respected). Add the three scores as columns to the gdm.dt data table. Fit the three scores in separate logistic regression models to test their association with gestational diabetes, and for each report odds ratio, 95% confidence interval and p-value.

Solution:

```r
# subset of SNPs with p-value < 1e-3
p3.SNP <- res.merged[which(res.merged$p.value < 1e-3)]

# subset of SNPs on the FTO gene
FTO.SNP <- res.merged[which(res.merged$gene == 'FTO')]
```

```r
# weighted genetic risk score for SNPs in hit.SNP
```

Problem 2.f (4 points)

File GDM.test.txt (available from the accompanying zip folder on Learn) contains genotypes of another 40 pregnant women with and without gestational diabetes (assume that the reference allele is the same one that was specified in file GDM.raw.txt). Read the file into variable gdm.test. For the set of patients in gdm.test, compute the three genetic risk scores as defined in problem 2.e using the same set of SNPs and corresponding weights. Add the three scores as columns to gdm.test (hint: use the same columnnames as before).

```r
# read data into R
gdm.test <- read.table("assignment2/GDM.test.txt", header = TRUE)
```

Problem 2.g (4 points)

Use the logistic regression models fitted in problem 2.e to predict the outcome of patients in gdm.test. Compute the test log-likelihood for the predicted probabilities from the three genetic risk score models.

```
# Enter code here.
```

Problem 2.h (4points)

File GDM.study2.txt (available from the accompanying zip folder on Learn) contains the summary statistics from a different study on the same set of SNPs. Perform a meta-analysis with the results obtained in problem 2.c (hint: remember that the effect alleles should correspond) and produce a summary of the meta-analysis results for the set of SNPs with meta-analysis p-value $< 10^{-4}$ sorted by increasing p-value.

```r
# read data into R
gdm.study2 <- read.table("assignment2/GDM.study2.txt", header = TRUE)
```

Problem 3 (33 points)

File nki.csv (available from the accompanying zip folder on Learn) contains data for 144 breast cancer patients. The dataset contains a binary outcome variable ("Event", indicating the insurgence of further complications after operation), covariates describing the tumour and the age of the patient, and gene expressions for 70 genes found to be prognostic of survival.

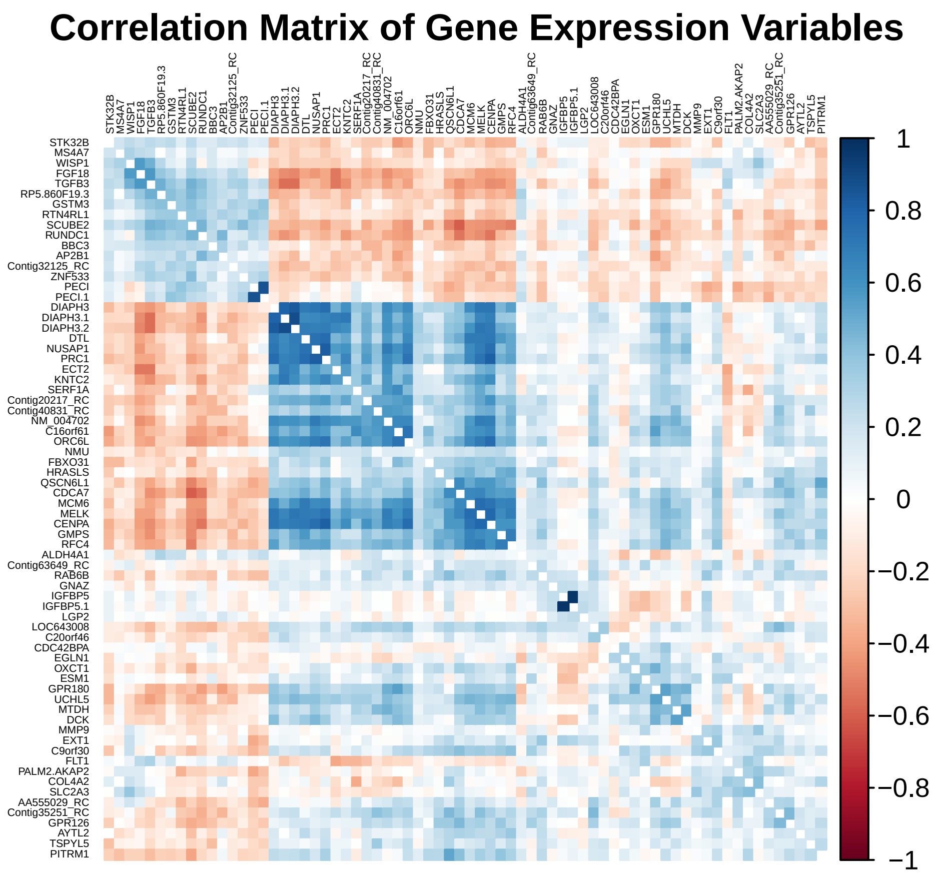
Problem 3.a (6 points)

Compute the matrix of correlations between the gene expression variables, and display it so that a block structure is highlighted. Discuss what you observe. Write some code to identify the unique pairs of (distinct) variables that have correlation coefficient greater than 0.80 in absolute value and report their correlation coefficients.

