Assignment 2

Biomedical Data Science (MATH11174), 22/23, Semester 2

Pablo Ruiz Barnada

April 6, 2023

Due on Thursday, 6th of April 2023, 5:00pm

! Pay Attention

The assignment is marked out of 100 points, and will contribute to ***30%*** of your final mark. The aim of this assignment is to produce a precise report in biomedical studies with the help of statistical and machine learning. Please complete this assignment using **Quarto/Rmarkdown file and render/knit this document only in PDF format** (rendering while solving the questions will prevent sudden panic before submission!). Submit using the **gradescope link on Learn** and ensure that **all questions are tagged accordingly**. You can simply click render on the top left of Rstudio (`Ctrl+Shift+K`). If you cannot render/knit to PDF directly, open **Terminal** in your RStudio (`Alt+Shift+R`) and type `quarto tools install tinytex`, otherwise please follow this link. If you have any code that does not run you will not be able to render nor knit the document so comment it as you might still get some grades for partial code.

Codes that are **clear and reusable will be rewarded**. Codes without proper indentation, choice of variable identifiers, **comments**, efficient code, etc will be penalised. 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 when required to make it easier to follow your code and reasoning. Ensure that all answers containing multiple values should be presented and formatted only with `kable()` and `kable_styling()` otherwise penalised (**no use of `print()` or `cat()`**). All plots must be displayed with clear title, label and legend otherwise penalised.

This is an **individual assignment**, and **no public discussions** will be allowed. If you have any question, please ask on Piazza by specifying your `Post to` option to `instructors`. To join Piazza, please follow this link.

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 and Lasso regression model which use 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 maximises the AUC and report the λ for both ridge and Lasso regression using `kable()`.
- ***Note : there is no need to use the `prepare.glmnet()` function from lab 4, using `as.matrix()` with the required columns is sufficient.***

```r
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment2")
wdbc2 <- fread("wdbc2.csv")
# head(wdbc2, 5)
# summary(wdbc2)
# colnames(wdbc2)
dim(wdbc2)
```

```
[1] 569  32
```

```r
sum(is.na(wdbc2))
```

```
[1] 0
```

First, we load the data and print its first 5 rows. We can analyse some basic statistics of each column with the function summary. Then, we check how big the dataset is (569 rows and 32 variables) and make sure there is no missing data.

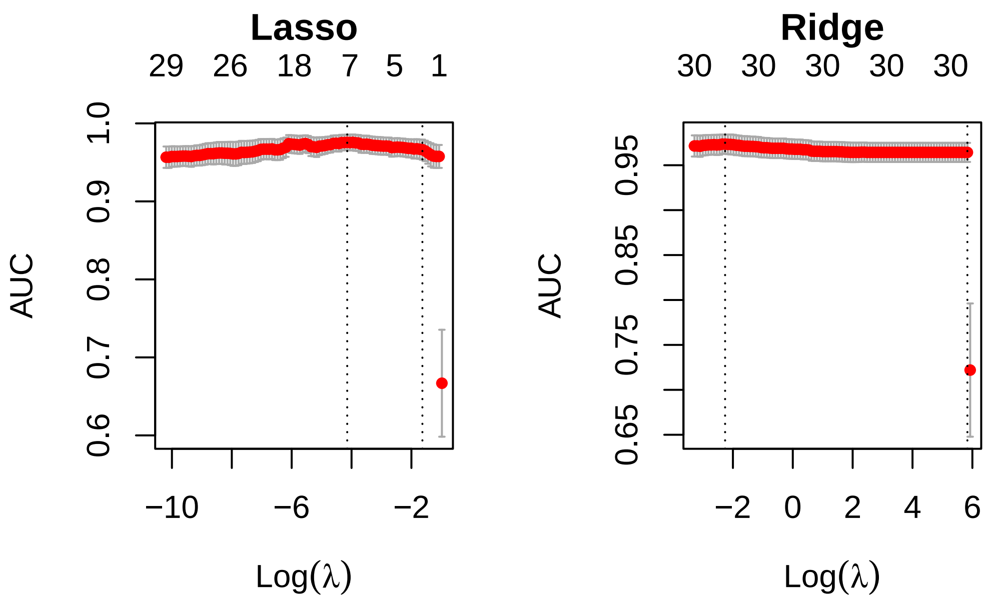
```r
# For reproducibility
set.seed(984065)

# Obtain the rows that will take part of the training set
```

```r
train.idx <- createDataPartition(wdbc2$diagnosis, p = 0.7)$Resample1

# Define train and test sets, for x and y
wdbc2.train <- wdbc2[train.idx,]
wdbc2.test <- wdbc2[-train.idx,]
```

```r
# Create design matrix to deal with factor variables
prepare.glmnet <- function(data, formula=~ .) {
  old.opts <- options(na.action='na.pass')
  x <- model.matrix(formula, data)
  options(old.opts)
  x <- x[, -match("(Intercept)", colnames(x))]
  return(x)
}

# Prepare train and test sets by changing factor variables
train1 <- prepare.glmnet(wdbc2.train, formula=~ .)
test1 <- prepare.glmnet(wdbc2.test, formula=~ .)

# Change name of variable
colnames(train1)[2] <- "malignant"
colnames(test1)[2] <- "malignant"

x.train1 <- train1[, -2]
x.test1 <- test1[, -2]
y.train1 <- train1[, 2]
y.test1 <- test1[, 2]

# nFolds <- 3
# foldid <- sample(rep(seq(nFolds), length.out = nrow(x.train1)))
# Fit models
fit.lasso1 <- cv.glmnet(x.train1[, -1], y.train1, family = "binomial",
                        type.measure = "auc")
fit.ridge1 <- cv.glmnet(x.train1[, -1], y.train1, alpha = 0, family = "binomial",
                        type.measure = "auc")
```

After defining the training and test sets, we transform the factor variables so they can be passed to the model. In this case, the only factor was the diagnostic of the tumour, which could be either malignant or benignant. Once this was done, we fitted the Lasso and Ridge models. The Ridge model is the same as the Lasso, but with the parameter `alpha = 0`.

```r
#Plot trajectories
par(mfrow=c(1,2), mar=c(4,4,5,2))
plot(fit.lasso1, main="Lasso")
plot(fit.ridge1, main="Ridge")
```



Above are printed the fitted regressions' AUC scores for various λ, with grey vars indicating standard errors. The objective is to find the λ for each model that maximises the AUC, as this would imply that the regression is more accurate. In each plot, two lines can be observed: the leftest one is the λ that minimises the error (maximises the AUC), and the rightest one corresponds to the largest λ such that the error lies within one standard error of distance from the minimum. The red curves depend on the choice of the error measure used in cross-validation.

```r
lambda.min.dt <- data.frame(Lasso = c(fit.lasso1$lambda.min),
                            Ridge = c(fit.ridge1$lambda.min))

rownames(lambda.min.dt) <- c("Minimum")

kable(lambda.min.dt, format = "latex",
      caption = "Penalty parameter for Lasso and Ridge regressions",
      align = "c", digits = 3) |>
```

```r
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 1: Penalty parameter for Lasso and Ridge regressions

	Lasso	Ridge
Minimum	0.016	0.104

The table contains information on the λ that lead to maximum AUC scores (minimum error), for both the Lasso and the Ridge regressions. As Ridge has no penalisations, the recorded value is really far away from the minimum. Clearly, the λ for ridge is higher, with a difference of magnitude 10.

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** that contains the corresponding λ, AUC and model size.
- Use 3 significant figures for floating point values and comment on these results.
- ***Note : The AUC values are stored in the field called*** `cvm`.

```r
# Create dataframes
lasso.dt <- data.frame(lambda_min = c(signif(fit.lasso1$lambda.min, 3),
                                      signif(fit.lasso1$cvm[fit.lasso1$index][1], 3),
                                      fit.lasso1$nzero[fit.lasso1$index][1]),
                       lambda_1se = c(signif(fit.lasso1$lambda.1se, 3),
                                      signif(fit.lasso1$cvm[fit.lasso1$index][2], 3),
                                      fit.lasso1$nzero[fit.lasso1$index][2]))


ridge.dt <- data.frame(lambda_min = c(signif(fit.ridge1$lambda.min, 3),
                                      signif(fit.ridge1$cvm[fit.ridge1$index][1], 3),
                                      fit.ridge1$nzero[fit.ridge1$index][1]),
                       lambda_1se = c(signif(fit.ridge1$lambda.1se, 3),
                                      signif(fit.ridge1$cvm[fit.ridge1$index][2], 3),
                                      fit.ridge1$nzero[fit.ridge1$index][2]))

# Change column and row names
rownames(lasso.dt) <- c("lambda", "AUC", "Number of parameters")
colnames(lasso.dt)[1:2] <- c("lambda (minimum)", "lambda (1 st. error away)")

rownames(ridge.dt) <- c("lambda", "AUC", "Number of parameters")
```

```r
colnames(ridge.dt)[1:2] <- c("lambda (minimum)", "lambda (1 st. error away)")


# Output table
kable(lasso.dt, format = "latex",
      caption = "General information about Lasso regression",
      align = "c") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 2: General information about Lasso regression

	lambda (minimum)	lambda (1 st. error away)
lambda	0.0159	0.195
AUC	0.9760	0.966
Number of parameters	7.0000	2.000

```r
kable(ridge.dt, format = "latex",
      caption = "General information about Ridge regression",
      align = "c") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 3: General information about Ridge regression

	lambda (minimum)	lambda (1 st. error away)
lambda	0.104	342.000
AUC	0.973	0.964
Number of parameters	30.000	30.000

The two previous tables contain information about the Lasso and Ridge regressions respectively, collecting information on the λ that minimises the error and the λ that is one standard error away from the minimum, and the AUC and the size of the models that are obtained from those λ. First, we will focus the analysis on the Lasso regression. It can be observed in the table that the model achieves the best results with $\lambda_{min} = 0.0159$, while $\lambda_{1se} = 0.195$ is one standard error away from λ_{min}. Despite the difference in λ, the AUC values are really similar, 0.976 vs 0.966. In fact, the main difference between these two Lasso models is that the optimal takes 7 parameters, while the one defined by λ_{1se} only uses 2.

With regards to the Ridge model, we see an incredibly large difference in the λ defining each model, with $\lambda_{min} = 0.104$ and $\lambda_{1se} = 342$. In fact, the AUC and the number of parameters are exactly the same for each model. Of course, the size of the models had to be the same, as

Ridge always includes all the variables in the final model.

Comparing Lasso and Ridge models, it can be seen that Lasso achieves slightly higher performance (higher AUC), and takes many fewer parameters. The fundamental difference between these two approaches is that Lasso can reduce the influence of some variables to exactly 0 in the final model, while Ridge will always include all variables, even if their effect is really small. Despite the small difference in AUC, it could be argued that the Lasso model is preferred as it eliminates the influence of some features that are likely to be irrelevant in diagnosing the type of tumour.

Problem 1.c (7 points)

- Perform both backward (we denote this as **model B**) and forward (**model S**) stepwise selection on the same training set derived in **problem 1.a**. Mute all the trace by setting `trace = FALSE`.
- Report the variables selected and their standardised regression coefficients in increasing order of the absolute value of their standardised regression coefficient.
- Discuss the results and how the different variables entering or leaving the model influenced the final result.
- ***Note : You can mute the warning by assigning {r warning = FALSE} for the chunk title***

```r
# Remove ID from dataframes
train1.df <- as.data.frame(train1[, -1])
test1.df <- as.data.frame(test1[, -1])

# Define null and full models
full.model <- glm(malignant ~ ., data = train1.df, family = "binomial")
null.model <- glm(malignant ~ 1, data = train1.df, family = "binomial")

# Do stepwise selection
modelB <- stepAIC(full.model, direction="back", trace = FALSE,
                  scope=list(lower=null.model)) # backward
modelS <- stepAIC(null.model, direction="forward", trace = FALSE,
                  scope=list(upper=full.model)) #Forward

summary(modelB)
```

```
Call:
glm(formula = malignant ~ perimeter + area + concavity + concavepoints +
    radius.stderr + texture.stderr + compactness.stderr + concavity.stderr +
    fractaldimension.stderr + radius.worst + texture.worst +
    area.worst + fractaldimension.worst, family = "binomial",
    data = train1.df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5080  -0.0798  -0.0049   0.0215   3.3383

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -6.604e+01  1.113e+01  -5.934 2.97e-09 ***
perimeter                9.504e-02  6.832e-02   1.391 0.164204
area                    -7.281e-03  4.999e-03  -1.456 0.145269
concavity               -6.572e+01  2.138e+01  -3.074 0.002115 **
concavepoints            1.420e+02  3.954e+01   3.590 0.000330 ***
radius.stderr            1.875e+01  5.675e+00   3.304 0.000953 ***
texture.stderr          -2.569e+00  1.278e+00  -2.010 0.044390 *
compactness.stderr      -1.915e+02  6.624e+01  -2.890 0.003850 **
concavity.stderr         2.293e+02  6.504e+01   3.526 0.000422 ***
fractaldimension.stderr -1.204e+03  5.166e+02  -2.331 0.019733 *
radius.worst             3.493e+00  9.654e-01   3.619 0.000296 ***
texture.worst            4.570e-01  1.119e-01   4.084 4.42e-05 ***
area.worst              -2.991e-02  6.295e-03  -4.750 2.03e-06 ***
fractaldimension.worst   1.700e+02  5.604e+01   3.033 0.002421 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 525.172  on 398  degrees of freedom
Residual deviance:  82.351  on 385  degrees of freedom
AIC: 110.35

Number of Fisher Scoring iterations: 9
```

```r
summary(modelS)
```

```

Call:
glm(formula = malignant ~ concavepoints.worst + radius.worst +
    area + texture + radius.stderr + area.worst + perimeter.stderr +
    symmetry.worst + symmetry.stderr + compactness.worst + concavity.worst +
    symmetry + smoothness, family = "binomial", data = train1.df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2367  -0.1085  -0.0094   0.0270   3.2518

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -5.446e+01  1.046e+01  -5.206 1.93e-07 ***
concavepoints.worst  3.551e+01  1.574e+01   2.256 0.024062 *
radius.worst         3.590e+00  7.046e-01   5.095 3.49e-07 ***
area                -4.680e-03  2.851e-03  -1.641 0.100699
texture              3.844e-01  9.166e-02   4.193 2.75e-05 ***
radius.stderr        1.839e+01  5.562e+00   3.306 0.000947 ***
area.worst          -2.747e-02  5.253e-03  -5.230 1.69e-07 ***
perimeter.stderr    -7.265e-01  6.268e-01  -1.159 0.246418
symmetry.worst       4.064e+01  1.326e+01   3.064 0.002182 **
symmetry.stderr     -1.731e+02  8.063e+01  -2.147 0.031754 *
compactness.worst   -1.554e+01  5.493e+00  -2.830 0.004659 **
concavity.worst      8.874e+00  3.852e+00   2.304 0.021243 *
symmetry            -4.164e+01  2.247e+01  -1.853 0.063816 .
smoothness           5.734e+01  3.759e+01   1.526 0.127123
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 525.172  on 398  degrees of freedom
Residual deviance:  92.228  on 385  degrees of freedom
AIC: 120.23

Number of Fisher Scoring iterations: 8
```

Above are printed the summaries of models B and S, obtained through stepwise selection of the features. Some deeper analysis is performed in the next cells.

```r
# Obtain the variables in each model, and their standard deviation
varsB <- c()
sdB <- c()
for (i in (attr(terms(modelB), "term.labels"))){
  varsB <- append(varsB, i)
}
for (var in varsB){
  sdB <- append(sdB, sd(train1[, var]))
}

varsS <- c()
sdS <- c()

for (i in (attr(terms(modelS), "term.labels"))){
  varsS <- append(varsS, i)
}
for (var in varsS){
  sdS <- append(sdS, sd(train1[, var]))
}

# standardised regression coefficients
beta.standardisedB <- coef(modelB)[-1] * sdB
beta.standardisedS <- coef(modelS)[-1] * sdS
```

As the variables we passed to the models were each measured on their own scale, their relative effect on the outcome estimation was difficult to compare. Thus, we standardised their regression coefficients by multiplying each of them by the standard deviation of their corresponding predictor. The results obtained were the following:

```r
# Define dataframes with required information
betaB <- data.frame(varsB, beta.standardisedB)
betaS <- data.frame(varsS, beta.standardisedS)

# Sort as required
betaB.df <- betaB %>% arrange(abs(beta.standardisedB))
betaS.df <- betaS %>% arrange(abs(beta.standardisedS))

# Set column names of dataset
colnames(betaB.df) <- c("Variable", "Model B Standardised Coefficients")
colnames(betaS.df) <- c("Variable", "Model S Standardised Coefficients")

# Print tables
kable(betaB.df, digits = 3,
      caption = "Standardised Regression Coefficients - Backward selection") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 4: Standardised Regression Coefficients - Backward selection

	Variable	Model B Standardised Coefficients
texture.stderr	texture.stderr	-1.558
perimeter	perimeter	2.270
area	area	-2.597
texture.worst	texture.worst	2.834
fractaldimension.worst	fractaldimension.worst	2.885
fractaldimension.stderr	fractaldimension.stderr	-3.109
compactness.stderr	compactness.stderr	-3.226
radius.stderr	radius.stderr	4.877
concavity	concavity	-4.963
concavepoints	concavepoints	5.500
concavity.stderr	concavity.stderr	8.280
radius.worst	radius.worst	17.144
area.worst	area.worst	-17.282

```r
kable(betaS.df, digits = 3,
      caption = "Standardised Regression Coefficients - Forward selection") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 5: Standardised Regression Coefficients - Forward selection

	Variable	Model S Standardised Coefficients
smoothness	smoothness	0.778
symmetry	symmetry	-1.138
symmetry.stderr	symmetry.stderr	-1.332
perimeter.stderr	perimeter.stderr	-1.358
texture	texture	1.655
area	area	-1.669
concavity.worst	concavity.worst	1.785
compactness.worst	compactness.worst	-2.310
symmetry.worst	symmetry.worst	2.311
concavepoints.worst	concavepoints.worst	2.350
radius.stderr	radius.stderr	4.783
area.worst	area.worst	-15.875
radius.worst	radius.worst	17.617

Each table shows the standardise coefficients of a model. The first table presents the coefficients of model B (backward selection), and we can see that the most influential variables (in relative terms) are `area.worst` and `radius.worst`. In particular, a standard deviation in the measures of each of these variables has more than double the effect on the prediction than such a deviation in the next more influential variable, `concavity.stderr`. It is worth noting that a higher `area.worst` is negatively correlated to the tumour being malignant, while `radius.worst` (and `concavity.stderr`) are directly correlated to the tumour being malignant. The feature `texture.stdrr` is the variable with the least effect on the diagnostic, and it is negatively correlated with a malignant diagnosis.

With regards to model S (forward selection), it can be observed that `radius.worst` and `area.worst` are again the variables for which a one-standard deviation has larger effect in the prediction of the diagnosis. In particular, a higher `radius.worst` is associated to more chances of a malignant tumour, while `area.worst` is associated with higher chances of a benignant tumour. The next most influential variable in this model is `radius.stderr` , but it has less than a quarter of the effect of `radius.worst` on the diagnosis. The weakest feature towards the diagnosis is `smoothness` in this case, and it is directly associated with malignant tumours.

By comparing the two tables, it can be seen that the differences in the models arise from the variables with a smaller effect on the outcome, not from the most influential (which, as we have mentioned before, are the same for both models, and have similar effects on the outcome estimation). In fact, despite both models taking the same number of variables (13), only four of them appear in both models (`area`, `area.worst`, `radius.worst`, `radius.stderr`), and unsurprisingly, they have some of the largest effect in the diagnostic. Aside from these, there are nine variables in each model (model B: `concavity.stderr`, `concavepoints`, `concavity`, `compactness.stderr`, `fractaldimension.stderr`, `fractaldimension.worst`, `texture.worst`, `perimeter`, `texture.stderr`; model S: `concavepoints.worse`, `symmetry.worst`, `compactness.worst`, `concavity.worst`, `texture`, `perimeter.stderr`, `symmetry.stderr`, `symmetry`, `smoothness`) that do not appear in the other model. All these variables, even if their effect is not extremely large, add up to the differences found between each model.

Problem 1.d (3 points)

- Compare the goodness of fit of **model B** and **model S**
- Interpret and explain the results you obtained.
- Report the values using `kable()`.

```r
# Deviance test to compare to null model
pb <- signif(pchisq(modelB$null.deviance - modelB$deviance,
                df = length(modelB$coefficients) - 1,
                lower.tail = FALSE), 3)
ps <- signif(pchisq(modelS$null.deviance - modelS$deviance,
                df = length(modelS$coefficients) - 1,
                lower.tail = FALSE), 3)
```

```

# Create dataframe with all values
gof.df <- data.frame(modelb = c(pb, round(summary(modelB)$aic,2)),
                     models = c(ps, round(summary(modelS)$aic,2)))

# Change names of rows and columns for the table
rownames(gof.df) <- c("p-value - Deviance test", "AIC")
colnames(gof.df)[1:2] <- c("Model B", "Model S")


# Generate table
kable(gof.df, format = "latex",
      caption = "Goodness-of-fit of models",
      align = "c") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 6: Goodness-of-fit of models

	Model B	Model S
p-value - Deviance test	0.00	0.00
AIC	110.35	120.23

First, we test whether models B and S are significantly different to the null model with a deviance test. The null model in both cases is "the model is the same as the null model", while the alternative hypothesis is "the model is different to the null". For either model B or S, the resulting p-value is practically 0, so we can reject the null hypothesis, and therefore claim that they are an improvement with respect to the null model.