Solution:

```r
# read data into R
nki <- read.csv("assignment2/nki.csv", header = TRUE)
```

```r
# subset of gene expression variables
gene.expression <- nki[, -c(1 : 6)]

# correlations matrix
gene.expression.corr <- cor(gene.expression)

# visualize the correlations using a heatmap
corrplot(gene.expression.corr, order = "hclust", method = "color",
         title = "Correlation Matrix of Gene Expression Variables",
         tl.col = "black", tl.cex = 0.3, diag = FALSE, mar = c(0, 0, 1, 0))
```



```r
# find the unique pairs of distinct variables with correlation > 0.8
# initialize an empty data frame
corr.08 <- data.frame()

# loop to get pairs with correlation > 0.8 except identity pairs
for (i in 2:ncol(gene.expression.corr)){
  for (j in 1:(i-1)){
    pair <- paste(rownames(gene.expression.corr)[i],
                  colnames(gene.expression.corr)[j], sep = ", ")
    corr <- gene.expression.corr[i, j]
    if (abs(corr) > 0.8){
      df <- data.frame(pair = pair, correlation = corr)
    corr.08 <- rbind(corr.08, df)
    }
  }
}

corr.08
```

```
##                  pair correlation
## 1   DIAPH3.1, DIAPH3   0.8031368
## 2   DIAPH3.2, DIAPH3   0.8338591
## 3 DIAPH3.2, DIAPH3.1   0.8868741
## 4        PECI.1, PECI   0.8697836
## 5   IGFBP5.1, IGFBP5   0.9775030
## 6        PRC1, NUSAP1   0.8298356
## 7        CENPA, PRC1   0.8175424
```

Problem 3.b (8 points)

Run PCA (only over the columns containing gene expressions), in order to derive a patient-wise summary of all gene expressions (dimensionality reduction). Decide which components to keep and justify your decision. Test if those principal components are associated with the outcome in unadjusted logistic regression models and in models adjusted for age, estrogen receptor and grade. Justify the difference in results between unadjusted and adjusted models.

Solution:

```r
# run PCA
pca.patients <- prcomp(gene.expression, center = TRUE, scale = TRUE)
summary(pca.patients)
```