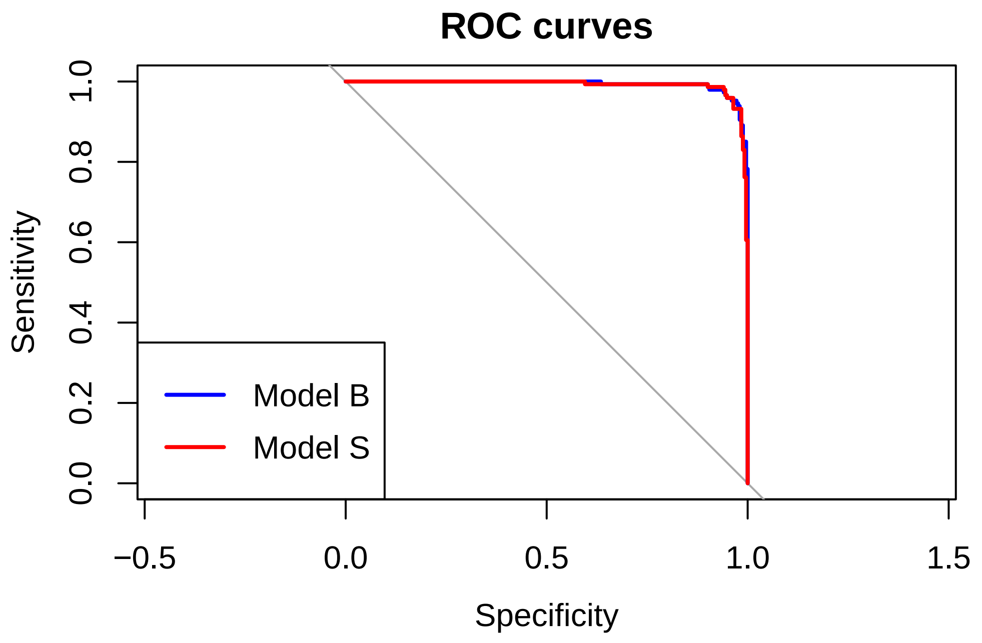
Next, we analyse their performance via the Akaike Information Criterion (AIC), a method that evaluates how well a model fits the data passed to it. Ideally, the AIC score has to be as low as possible, and it is also important to note that it penalises the number of variables used (simpler models are generally preferred). The AIC for model B is approximately 110.35, slightly better than for model S, approximately 120.23. If we had to choose only one model to continue analysing the data, the chosen model would be Model B.

Problem 1.e (2 points)

- Plot the ROC curve of the trained model for both **model B** and **model S**. Display with clear title, label and legend.
- Report AUC values in 3 significant figures for both **model B** and **model S** using `kable()`.

- Discuss which model has a better performance.

```r
# ROC curveS
suppressMessages(invisible({

roc.mb <- roc(train1.df$malignant,
              predict(modelB, type = "response"), plot = TRUE,
              xlim = c(0,1),
              silent = TRUE,
              col = "blue",
              main = "ROC curves")
roc.ms <- roc(train1.df$malignant,
              predict(modelS, type = "response"),
              plot = TRUE,
              silent = TRUE,
              col = "red",
              add = TRUE)

legend("bottomleft",
       legend = c("Model B", "Model S"),
       col = c("blue", "red"), lwd = 2)
}))
```



```r
# Report AUC
auc.df <- data.frame(AUC = c(signif(roc.mb$auc, 3), signif(roc.ms$auc, 3)))
rownames(auc.df) <- c("Model B", "Model S")

kable(auc.df, format = "latex",
      caption = "AUC of models",
      align = "c") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 7: AUC of models

	AUC
Model B	0.993
Model S	0.991

The ROC curves give us an idea of how good Models B and S are when predicting the data that was used for training, for which they almost do not make any mistakes. Their AUC scores are 0.993 and 0.991, extremely high values. As the AUC has to be as large as possible (predicting all diagnosis correctly would lead to an AUC = 1), the conclusion would be that model B is better than model S, although the difference is really small. This agrees with the results in question 1.d, where model B was also preferred.

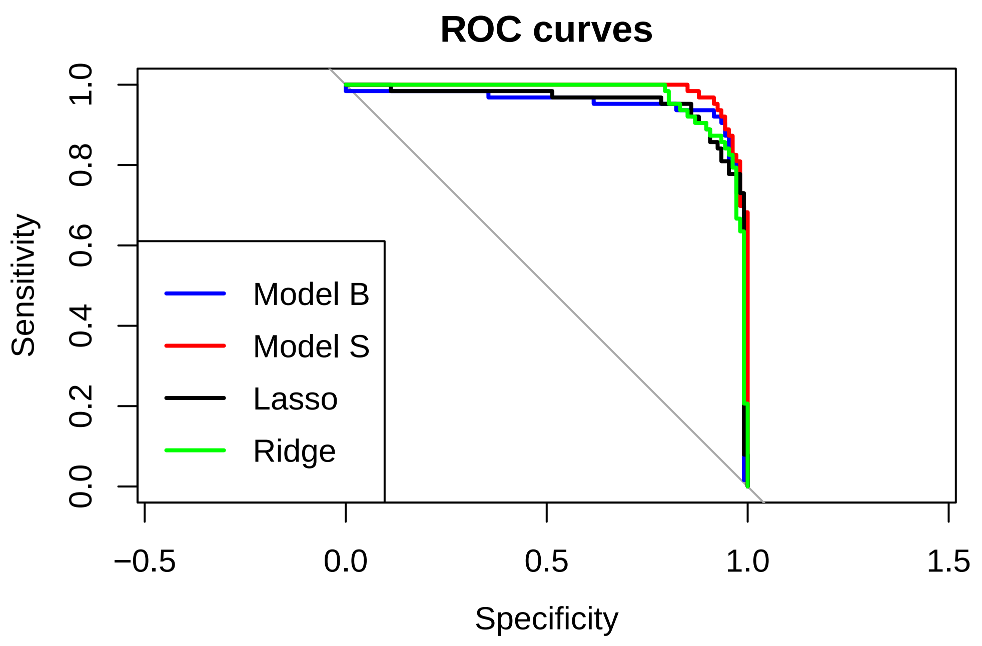
Problem 1.f (6 points)

- Use the four models to predict the outcome for the observations in the test set (use the λ 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.
- Display with clear title, label and legend.
- Compare the training AUCs obtained in **problems 1.b and 1.e** with the test AUCs and discuss the fit of the different models.

```r
# Use models for predictions
pred.lasso <- predict(fit.lasso1, x.test1[, -1], s = "lambda.1se", type = "response")
pred.ridge <- predict(fit.ridge1, x.test1[, -1], s = "lambda.1se", type = "response")
pred.mB <- predict(modelB, as.data.frame(x.test1[, -1]), type = "response")
pred.mB <- predict(modelS, as.data.frame(x.test1[, -1]), type = "response")



```

```r
# Roc curves
suppressMessages(invisible({

roc.mb.test <- roc(test1.df$malignant,
              predict(modelB, as.data.frame(x.test1[, -1]),
                      type = "response"),
              plot = TRUE,
              xlim = c(0,1),
              silent = TRUE,
              col = "blue",
              main = "ROC curves")
roc.ms.test <- roc(test1.df$malignant,
              predict(modelS, as.data.frame(x.test1[, -1]),
                      type = "response"),
              plot = TRUE,
              silent = TRUE,
              col = "red",
              add = TRUE)
lasso.test <- roc(test1.df$malignant,
                  pred.lasso, plot = TRUE,
                  silent = TRUE,
                  col = "black",
                  add = TRUE)
ridge.test <- roc(test1.df$malignant,
                  pred.ridge, plot = TRUE,
                  silent = TRUE,
                  col = "green",
                  add = TRUE)

legend("bottomleft",
       legend = c("Model B", "Model S", "Lasso", "Ridge"),
       col = c("blue", "red", "black", "green"), lwd = 2)
}))
```

```
Warning in roc.default(test1.df$malignant, pred.lasso, plot = TRUE, silent =
TRUE, : Deprecated use a matrix as predictor. Unexpected results may be
produced, please pass a numeric vector.

Warning in roc.default(test1.df$malignant, pred.ridge, plot = TRUE, silent =
TRUE, : Deprecated use a matrix as predictor. Unexpected results may be
produced, please pass a numeric vector.
```



```r
# Report AUC
auc.df <- data.frame(AUC = c(signif(roc.mb.test$auc, 3),
                             signif(roc.ms.test$auc, 3),
                             signif(lasso.test$auc, 3),
                             signif(ridge.test$auc, 3)))
rownames(auc.df) <- c("Model B", "Model S", "Lasso", "Ridge")

# Table
kable(auc.df, format = "latex",
      caption = "AUC of models",
      align = "c") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 8: AUC of models

	AUC
Model B	0.949
Model S	0.985
Lasso	0.952
Ridge	0.968

Above we see the ROC plots for the four models, together with their respective AUC. From the table, we can observe that Model S has the best AUC out of the four models, followed by the Ridge model, Model B and Lasso, respectively. As expected, the four models have reduced their AUC score in the out-of-sample predictions, using the test set. However, this reduction in efficiency is not as large as it could have been expected (recall tha, for the training set, the AUC of Model S was 0.991, only 0.006 more then with the training set). Moreover, it is a bit surprising that the ridge model has even slightly improved its AUC score; some more investigation on this should be carried out to obtain an explanation.

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 in file `GDM.raw.txt` into a data table named `gdm.dt`.
- Impute missing values in `gdm.dt` according to SNP-wise median `allele` count.
- Display first 10 rows and first 7 columns using `kable()`.

```r
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment2")
gdm.dt <- fread("GDM.raw.txt", stringsAsFactors = F)
dim(gdm.dt)
```

```
[1] 789 179
```

```r
# colnames(gdm.dt)
sum(is.na(gdm.dt))
```

```
[1] 649
```

```r
# Median imputation
for (colnm in colnames(gdm.dt)[-c(1,2,3)]) {
    gdm.dt[[colnm]][is.na(gdm.dt[[colnm]])] <- median(gdm.dt[[colnm]], na.rm = T)
}

# Output table
kable(gdm.dt[1:10, 1:7], format = "latex",
      caption = "GDM dataset (first 10 rows, 7 columns)",
      align = "c") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

The GDM dataset contains 649 missing values across the 176 different SNPs. Thus, for each SNP, we input the median value wherever the value is missing. After the missing data has

Table 9: GDM dataset (first 10 rows, 7 columns)

ID	sex	pheno	rs7513574_T	rs1627238_A	rs1171278_C	rs1137100_A
1	FALSE	0	1	0	0	2
2	FALSE	0	0	0	0	1
4	FALSE	1	2	1	1	1
5	FALSE	1	0	1	1	1
6	FALSE	1	0	1	1	1
7	FALSE	0	1	1	1	0
8	FALSE	0	0	0	0	1
12	FALSE	1	1	1	1	1
13	FALSE	1	2	0	0	2
18	FALSE	0	1	0	0	0

been replaced, the dataset looks as in the printed table (only the first 10 rows and 7 columns are shown).

Problem 2.b (8 points)

- Write function `univ.glm.test()` where it takes 3 arguements, `x`, `y` and `order`.
- `x` is a data table of `SNP`s, `y` is a binary outcome vector, and `order` is a boolean which takes `false` as a default value.
- 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.

```r
univ.glm.test <- function(y,x, order = FALSE){

  # Initiate all lists of values to be recorded
  snp.names <- c();  coeff.b0 <- c(); coeff.b1 <- c();  odds <- c()
  se.b0 <- c(); se.b1 <- c();  p.vals.b0 <- c(); p.vals.b1 <- c()

    # Fit regressions and record required values
  for (i in colnames(x)){
    reg <- glm(unlist(y) ~ x[[i]], family = binomial(link="logit"))
    snp.names <- append(snp.names, i)
    coeff.b0 <- append(coeff.b0, reg$coefficients[1])
    coeff.b1 <- append(coeff.b1, reg$coefficients[2])
    se.b0 <- append(se.b0, summary(reg)$coefficients[1, 2])
    se.b1 <- append(se.b1, summary(reg)$coefficients[2, 2])
    p.vals.b0 <- append(p.vals.b0, coef(summary(reg))[1,4])
    p.vals.b1 <- append(p.vals.b1, coef(summary(reg))[2,4])
    odds <- append(odds, exp(coef(reg)[2])) # exponentiate to get the odds ratio
  }

  return.dt <- data.table(snp = snp.names, coeff.b0 = coeff.b0,
                          coeff.b1 = coeff.b1, se.b0 = se.b0, se.b1 = se.b1,
                          p.vals.b0 = p.vals.b0, p.vals.b1 = p.vals.b1,
                          odds = odds)
  # Change order if required
  if (order == TRUE){
     return.dt <- return.dt %>% arrange(abs(return.dt$p.vals.b1))
     }
  return(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`) and name the output data table as `gdm.as.dt`.
- Print the first 10 values of the output from `univ.glm.test()` using `kable()`.
- 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 using `kable()` from the GWAS.
- Report the 95% and 99% confidence intervals on the odds ratio using `kable()`.

```r
# Run regressions
gdm.as.dt <- univ.glm.test(as.list(gdm.dt[, 3]), gdm.dt[, -c(1, 2, 3)])

# Change column names
colnames(gdm.as.dt) <- c("SNP", "B0", "B1", "SE B0", "SE B1",
                         "p-value (B0)", "p-value (B1)",
                         "Odds ratio")

# Table
kable(gdm.as.dt[1:10,], format = "latex",
     caption = "Regressions summary", digits = 3,
     align = "c") |>
 kable_styling(full_width = F, position = "center",
                   latex_options = "hold_position")
```

Table 10: Regressions summary

SNP	B0	B1	SE B0	SE B1	p-value (B0)	p-value (B1)	Odds ratio
rs7513574_T	0.118	0.002	0.105	0.105	0.264	0.984	1.002
rs1627238_A	0.048	0.115	0.100	0.114	0.632	0.314	1.121
rs1171278_C	0.044	0.121	0.100	0.114	0.660	0.286	1.129
rs1137100_A	0.082	0.060	0.099	0.110	0.406	0.586	1.062
rs2568958_A	0.058	0.149	0.087	0.123	0.503	0.226	1.161
rs1514175_A	0.079	0.056	0.104	0.105	0.448	0.593	1.058
rs1555543_C	0.185	-0.078	0.115	0.106	0.107	0.462	0.925
rs10923931_C	0.156	-0.215	0.078	0.183	0.045	0.239	0.807
rs516636_A	0.095	0.055	0.090	0.123	0.289	0.654	1.057
rs574367_G	0.094	0.059	0.089	0.124	0.294	0.635	1.061

The table above shows statistics that summarise the fitted regressions, that took the form

$P = \frac{exp(\hat{\beta}_0+\hat{\beta}_1 x)}{1+exp(\hat{\beta}_0+\hat{\beta}_1 x)}$,

where P represents the inferred probability (of having gestational diabetes), $\hat{\beta}_0$ is the intercept of the regression (in the table represented as B0) and $\hat{\beta}_1$ (B1 in the table) is the coefficient that multiplies the SNP observations. The table also shows information relative to the standard errors and associated p-values of both $\hat{\beta}_0$ and $\hat{\beta}_1$, and the odds ratios. If the SNP values had no effect on gestational diabetes, the odds ratio would be 1. As an example, an odds ratio of 0.8 indicates that the SNP reduces the odds of suffering gestational diabetes by 20%. Similarly, an odds ratio larger than 1 would lead to an increase probability of having gestational diabetes.

```r
# Obtain the SNPs of max risk and pretection
max.idx <- which(gdm.as.dt$`Odds ratio` == max(gdm.as.dt[, `Odds ratio`]))
min.idx <- which(gdm.as.dt$`Odds ratio` == min(gdm.as.dt[, `Odds ratio`]))
max.risk.name <- gdm.as.dt[max.idx, SNP]
min.risk.name <- gdm.as.dt[min.idx, SNP]
maxi <- which(colnames(gdm.dt) == max.risk.name)
mini <- which(colnames(gdm.dt) == min.risk.name)

# Subset and bind the SNPs data
max.risk <- gdm.dt[, ..maxi]
min.risk <- gdm.dt[, ..mini]
extreme.risk <- cbind(max.risk, min.risk)

# Initialise lists to store values
```

```r
b0 <- c();b1 <- c();se.b0 <- c();se.b1 <- c()
z0 <- c();z1 <- c();p0 <- c();p1 <- c()
ci.25 <- c(); ci.975 <- c(); ci.05 <- c(); ci.995 <- c()

# Get required information for each SNP
for (i in colnames(extreme.risk)){
  reg <- glm(unlist(as.list(gdm.dt[, 3])) ~ extreme.risk[[i]],
             family = binomial(link="logit"))
  b0 <- append(b0, coef(summary(reg))[1])
  b1 <- append(b1, coef(summary(reg))[2])
  se.b0 <- append(se.b0, coef(summary(reg))[3])
  se.b1 <- append(se.b1, coef(summary(reg))[4])
  z0 <- append(z0, coef(summary(reg))[5])
  z1 <- append(z1, coef(summary(reg))[6])
  p0 <- append(p0, coef(summary(reg))[7])
  p1 <- append(p1, coef(summary(reg))[8])
  ci.25 <- append(ci.25, exp(confint(reg)[2, 1]))
  ci.975 <- append(ci.975, exp(confint(reg)[2, 2]))
  ci.05 <- append(ci.05, exp(confint(reg, level = 0.99)[2, 1]))
  ci.995 <- append(ci.995, exp(confint(reg, level = 0.99)[2, 2]))
}
```

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

```r
# Define dataframe, change names and output table
extreme.dt <- data.frame(b0 = b0, b1 = b1, se.b0 = se.b0, se.b1 = se.b1,
                         z0 = z0, z1 = z1, p0 = p0, p1 = p1)
colnames(extreme.dt) <- c("B0", "B1", "SE (B0)", "SE (B1)",
                          "z-val (B0)", "z-val (B1)",
                          "p-val (B0)", "p-val (B1)")
rownames(extreme.dt) <- c("Max risk", "Max protective")

kable(extreme.dt, caption = "Summary Statistics" , digits = 3) |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 11: Summary Statistics

	B0	B1	SE (B0)	SE (B1)	z-val (B0)	z-val (B1)	p-val (B0)	p-val (B1)
Max risk	0.082	0.651	0.073	0.317	1.123	2.056	0.262	0.040
Max protective	0.143	-0.602	0.073	0.376	1.960	-1.603	0.050	0.109

The summary statistics for the SNPs that carry higher and lower risk of being diagnosed with gestational diabetes are shown in the table above. The SNP associated with higher risk is rs1423096_T, and its presence leads to more chances of diagnosing diabetes. On the other hand, higher values of the SNP rs11575839_C are associated with lower risk (larger protection) against diabetes. As it can be seen from the table, however, the p-value of the $\hat{\beta}_1$ coefficient of rs11575839_C is not statistically significant to the 5% level, so some further studies should be carried out to conclude its effect on diabetes.

```r
# Define CI dataframes, change names
ci.95df <- data.frame(ci25 = ci.25, ci977 = ci.975)
ci.99df <- data.frame(ci05 = ci.05, ci997 = ci.995)

colnames(ci.95df) <- c("2.5%", "97.5%")
colnames(ci.99df) <- c("0.5%", "99.5%")
rownames(ci.95df) <- c("Maximum risk", "Maximum protection")
rownames(ci.99df) <- c("Maximum risk", "Maximum protection")

# Tables
kable(ci.95df, digits = 3, caption = "Maximum and minimum risk SNPs, 95\\% CI")|>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 12: Maximum and minimum risk SNPs, 95% CI