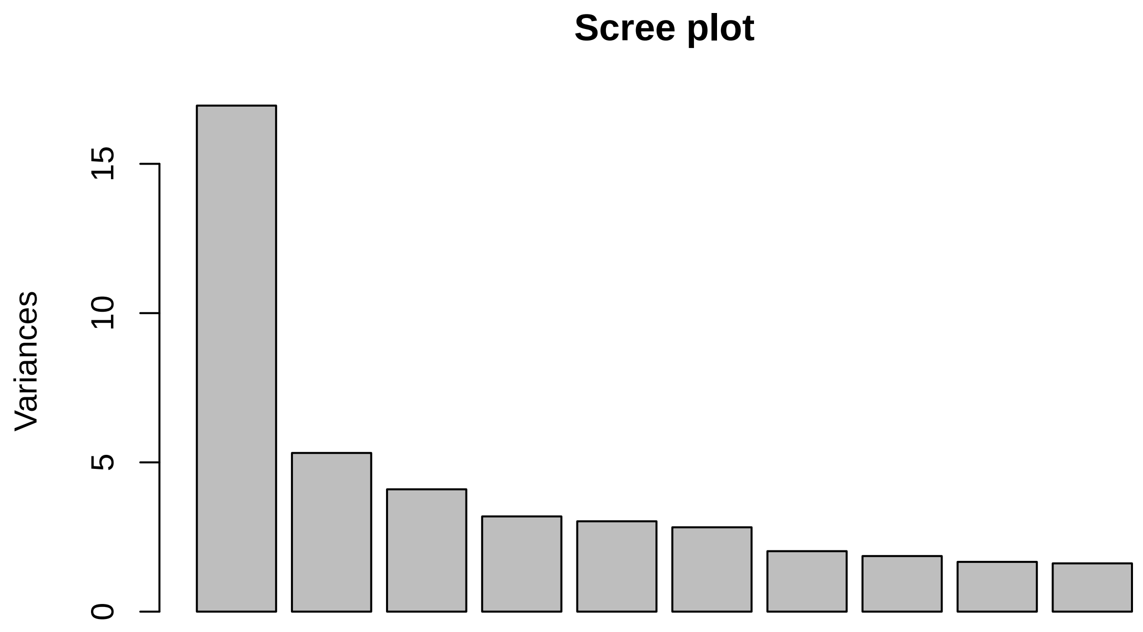
```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     4.1171 2.30541 2.02437 1.78597 1.73982 1.68091 1.42309
## Proportion of Variance 0.2422 0.07593 0.05854 0.04557 0.04324 0.04036 0.02893
## Cumulative Proportion  0.2422 0.31808 0.37662 0.42219 0.46543 0.50580 0.53473
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.36441 1.29119 1.2715 1.24741 1.18388 1.15101 1.13883
## Proportion of Variance 0.02659 0.02382 0.0231 0.02223 0.02002 0.01893 0.01853
## Cumulative Proportion  0.56132 0.58514 0.6082 0.63046 0.65049 0.66941 0.68794
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     1.09473 1.07016 1.04187 1.00234 0.99086 0.94095 0.93322
## Proportion of Variance 0.01712 0.01636 0.01551 0.01435 0.01403 0.01265 0.01244
## Cumulative Proportion  0.70506 0.72142 0.73693 0.75128 0.76531 0.77796 0.79040
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.90727 0.89675 0.88859 0.86019 0.84462 0.82782 0.82368
## Proportion of Variance 0.01176 0.01149 0.01128 0.01057 0.01019 0.00979 0.00969
## Cumulative Proportion  0.80216 0.81364 0.82492 0.83549 0.84569 0.85548 0.86517
##                           PC29    PC30    PC31    PC32    PC33    PC34   PC35
## Standard deviation     0.78694 0.75594 0.73942 0.70569 0.69414 0.67129 0.6639
## Proportion of Variance 0.00885 0.00816 0.00781 0.00711 0.00688 0.00644 0.0063
## Cumulative Proportion  0.87401 0.88218 0.88999 0.89710 0.90399 0.91042 0.9167
##                           PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.63815 0.61964 0.59947 0.58447 0.57195 0.55097 0.53820
## Proportion of Variance 0.00582 0.00549 0.00513 0.00488 0.00467 0.00434 0.00414
## Cumulative Proportion  0.92254 0.92802 0.93316 0.93804 0.94271 0.94705 0.95118
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.52029 0.51211 0.49533 0.48712 0.47079 0.44565 0.41879
## Proportion of Variance 0.00387 0.00375 0.00351 0.00339 0.00317 0.00284 0.00251
## Cumulative Proportion  0.95505 0.95880 0.96230 0.96569 0.96886 0.97170 0.97420
##                           PC50    PC51   PC52    PC53    PC54    PC55    PC56
## Standard deviation     0.40556 0.39328 0.3925 0.38502 0.36669 0.36205 0.33734
## Proportion of Variance 0.00235 0.00221 0.0022 0.00212 0.00192 0.00187 0.00163
## Cumulative Proportion  0.97655 0.97876 0.9810 0.98308 0.98500 0.98687 0.98850
##                           PC57    PC58    PC59    PC60    PC61    PC62    PC63
## Standard deviation     0.32150 0.30744 0.28898 0.28186 0.27274 0.25622 0.24118
## Proportion of Variance 0.00148 0.00135 0.00119 0.00113 0.00106 0.00094 0.00083
## Cumulative Proportion  0.98998 0.99133 0.99252 0.99365 0.99472 0.99565 0.99649
##                           PC64    PC65    PC66    PC67    PC68   PC69    PC70
## Standard deviation     0.23024 0.21442 0.19886 0.19371 0.17927 0.1677 0.09833
## Proportion of Variance 0.00076 0.00066 0.00056 0.00054 0.00046 0.0004 0.00014
## Cumulative Proportion  0.99724 0.99790 0.99846 0.99900 0.99946 0.9999 1.00000
```