	2.5%	97.5%
Maximum risk	1.050	3.670
Maximum protection	0.255	1.131

```r
kable(ci.99df, digits = 3, caption = "Maximum and minimum risk SNPs, 99\\% CI")|>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 13: Maximum and minimum risk SNPs, 99% CI

	0.5%	99.5%
Maximum risk	0.874	4.567
Maximum protection	0.198	1.419

Tables 12-13 show the 95% and 99% confidence intervals for the odds ratios obtained from the regressions of the two SNPs that had the largest and lowest odd ratios, or what is the same, the SNPs that yield highest risk and highest protection against gestational diabetes, respectively. First, we focus on rs1423096_T, the SNP that is associated with higher gestational diabetes risk. As expected, the 99% intervals are wider than the 95%, as they are meant to cover a wider range of estimates of $\hat{\beta}_1$, which is the estimate that induces odds ratios. By centering our attention to the 95% confidence interval first, we see that it the odds are expected to be larger than 1 always. From this, we could infer that in more than 95% of the times, the presence of these gene is expected to cause and increase in the chances of gestational diabetes. However, if we observe the 99% confidence interval, we see that there exists a small chance of the odds being lower than one. This results agree with the summary statistics shown before, where $\hat{\beta}_1$ was statistically significant to the 5% level, but not the 1%.

If we know study the odds of rs11575839_C, who is associated with a decrease in probabilities of suffering gestational diabetes, less decisive results are obtained. The 95% confidence interval show that the odds are (likely to be) less than 1, although there exists a possibility of them being equal to or larger than 1. Again, this agrees with the p-values from $\hat{\beta}_1$ before, which was not significant to the 5% level. The uncertainty of the odds not being smaller than 1 is obviously enlarged when we pay attention the the 99% confidence interval. In any case, from both confidence intervals it can be inferred that it is more probable that the odds are lower than 1, than them to be equal to or larger.

Problem 2.d (4 points)

- 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, corresponding `gene` name and `pos`.
- Using `kable()`, report for each `snp.hit` the names of the genes that are within a 1Mb window from the SNP position on the chromosome.
- ***Note: That are genes that fall within +/- 1,000,000 positions using the pos column in the dataset.***

```r
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment2")
gdm.annot <- fread("GDM.annot.txt", stringsAsFactors = F)
dim(gdm.annot)
```

```
[1] 176   4
```

```r
colnames(gdm.annot)
```

```
[1] "chrom" "snp"   "pos"   "gene"
```

```r
sum(is.na(gdm.annot))
```

```
[1] 0
```

```r
# Change name of colummns for easier understanding, merge datasets
colnames(gdm.as.dt)[1] <- "snp.long"
gdm.as.dt$snp = substr(gdm.as.dt$snp.long, 1, nchar(gdm.as.dt$snp.long)-2)
gdm.merge <- join(gdm.as.dt, gdm.annot, by = "snp")

# Select required SNPs and create subdataset
hit.snp <- gdm.merge[gdm.merge$`p-value (B1)` < 10**(-4),
                     c("snp", "snp.long", "chrom", "gene", "pos")]
hit.snp$snp.long <- substr(hit.snp$snp.long, nchar(hit.snp$snp.long),
                           nchar(hit.snp$snp.long))
colnames(hit.snp)[2] <- "allele"

# Table
kable(hit.snp, caption = "SNPs with p-value (less than $10^{-4}$)")|>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 14: SNPs with p-value (less than $10^{-4}$)

snp	allele	chrom	gene	pos
rs12243326	A	10	TCF7L2	114788815
rs2237897	T	11	KCNQ1	2858546

In the previous table, the SNPs that have a highly significant p-value ($< 10^{-4}$) are shown.

```r
# For each gene in hit.snp, calculate 1b distance and
# obtain genes within that distance
for(i in 1:dim(hit.snp)[1]){
  pos <- hit.snp[i, "pos"]
  max.pos <- unlist(pos + 1000000)
  min.pos <- unlist(pos - 1000000)
  close <- gdm.annot[gdm.annot$pos > min.pos & gdm.annot$pos < max.pos, "gene"]
  nam <- paste("close", i, sep = ".")
  assign(nam, close)
}

```

```r
# Change column names and output table
colnames(close.1) <- colnames(close.2) <- "Gene"
kable(unique(close.1), caption = "Genes close to rs12243326")|>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 15: Genes close to rs12243326

Gene
TCF7L2

```r
kable(unique(close.2), caption = "Genes close to rs2237897")|>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 16: Genes close to rs2237897

Gene
TH
KCNQ1
CACNA2D4
SMG6

In the two previous tables we can observe which genes are close to the the SNPs that had a highly significant p-value. The SNP rss12243326 only has one gene close (actually, that gene is precisely the gene associated to that SNP), while rs2237897 has four genes close to it.

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.
- Report odds ratio, 95% confidence interval and p-value using `kable()` for each score.

```r
# SUbset with p-val < 10**-4
risk.weighted4 <- gdm.merge[gdm.merge$`p-value (B1)` < 10**(-4), ]
risk4 <- gdm.dt[, .SD, .SDcols = risk.weighted4$snp.long]

# SUbset with p-val < 10**-3
risk.weighted3 <- gdm.merge[gdm.merge$`p-value (B1)` < 10**(-3), ]
```

```r
risk3 <- gdm.dt[, .SD, .SDcols = risk.weighted3$snp.long]

# SUbset for FTO
risk.weightedfto <- gdm.merge[gdm.merge$gene == "FTO", ]
risk.fto <- gdm.dt[, .SD, .SDcols = risk.weightedfto$snp.long]

# Calculate scores
weighted.score4 <- as.matrix(risk4) %*% risk.weighted4$B1
weighted.score3 <- as.matrix(risk3) %*% risk.weighted3$B1
weighted.scorefto <- as.matrix(risk.fto) %*% risk.weightedfto$B1

# Add as columns to gdm.dt
gdm.dt$wscore4 <- weighted.score4
gdm.dt$wscore3 <- weighted.score3
gdm.dt$wscorefto <- weighted.scorefto

# Fit three different models
mod.w4 <- glm(gdm.dt$pheno ~ weighted.score4, family = binomial(link = "logit"))
mod.w3 <- glm(gdm.dt$pheno ~ weighted.score3, family = binomial(link = "logit"))
mod.wfto <- glm(gdm.dt$pheno ~ weighted.scorefto, family = binomial(link = "logit"))

# Define dataframes for each model, change names
w4.table <- data.frame(odds = exp(coef(mod.w4)[2]),
                       ci25 = exp(confint(mod.w4))[2,1],
                       ci975 = exp(confint(mod.w4))[2,2],
                       pval = coef(summary(mod.w4))[2,4])
```

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

```r
w3.table <- data.frame(odds = exp(coef(mod.w3)[2]),
                       ci25 = exp(confint(mod.w3))[2,1],
                       ci975 = exp(confint(mod.w3))[2,2],
                       pval = coef(summary(mod.w3))[2,4])
```

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

```r
wfto.table <- data.frame(odds = exp(coef(mod.wfto)[2]),
                         ci25 = exp(confint(mod.wfto))[2,1],
                         ci975 = exp(confint(mod.wfto))[2,2],
                         pval = coef(summary(mod.wfto))[2,4])
```

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

```r
colnames(w4.table) <- colnames(w3.table) <-
  colnames(wfto.table) <- c("Odds", "2.5%", "97.5%", "p-value")
rownames(w4.table) <-  rownames(w3.table) <-
  rownames(wfto.table) <- "Fit statistics"

# Tables
kable(w4.table, digits = 4,
      caption = "Summary Statistics with Weighted
              Score (p-val less than $10^{-4}$)") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 17: Summary Statistics with Weighted Score (p-val less than $10^{-4}$)

	Odds	2.5%	97.5%	p-value
Fit statistics	2.7294	1.9244	3.9111	0

```r
kable(w3.table, digits = 4,
      caption = "Summary Statistics with Weighted Score (p-val less than $10^{-3}$)") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 18: Summary Statistics with Weighted Score (p-val less than $10^{-3}$)

	Odds	2.5%	97.5%	p-value
Fit statistics	1.4519	1.2814	1.6511	0

```r
kable(wfto.table, digits = 4,
      caption = "Summary Statistics with Weighted Score (FTO)") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Above are printed the requested summary statistics for the three different models, according to their p-value and gene. For the three models, the odds are larger than 1, from which we infer that the SNPs that had strongly significant results in the previous models have a positive

Table 19: Summary Statistics with Weighted Score (FTO)

	Odds	2.5%	97.5%	p-value
Fit statistics	1.4139	0.8191	2.4526	0.2152

correlation to gestational diabetes, as well as the SNPs found in the gene FTO. The "positive correlation to gestational diabetes" is strongly significant for the SNPs with p-values $< 10^{-4}$ and $< 10^{-3}$. This is supported by the confidence intervals, for which, in both cases, the possible odds are never lower than 1.

Conversely, the positive association between FTO and gestational diabetes is not statistically significant to the 5% level (p-value = 0.2152), so it could be the case that this is not true, and FTO is negatively associated with diabetes, or not associated at all. This is shown by the confidence interval as well, where the 2.5% limit for odds is established at $0.8191 < 1$.

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
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment2")
gdm.test <- fread("GDM.test.txt", stringsAsFactors = F)
dim(gdm.test)
```

```
[1]  40 179
```

```r
# colnames(gdm.test)
sum(is.na(gdm.test))
```

```
[1] 0
```

```r
# Obtain subsets of observations
prisk.weighted4 <- gdm.merge[gdm.merge$`p-value (B1)` < 10**(-4), ]
prisk4 <- gdm.test[, .SD, .SDcols = risk.weighted4$snp]

prisk.weighted3 <- gdm.merge[gdm.merge$`p-value (B1)` < 10**(-3), ]
prisk3 <- gdm.test[, .SD, .SDcols = risk.weighted3$snp]

prisk.weightedfto <- gdm.merge[gdm.merge$gene == "FTO", ]
prisk.fto <- gdm.test[, .SD, .SDcols = risk.weightedfto$snp]


# Calculate scores
pweighted.score4 <- as.matrix(prisk4) %*% prisk.weighted4$B1
pweighted.score3 <- as.matrix(prisk3) %*% prisk.weighted3$B1
pweighted.scorefto <- as.matrix(prisk.fto) %*% prisk.weightedfto$B1

# Add to gdm.test
gdm.test$wscore4 <- pweighted.score4
gdm.test$wscore3 <- pweighted.score3
gdm.test$wscorefto <- pweighted.scorefto
```

After analysing the new dataset for missing data, we perform the required calculations (cell above).

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 and present them using `kable()`

```r
# predictions
pred.m4 <- predict(mod.w4, newdata=list(gdm.test$wscore4),
                   type = "response")
pred.m3 <- predict(mod.w3, newdata=list(gdm.test$wscore3),
                   type = "response")
pred.fto <- predict(mod.wfto, newdata=list(gdm.test$wscorefto),
                    type = "response")

# Calculate log-likelihood
loglik4 <- sum(log(ifelse(gdm.test$pheno == 1, pred.m4, 1 - pred.m4)))
```

```r
loglik3 <- sum(log(ifelse(gdm.test$pheno == 1, pred.m3, 1 - pred.m3)))
loglikfto <- sum(log(ifelse(gdm.test$pheno == 1, pred.fto, 1 - pred.fto)))

ll.df <- data.frame(ll = c(loglik4, loglik3, loglikfto))
colnames(ll.df) <- "Log-likelihood"
rownames(ll.df) <- c("Model 1 (p-value < 10^-4)",
                     "Model 2 (p-value < 10^-3)",
                     "Model 3 (FTO)")

# Table
kable(ll.df, digits = 3,
      caption = "Log-likelihood of models") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 20: Log-likelihood of models

	Log-likelihood
Model 1 (p-value < 10^-4)	-25.068
Model 2 (p-value < 10^-3)	-24.777
Model 3 (FTO)	-28.054

Table 20 shows the log-likelihoods of the different models. Ideally, the more negative, the better. We see that all models perform similarly with their subset of data. Nevertheless, the best performing is Model 3, that only takes FTO genes.

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***)
- 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 using `kable()`.

```r
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment2")
gdm.study <- fread("GDM.study2.txt", stringsAsFactors = F)
dim(gdm.study)
```

```
[1] 176   5
```

```r
colnames(gdm.study)
```

```
[1] "snp"            "effect.allele" "other.allele"  "beta"
[5] "se"
```

```r
sum(is.na(gdm.study))
```

```
[1] 0
```

```r
# HAve acolumn with the allele of the SNP
gdm.as.dt$effect.allele <- substr(gdm.as.dt$snp.long,
                                  nchar(gdm.as.dt$snp.long),
                                  nchar(gdm.as.dt$snp.long))

# Order dataset by SNP
gdm.as.dt <- gdm.as.dt[1:176,]
gdm.as.dt <- gdm.as.dt[order(snp)]
gdm.study <- gdm.study[order(snp)]


# Check which alleles are flipped and which are not
# Invert the betas for those that are flipped
both.ok <- which(gdm.as.dt$effect.allele == gdm.study$effect.allele)
gdm.study[-both.ok, "beta"] <- -gdm.study[-both.ok, "beta"]

# Perform fixed effect meta-analysis by using inverse variance weighting
weight.1 <- 1 / gdm.as.dt$`SE B1`^2
weight.2 <- 1 / gdm.study$se^2
```

```r
# Tables
kable(t(head(weight.1)),
      caption = "Weight of 1st GWAS") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 21: Weight of 1st GWAS

71.04515	95.95663	48.35218	51.12222	80.72417	32.39329

```r
kable(t(head(weight.2)),
      caption = "Weight of 2nd GWAS") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

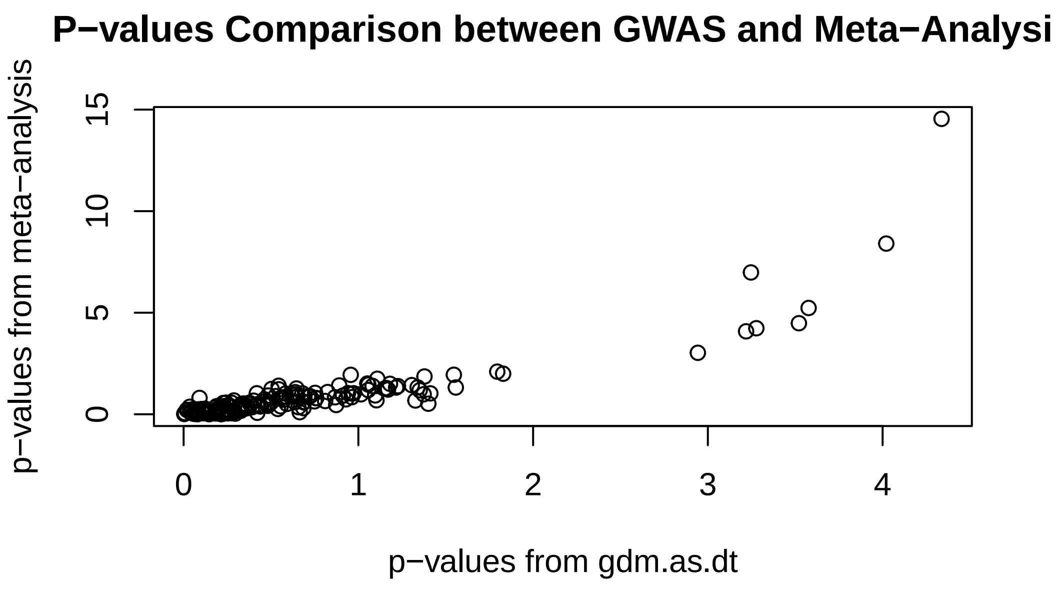
Table 22: Weight of 2nd GWAS

6.112876	9.14147	6.919489	7.069651	10.43071	3.328963

By looking at the weights, it looks like the first study is more powered. Next, we compute the meta analysis, obtaining betas and standard errors according to the weights that have just been calculated.

```r
# Compute meta analysis coeffs and SE
beta.ma <- (weight.1 * gdm.as.dt$B1 + weight.2 * gdm.study$beta) /
  (weight.1 + weight.2)
se.ma <- sqrt(1 / (weight.1 + weight.2))
```

```r
# Calculate p-values and plot
pval.ma <- 2 * pnorm(abs(beta.ma / se.ma), lower.tail = FALSE)
plot(-log10(gdm.as.dt$`p-value (B1)`), -log10(pval.ma),
     # xlim = c(0, 8), ylim = c(0, 16),
     xlab = "p-values from gdm.as.dt",
     ylab = "p-values from meta-analysis",
     main = "P-values Comparison between GWAS and Meta-Analysis"
     )
```



The improvement from the meta-analysis in comparison to individual analysis can be observed by comparing the p-values of the original dataset (gdm.as.dt) against those of the meta-analysis. As it can be seen, the p-values are generally smaller for the meta-analysis, and thus more significant. The meta-analysis yields better results overall.

```r
# Calculate p-values lower tan specified
low.p <- which(pval.ma < 10**(-4))

# Define meta-analysis dataframe, subset as need
# Set the ordering as of increasing pvalues
meta.an <- data.frame(SNP = gdm.as.dt$snp,
                      beta = beta.ma,
                      SE = se.ma,
                      pvalue = pval.ma)
meta.an <- meta.an[low.p, ] %>% arrange(abs(pvalue))

# Table
kable(meta.an, digits = 3,
      caption = "Summary of meta-analysis") |>
 kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 23: Summary of meta-analysis

SNP	beta	SE	pvalue
rs12243326	0.892	0.113	0
rs2237897	-0.590	0.100	0
rs3786897	0.607	0.114	0
rs2237892	-0.483	0.107	0
rs4506565	0.540	0.130	0
rs7903146	0.535	0.133	0
rs7901695	0.541	0.137	0

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.

```r
# Load data
setwd("D:\\Pablo\\Documents\\Universidad\\MASTER\\BiomedicalDS\\data_assignment2")
nki <- fread("nki.csv")
# head(nki)
# summary(nki)
colnames(nki)
```

```
 [1] "Event"             "Diam"              "LymphNodes"        "EstrogenReceptor"
 [5] "Grade"             "Age"               "TSPYL5"            "Contig63649_RC"
 [9] "DIAPH3"            "NUSAP1"            "AA555029_RC"       "ALDH4A1"
[13] "QSCN6L1"           "FGF18"             "DIAPH3.1"          "Contig32125_RC"
[17] "BBC3"              "DIAPH3.2"          "RP5.860F19.3"      "C16orf61"
[21] "SCUBE2"            "EXT1"              "FLT1"              "GNAZ"
[25] "OXCT1"             "MMP9"              "RUNDC1"            "Contig35251_RC"
[29] "ECT2"              "GMPS"              "KNTC2"             "WISP1"
[33] "CDC42BPA"          "SERF1A"            "AYTL2"             "GSTM3"
[37] "GPR180"            "RAB6B"             "ZNF533"            "RTN4RL1"
[41] "UCHL5"             "PECI"              "MTDH"              "Contig40831_RC"
[45] "TGFB3"             "MELK"              "COL4A2"            "DTL"
[49] "STK32B"            "DCK"               "FBXO31"            "GPR126"
[53] "SLC2A3"            "PECI.1"            "ORC6L"             "RFC4"
[57] "CDCA7"             "LOC643008"         "MS4A7"             "MCM6"
[61] "AP2B1"             "C9orf30"           "IGFBP5"            "HRASLS"
```

```
[65] "PITRM1"          "IGFBP5.1"        "NMU"             "PALM2.AKAP2"
[69] "LGP2"            "PRC1"            "Contig20217_RC"  "CENPA"
[73] "EGLN1"           "NM_004702"       "ESM1"            "C20orf46"
```

```r
dim(nki)
```

```
[1] 144  76
```

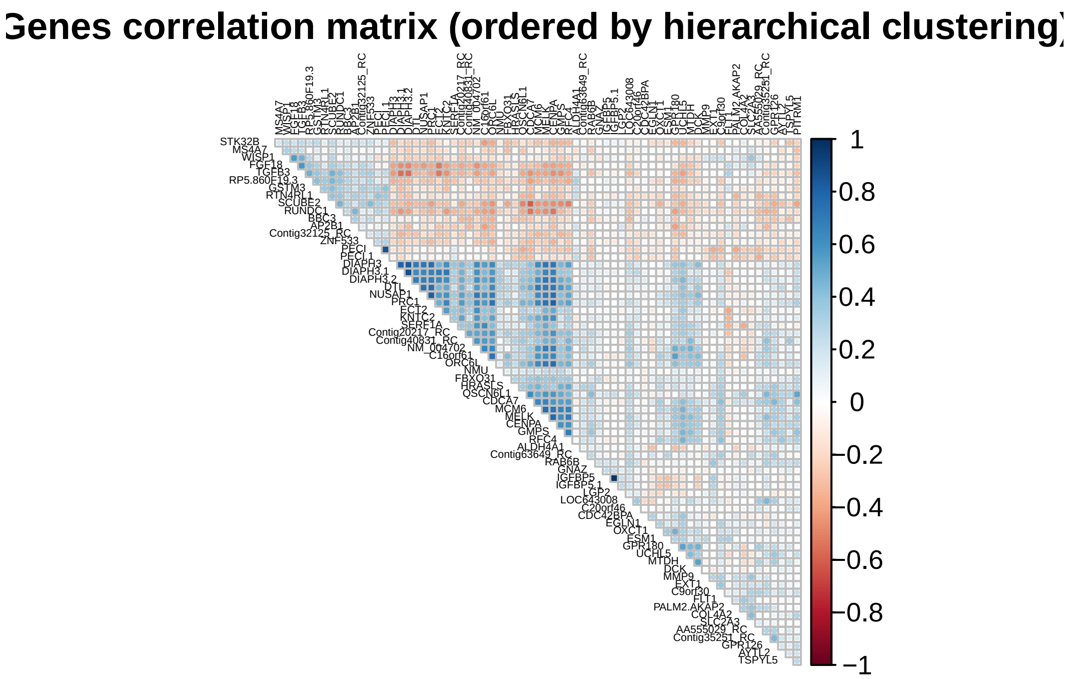
Problem 3.a (6 points)