The amount of variability explained by the components can be computed bearing in mind that the square root of the eigenvalues is stored in vector `sdev` of the PCA object. The variance explained by the principal components can be visualized through a scree plot.

```
# the proportion explained by the first 3 PCs
perc.expl <- pca.patients$sdev^2 / sum(pca.patients$sdev^2)
sum(perc.expl[1:3])
```

```
## [1] 0.3766231
```

```
# scree plot
screeplot(pca.patients, main = "Scree plot")
```



```
# subset of the first 3 PCs
pca.components <- pca.patients$x[, 1:3]

# logistic regressions on each PC without adjustment on other variables
fit.pc1 <- glm(nki$Event ~ pca.components[, 1],
               family = "binomial"(link = "logit"))
```

```
fit.pc2 <- glm(nki$Event ~ pca.components[, 2],
               family = "binomial"(link = "logit"))

fit.pc3 <- glm(nki$Event ~ pca.components[, 3],
               family = "binomial"(link = "logit"))

# logistic regression on each PC with adjustments
fit.pc1.adj <- glm(nki$Event ~ pca.components[, 1]
                   + nki$EstrogenReceptor + nki$Grade + nki$Age,
                   family = "binomial"(link = "logit"))

fit.pc2.adj <- glm(nki$Event ~ pca.components[, 2]
                   + nki$EstrogenReceptor + nki$Grade + nki$Age,
                   family = "binomial"(link = "logit"))

fit.pc3.adj <- glm(nki$Event ~ pca.components[, 3]
                   + nki$EstrogenReceptor + nki$Grade + nki$Age,
                   family = "binomial"(link = "logit"))
```

```
# regression coefficients and p-values in unadjusted and adjusted models
# principal component 1
data.frame(coef = c(fit.pc1$coefficients[2], fit.pc1.adj$coefficients[2]),
           p.value = c(summary(fit.pc1)$coefficients[2, 4],
                       summary(fit.pc1.adj)$coefficients[2, 4]),
           row.names = c("unadjusted", "adjuested"))
```

```
##                  coef     p.value
## unadjusted 0.11768352 0.009425039
## adjuested  0.07215917 0.272381200
```

```
# principal component 2
data.frame(coef = c(fit.pc2$coefficients[2], fit.pc2.adj$coefficients[2]),
           p.value = c(summary(fit.pc2)$coefficients[2, 4],
                       summary(fit.pc2.adj)$coefficients[2, 4]),
           row.names = c("unadjusted", "adjuested"))
```

```
##                    coef   p.value
## unadjusted -0.067166804 0.3885231
## adjuested   0.005164421 0.9550636
```

```
# principal component 3
data.frame(coef = c(fit.pc3$coefficients[2], fit.pc3.adj$coefficients[2]),
           p.value = c(summary(fit.pc3)$coefficients[2, 4],
                       summary(fit.pc3.adj)$coefficients[2, 4]),
           row.names = c("unadjusted", "adjuested"))
```

```
##                 coef    p.value
## unadjusted 0.2435485 0.00863000
## adjuested  0.2183706 0.02454557
```

```r
fit.pca.adj <- glm(nki$Event ~ pca.components[, 1] + pca.components[, 2]
                   + pca.components[, 3] + nki$EstrogenReceptor
                   + nki$Grade + nki$Age, family = "binomial"(link = "logit"))
summary(fit.pca.adj)
```