- Compute the correlation matrix between the gene expression variables, and display it so that a block structure is highlighted using the `corrplot` package.
- Discuss what you observe.
- Identify the unique pairs of (distinct) variables that have correlation coefficient greater than 0.80 in absolute value and report their correlation coefficients.

```r
# Subset genes data
genes <- nki[, -c(1:6)]
cor.genes <- cor(genes, use="pairwise.complete")
dim(cor.genes)
```

```
[1] 70 70
```

```r
# order the variablesby correlation clusters
corrplot(cor.genes, order="hclust",
         # remove the diagonal elements
         diag=FALSE,
         # change the colour and size of the labels
         tl.col="black", tl.cex = 0.30,
         title="Genes correlation matrix (ordered by hierarchical clustering)",
         # display the upper triangle only
         type = 'upper',
         # change the size of the margins (bottom, left, top, right)
         mar=c(0,0,1.2,0))
```



Out of the 76 variables in the dataset, 70 correspond to gene expressions. Thus, we carried out a correlation analysis between these variables to see which genes could be tightly linked to each other. As seen from the correlation plot, the majority of genes are unrelated to each other, or with a extremely weak dependence. However, there are a few that do exhibit some correlation, presented in the following output:

```r
# Subset corelation matrix, set diagonal
# and lower triangle values to NA
tmp <- cor.genes
tmp[lower.tri(tmp)] <- NA
diag(tmp) <- NA
all.cors <- na.omit(melt(tmp))

# Subset high correlations, output them
high.cor <- all.cors[abs(all.cors$value) > 0.8, ]
colnames(high.cor)[3] <- "Correlation"
high.cor
```

```
        Var1     Var2 Correlation
563   DIAPH3 DIAPH3.1   0.8031368
773   DIAPH3 DIAPH3.2   0.8338591
779 DIAPH3.1 DIAPH3.2   0.8868741
```

```
3326     PECI   PECI.1   0.8697836
4187   IGFBP5 IGFBP5.1   0.9775030
4414   NUSAP1     PRC1   0.8298356
4614     PRC1    CENPA   0.8175424
```

```r
# Check high, negative correlations
all.cors[min(all.cors$value) == all.cors$value, ]
```

```
       Var1  Var2      value
3515 SCUBE2 CDCA7 -0.6062384
```

The following are the highest correlated pairs of variables, using a cut-off of 0.8 (in absolute value):

- DIAPH3 and DIAPH3.1 - Thesevariables have a correlation of 0.803.
- DIAPH3 and DIAPH3.2 - These variables have a correlation of 0.833.
- DIAPH3.1 and DIAPH3.2 - These variables have a correlation of 0.887.
- PECI and PECI.1 - These variables have a correlation of 0.87.
- IGFBP5 and IGFBP5.1 - These variables have a correlation of 0.978.
- NUSAP1 and PRC1 - These variables have a correlation of 0.83.
- PRC1 and CENPA - These variables have a correlation of 0.818.

These set of seven pairs of variables are strongly correlated. Some more medical insight would be needed, however it looks like some of these genes are functionally related to each other, as they have very similar names (for instance, DIAPH3 and DIAPH3.1). Aside from this, we see that DIAPH3, DIAPH3.1, DIAPH3.2 and PRC1 are the only genes strongly correlated to two other genes.

Furthermore, there seems to be no genes as strongly negatively correlated, being SCUBE2 and CDCA7 (correlation = -0.606) the genes that show the most negative correlation between each other.

Problem 3.b (8 points)

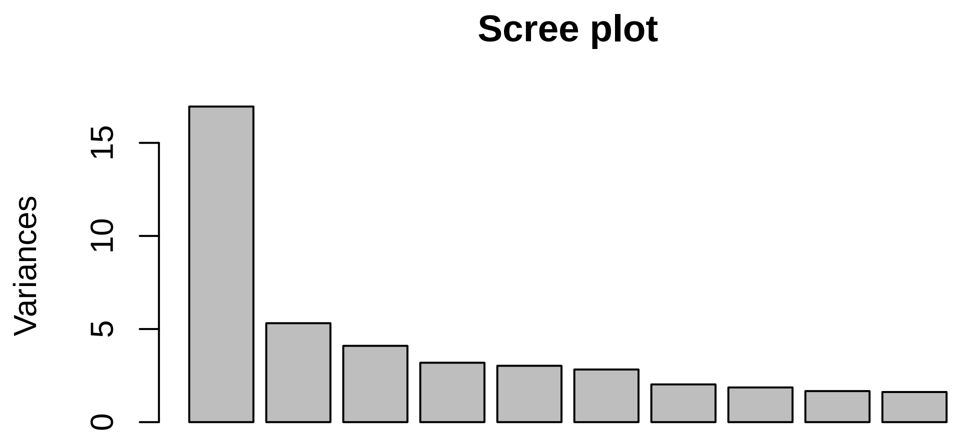
- Perform PCA analysis (only over the columns containing gene expressions) in order to derive a patient-wise summary of all gene expressions (dimensionality reduction).
- Decide which components to keep and justify your decision.
- Test if those principal components are associated with the outcome in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

```r
# Perform PCA
pca.genes <- prcomp(genes, center = T, scale = T)
summary(pca.genes)
```

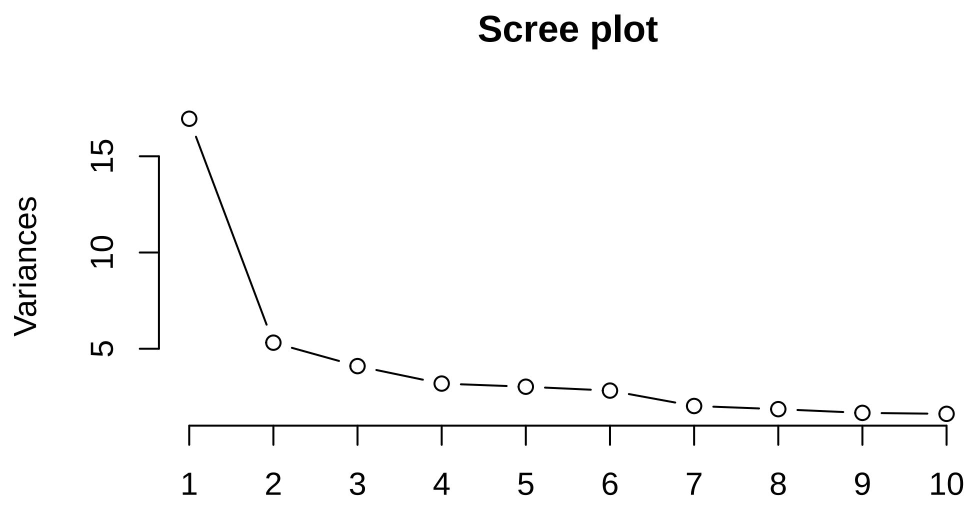
```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     4.1171 2.30541 2.02437 1.78597 1.73982 1.68091 1.42309
Proportion of Variance 0.2422 0.07593 0.05854 0.04557 0.04324 0.04036 0.02893
Cumulative Proportion  0.2422 0.31808 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
```

```r
# Scree plots
screeplot(pca.genes, main = "Scree plot")
```



```r
plot(pca.genes, type = "line", main = "Scree plot")
```



In the cell above, PCA is carried out for the 70 gene expression variables that are available. By observing the scree plots and the printed summary, we have decided to use the first 22 Principal Components (PCs) for further analysis. These PCs explain 80% of the variance found in the dataset, and all of them are able to explain at least 1 standard deviation of the data. This way, we cut down the number of variables that need to be used, while barely losing any information.

By looking at the scree plots, it is observed that the curve flattens around the 10^{th} and 11^{th} component, however these only explain about 60% of the variance in the data, and thus it was preferred to take a higher number of PCs for the coming testing.

```r
# Calculate PCA predictions for training data
scores <- predict(pca.genes)[, 1:22]

# Perform regressions
regr.unadjusted <- glm(nki$Event ~ scores, family = binomial(link = "logit"))
summary(regr.unadjusted)
```

```
Call:
glm(formula = nki$Event ~ scores, family = binomial(link = "logit"))
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0537  -0.6799  -0.2771   0.5553   2.1124

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.20028    0.27376  -4.384 1.16e-05 ***
scoresPC1    0.20690    0.06801   3.042 0.002349 **
scoresPC2   -0.09595    0.10717  -0.895 0.370661
scoresPC3    0.35224    0.12941   2.722 0.006490 **
scoresPC4   -0.30003    0.14715  -2.039 0.041460 *
scoresPC5   -0.03990    0.13377  -0.298 0.765523
scoresPC6    0.20574    0.13697   1.502 0.133082
scoresPC7   -0.15811    0.17863  -0.885 0.376070
scoresPC8    0.28430    0.18132   1.568 0.116889
scoresPC9   -0.03697    0.17497  -0.211 0.832671
scoresPC10  -0.42177    0.21071  -2.002 0.045320 *
scoresPC11  -0.51998    0.20843  -2.495 0.012606 *
scoresPC12  -0.07031    0.21732  -0.324 0.746283
scoresPC13  -0.08521    0.20302  -0.420 0.674699
scoresPC14  -0.19484    0.21607  -0.902 0.367198
scoresPC15   0.03934    0.25485   0.154 0.877315
scoresPC16  -0.14935    0.21444  -0.696 0.486140
scoresPC17  -0.50057    0.23279  -2.150 0.031533 *
scoresPC18  -0.33433    0.25870  -1.292 0.196241
scoresPC19   1.25848    0.33208   3.790 0.000151 ***
scoresPC20  -0.01934    0.24767  -0.078 0.937766
scoresPC21  -0.13421    0.24977  -0.537 0.591036
scoresPC22   0.22182    0.25593   0.867 0.386092
---
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: 119.97  on 121  degrees of freedom
AIC: 165.97

Number of Fisher Scoring iterations: 6
```

```r
regr.adjusted <- glm(nki$Event ~ nki$Age + nki$EstrogenReceptor +
                       nki$Grade + scores, family = binomial(link = "logit"))
summary(regr.adjusted)
```

```

Call:
glm(formula = nki$Event ~ nki$Age + nki$EstrogenReceptor + nki$Grade +
    scores, family = binomial(link = "logit"))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0887  -0.6440  -0.2146   0.4856   2.0638

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                   3.552371   2.480550   1.432 0.152118
nki$Age                      -0.070725   0.051415  -1.376 0.168954
nki$EstrogenReceptorPositive -2.014190   1.284564  -1.568 0.116882
nki$GradePoorly diff          0.581981   0.665887   0.874 0.382122
nki$GradeWell diff           -0.902411   0.710674  -1.270 0.204157
scoresPC1                     0.018918   0.112378   0.168 0.866313
scoresPC2                     0.095743   0.155004   0.618 0.536785
scoresPC3                     0.361874   0.135972   2.661 0.007782 **
scoresPC4                    -0.333277   0.160599  -2.075 0.037967 *
scoresPC5                    -0.005460   0.146296  -0.037 0.970231
scoresPC6                     0.362998   0.168669   2.152 0.031387 *
scoresPC7                    -0.225649   0.189672  -1.190 0.234172
scoresPC8                     0.427511   0.209166   2.044 0.040965 *
scoresPC9                     0.013745   0.184380   0.075 0.940576
scoresPC10                   -0.493765   0.219926  -2.245 0.024759 *
scoresPC11                   -0.515269   0.221733  -2.324 0.020135 *
scoresPC12                   -0.117784   0.234762  -0.502 0.615867
scoresPC13                    0.004174   0.225360   0.019 0.985222
scoresPC14                   -0.177659   0.247854  -0.717 0.473506
scoresPC15                   -0.073477   0.268274  -0.274 0.784170
scoresPC16                   -0.138034   0.229900  -0.600 0.548234
scoresPC17                   -0.306240   0.245819  -1.246 0.212840
scoresPC18                   -0.352518   0.261765  -1.347 0.178077
scoresPC19                    1.460778   0.382649   3.818 0.000135 ***
scoresPC20                    0.034199   0.271095   0.126 0.899613
scoresPC21                   -0.281986   0.285198  -0.989 0.322791
scoresPC22                    0.254943   0.280480   0.909 0.363374
```

```
---
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: 112.81  on 117  degrees of freedom
AIC: 166.81

Number of Fisher Scoring iterations: 6
```