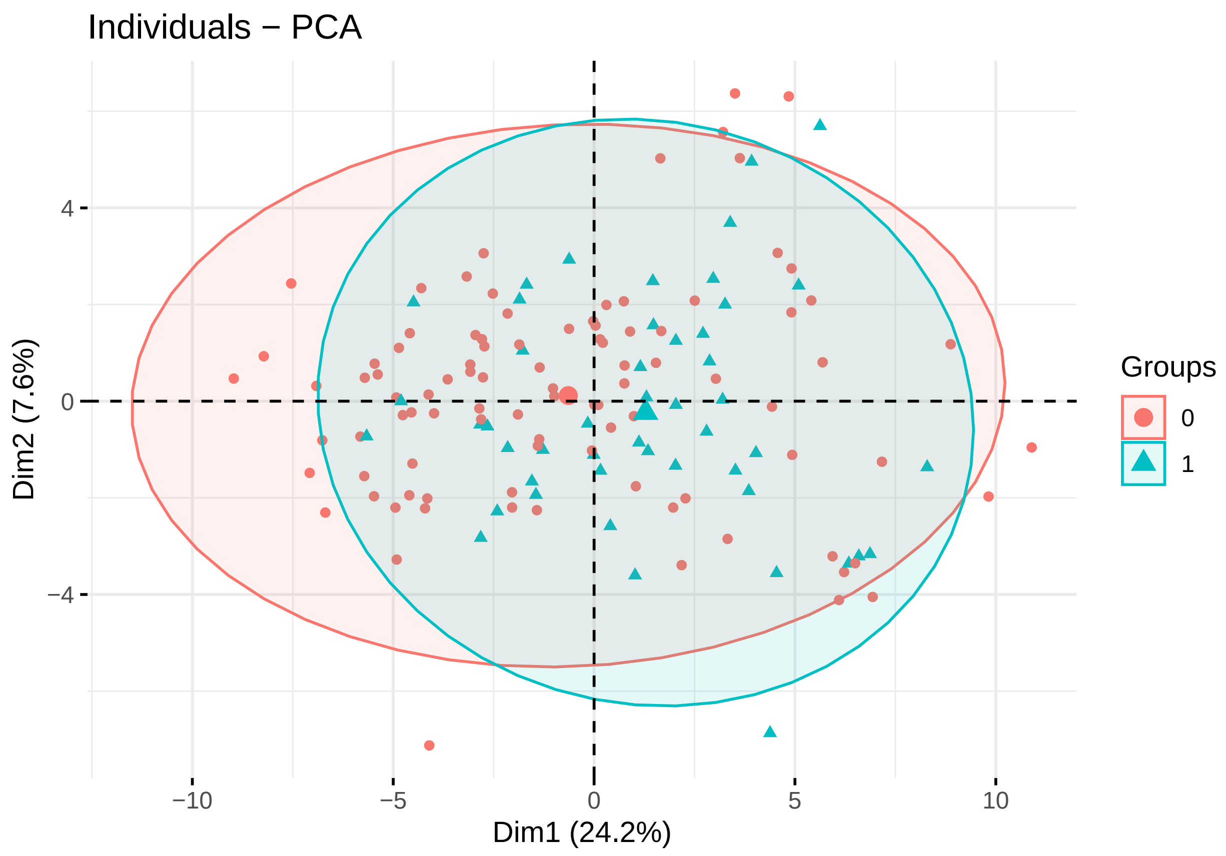
```
##
## Call:
## glm(formula = nki$Event ~ pca.components[, 1] + pca.components[,
##     2] + pca.components[, 3] + nki$EstrogenReceptor + nki$Grade +
##     nki$Age, family = binomial(link = "logit"))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8178  -0.8782  -0.5907   1.1119   2.1029
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   2.24591    1.69668   1.324   0.1856
## pca.components[, 1]           0.09426    0.07333   1.285   0.1986
## pca.components[, 2]          -0.04621    0.10305  -0.448   0.6539
## pca.components[, 3]           0.22688    0.09899   2.292   0.0219 *
## nki$EstrogenReceptorPositive  0.09780    0.69101   0.142   0.8875
## nki$GradePoorly diff          0.15228    0.48885   0.312   0.7554
## nki$GradeWell diff           -0.63836    0.55341  -1.153   0.2487
## nki$Age                      -0.06788    0.03632  -1.869   0.0616 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 183.32  on 143  degrees of freedom
## Residual deviance: 162.25  on 136  degrees of freedom
## AIC: 178.25
##
## Number of Fisher Scoring iterations: 4
```

Problem 3.c (8 points)

Use plots to compare with the correlation structure observed in problem 2.a and to examine how well the dataset may explain your outcome. Discuss your findings and suggest any further steps if needed.

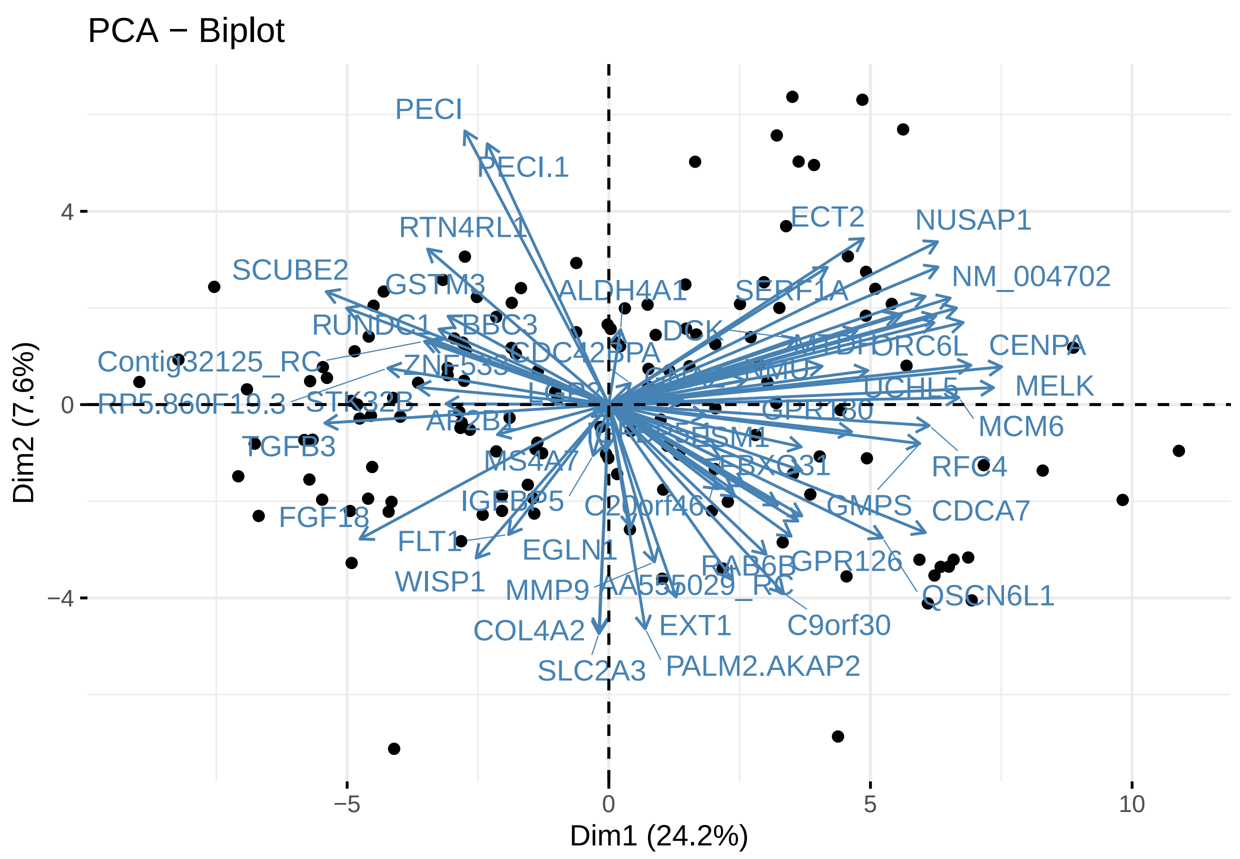
Solutions:

```
# PC1 vs PC2
fviz_pca_ind(pca.patients, geom = 'point',
             habillage = nki$Event, addEllipses = TRUE)
```

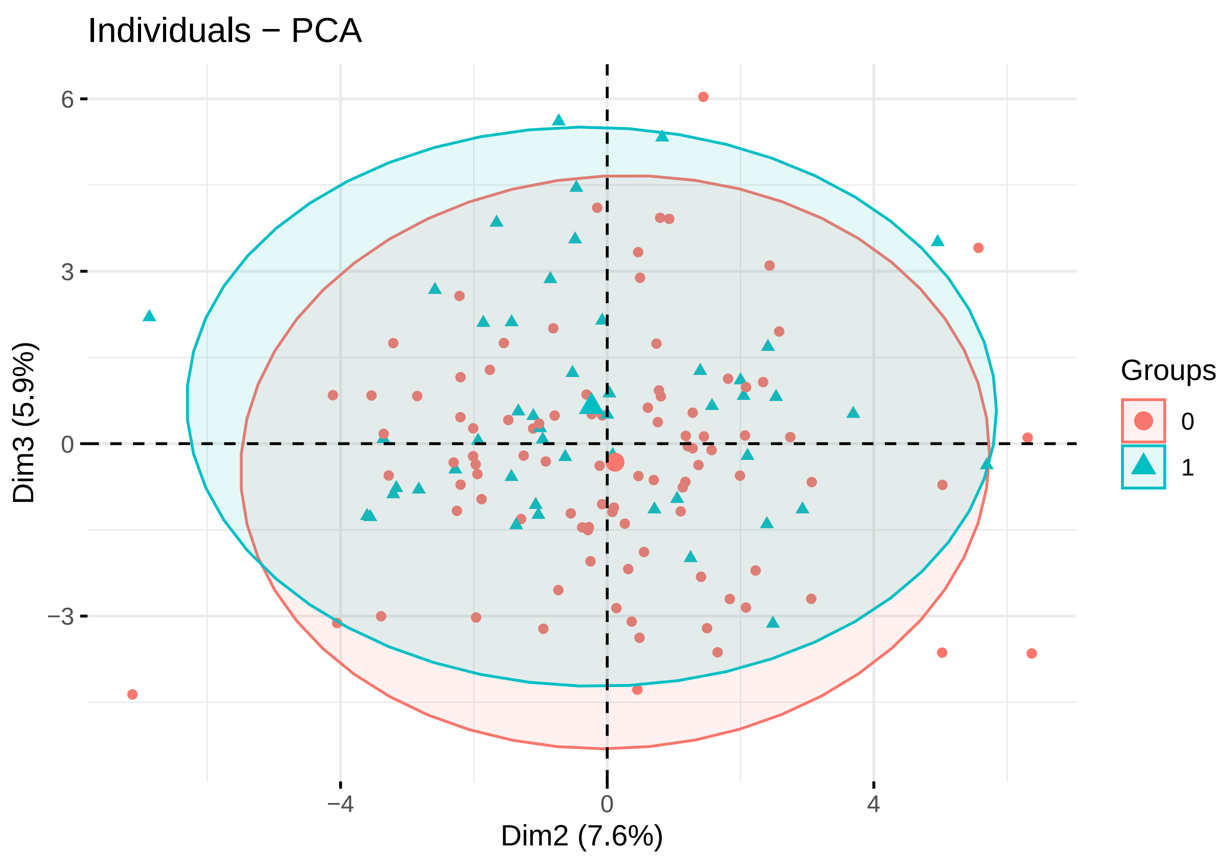


```
fviz_pca_biplot(pca.patients, geom = 'point', repel = TRUE)
```

```
## Warning: ggrepel: 17 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

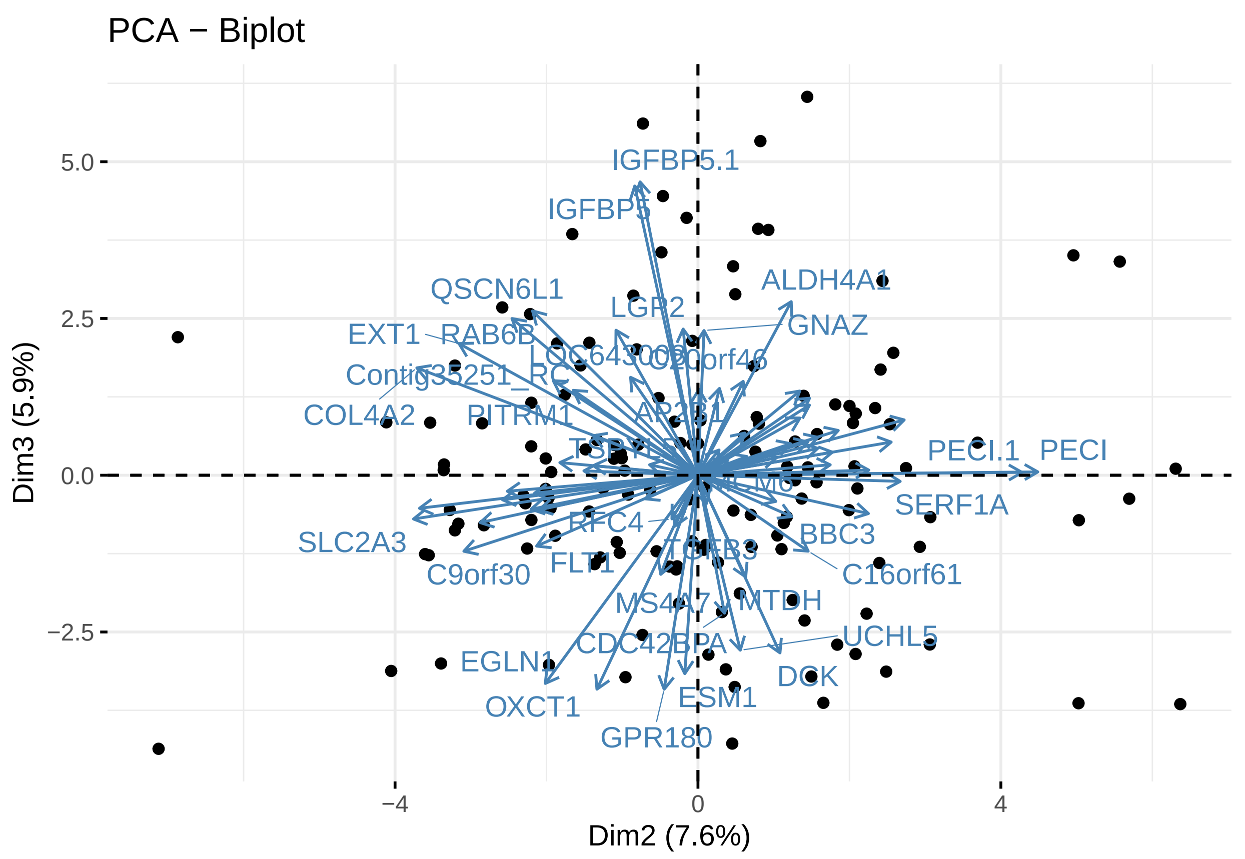


```
# PC2 vs PC3
fviz_pca_ind(pca.patients, geom = 'point', axes = c(2, 3),
             habillage = nki$Event, addEllipses = TRUE)
```



```
fviz_pca_biplot(pca.patients, geom = 'point', axes = c(2, 3), repel = TRUE)
```

```
## Warning: ggrepel: 35 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```


PCA − Biplot
Dim3 (5.9%)
Dim2 (7.6%)
5.0
2.5
0.0
−2.5
−4
0
4
IGFBP5.1
IGFBP5
QSCN6L1
LGP2
ALDH4A1
GNAZ
EXT1
RAB6B
LOC643008
C20orf46
Contig35251_RC
COL4A2
PITRM1
AP2B1
TSPYL5
PECI.1
PECI
SERF1A
SLC2A3
RFC4
BBC3
C9orf30
FLT1
TGFB3
C16orf61
MS4A7
MTDH
CDC42BPA
UCHL5
EGLN1
DCK
OXCT1
ESM1
GPR180

Problem 3.d (11 points)

Based on the models we examined in the labs, fit an appropriate model with the aim to provide the most accurate prognosis you can for patients. Discuss and justify your decisions.

Solution:

```r
# helper function in Lab4
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)
                options(old.opts)

                ## remove the intercept column, as glmnet will add one by default
                x <- x[, -match("(Intercept)", colnames(x))]
                return(x)
}
```

```r
# define x and y
x.mat <- prepare.glmnet(nki, ~ . - Event)
y.mat <- nki$Event

# create training and testing sets
set.seed(984065)
train.idx <- createDataPartition(y = nki$Event, p = 0.7)$Resample1
train.x.mat <- x.mat[train.idx, ]
test.x.mat <- x.mat[-train.idx, ]
train.y.mat <- y.mat[train.idx]
test.y.mat <- y.mat[-train.idx]
```

```r
# Lasso regressions using CV
fit.cv.Lasso <- cv.glmnet(train.x.mat, train.y.mat,
                          family = "binomial", type.measure = "auc")
fit.cv.Lasso.gene <- cv.glmnet(train.x.mat[, -c(1:6)], train.y.mat,
                               family = "binomial", type.measure = "auc")

# AUCs with best lambda
Lasso.auc <- fit.cv.Lasso$cvm[fit.cv.Lasso$index[1]]
Lasso.gene.auc <- fit.cv.Lasso.gene$cvm[fit.cv.Lasso.gene$index[1]]
```

```r
# Ridge regression using CV
fit.cv.Ridge <- cv.glmnet(train.x.mat, train.y.mat, alpha = 0,
                          family = "binomial", type.measure = "auc")
fit.cv.Ridge.gene <- cv.glmnet(train.x.mat[, -c(1:6)], train.y.mat, alpha = 0,
                               family = "binomial", type.measure = "auc")

# AUCs with best lambda
Ridge.auc <- fit.cv.Ridge$cvm[fit.cv.Ridge$index[1]]
Ridge.gene.auc <- fit.cv.Ridge.gene$cvm[fit.cv.Ridge.gene$index[1]]
```

```
# AUCs of Lasso and Ridge regressions on testing set
pred.Lasso.test <- predict(fit.cv.Lasso, newx = test.x.mat,
                           s = fit.cv.Lasso$lambda.min, type = "response")
pred.Lasso.gene.test <- predict(fit.cv.Lasso.gene, newx = test.x.mat[, -c(1:6)],
                                s = fit.cv.Lasso$lambda.min, type = "response")
pred.Ridge.test <- predict(fit.cv.Ridge, newx = test.x.mat,
                           s = fit.cv.Ridge$lambda.min, type = "response")
pred.Ridge.gene.test <- predict(fit.cv.Ridge.gene, newx = test.x.mat[, -c(1:6)],
                                s = fit.cv.Ridge$lambda.min, type = "response")

Lasso.test.auc <- roc(test.y.mat ~ pred.Lasso.test)$auc
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
Lasso.gene.test.auc <- roc(test.y.mat ~ pred.Lasso.gene.test)$auc
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
Ridge.test.auc <- roc(test.y.mat ~ pred.Ridge.test)$auc
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
Ridge.test.gene.auc <- roc(test.y.mat ~ pred.Ridge.gene.test)$auc
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
data.frame(train.AUC = c(Lasso.auc, Lasso.gene.auc, Ridge.auc, Ridge.gene.auc),
           modelsize = c(fit.cv.Lasso$nzero[fit.cv.Lasso$index[1]],
                         fit.cv.Lasso.gene$nzero[fit.cv.Lasso.gene$index[1]],
                         fit.cv.Ridge$nzero[fit.cv.Ridge$index[1]],
                         fit.cv.Ridge.gene$nzero[fit.cv.Ridge.gene$index[1]]),
           test.AUC = c(Lasso.test.auc, Lasso.gene.test.auc,
                        Ridge.test.auc, Ridge.test.gene.auc),
           row.names = c("Lasso", "Lasso (gene only)",
                         "Ridge", "Ridge (gene only)"))
```

```
##                   train.AUC modelsize  test.AUC
## Lasso             0.6835691         6 0.7846154
## Lasso (gene only) 0.7349914        37 0.7384615
## Ridge             0.6784418        76 0.8897436
## Ridge (gene only) 0.7029742        70 0.8179487
```