Two regressions have been carried out with the PCs selected previosuly. The first of these regressions (Unadjusted) used the 22 PCs to preedict `Event`, a binary variable that represents whether there were complications after surgery. This regression has eight statistically significant parameters to the 5% level, one of them being the intercept. The other significant coefficients are those for PC1, PC3, PC4, PC10, PC11, PC17 and PC19.

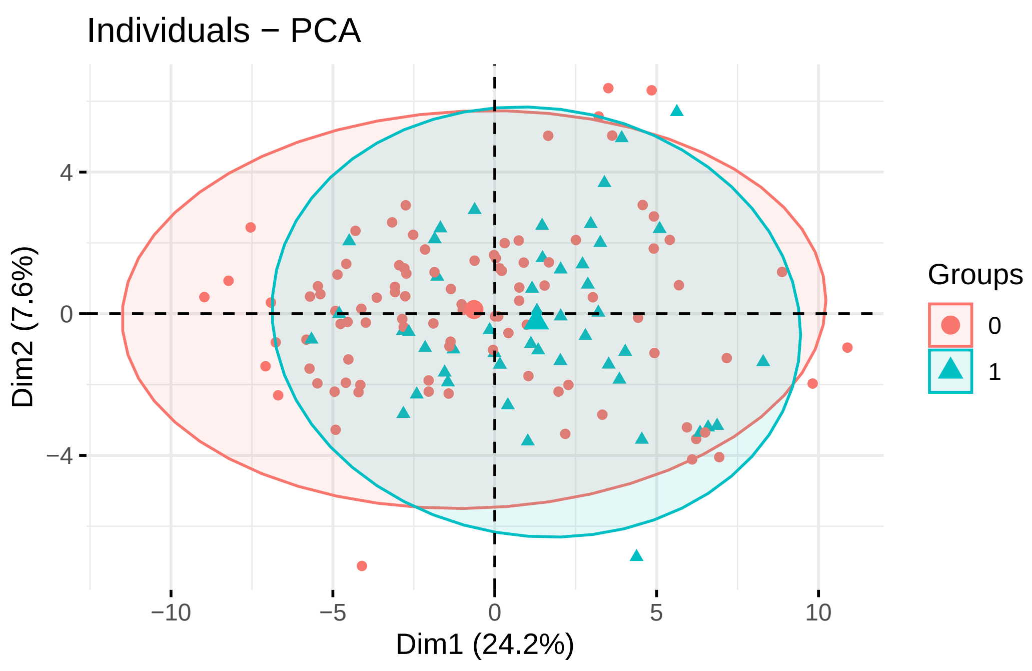
The second regression (Adjusted) includes variables like age, estrogen receptors and grade (made up of three factors: "Well diff", "intermediate" and "Poorly diff") together with the 22 PCs to forecast `Event`. This time, only seven coefficients are statistically significant to the 5% level: PC3, PC4, PC6, PC8, PC10, PC11 and PC19. None of the newlyt added variables nor the intercept are statistically significant.

Comparing both regressions, we observe that PC3, PC4, PC10, PC11 and PC19 are significant in both regressions. Moreover, their effect on the estimation of `Event` is of similar magnitude in both regressions. The AIC score obtained by the Unadjusted regression is of 165.97, slightly better than for the Adjusted, 166.81. Considering that their predictive performance looks similar, the difference in AIC between the models may be due to the penalisation for number of variables that is taken when computing AIC, which is larger for the Adjusted model. Therefore, if we had to take a single model, we would stick to the Unadjusted model.

Problem 3.c (8 points)

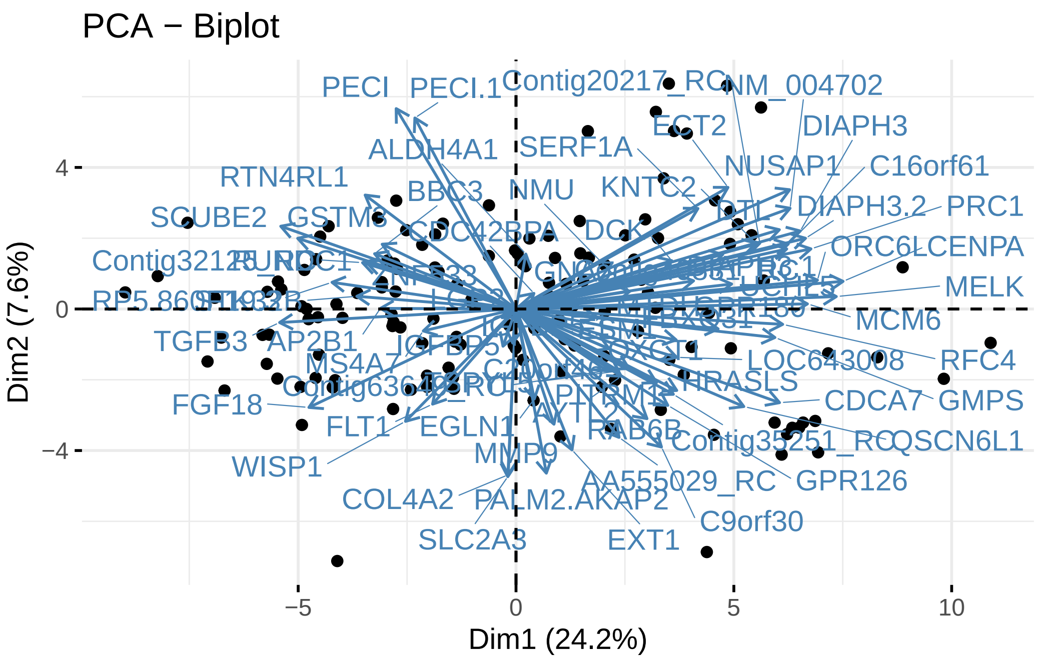
- Use PCA plots to compare the main driverswith the correlation structure observed in **problem 3.a**.
- Examine how well the dataset may explain your outcome.
- Discuss your findings in full details and suggest any further steps if needed.

```r
# Plot
fviz_pca_ind(pca.genes, geom = "point", habillage = nki$Event, addEllipses = T)
```



The plot above shows each observation in the dataset in terms of their score for PC1 and PC2. They are coloured depending on their `Event` value, being red if `Event = 0` and blue if `Event = 1` . It can be seen in the plot that PC1 (x-axis) explains 24.2% of the total variance, while PC2 (y-axis) only accounts for 7.6% of the variance in the dataset. The ellipses contain 95% of the data in each `Event` group, and are defined around a centroid that is represented a as slightly larger datapoint. Clearly, the ellipses overlap, meaning that these two PCs on their own are not a good way to classify the data. This is unsurprising, as PC1 and PC2 only account for a maximum of 31.8% of the total variance found in the data. In any case, if we look at the centroids of the two groups, we see that `Event = 0` lies in the negative side of PC1, while `Event = 1` remains in the positive half of PC1, with barely no distinction in their y-axis (PC2) value. This means that the model predicts a higher chance of having post-surgery complications if, given your genes, the score resulting from PC1 is positive.

```r
# Plot
fviz_pca_biplot(pca.genes, geom = "point", repel = T)
```



Now, we analyse the biplot given by the first two PCs in the PCA. We see no evident outliers in terms of scores of PC1 and PC2. The variables that influence each PC the most are the ones with a longer arrow along its respective axis. For instance, it can be argued that CENPA and MELK have a strong, positive effect in the PC1 score, while TGFB3 will also be strong but negative.

Regarding the variables that we previously highlighted to be strongly correlated, we see that their effect in each component is similar, as was expected. DIAPH3, DIAPH3.1 and DIAPH3.2 can be found in the right hand side of the plot, with strong positive effects on PC1, and a weaker and positive effect on PC2. On the contrary, PECI and PECI.1 have a strong, positive effect on PC2, but a weak and negative effect on PC1. IGFPB5 and IGFPB5.1 are both weakly and negativel related to PC1 and PC2. PRC1 and CENPA have similar effects as well, strong and positive on PC1 and weak and positive in PC2. Lastly, NUSAP1 has a similar effect to PRC1 on PC1, with a stronger effect on PC2. Thus, we conclude that these strongly correlated variables indeed have comparable effects in the PCs, and it could be argued that, for the sake of simplicity of models, one element of each pair of correlated variables could be eliminated, without damaging the predictive power of the models.

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 with several experiments and evidences.

The chosen model is Ridge. The advantages of Ridge is that it takes all variables available, and the magnitude of the coefficients of each variable will depend on how relevant they are in the estimation of complications after surgery. Moreover, to improve the accuracy, we will perform cross-validation to ensure a better generalisation of the model to new data. For the Ridgemodel, we will use all variables (including age, grade, etc.) because they might be able to add information in determining future complications, and otherwise their coefficient in the regression will be reduced until they are neglectible in the forecast. We do not need to take into account missing values because there are none in the dataset.

We will split the dataset into training and testing set, with the training set being a random subset of 75% of the original dataset. The model will be finetuned with the training set, and the test set will be used to evaluate its predictive power.

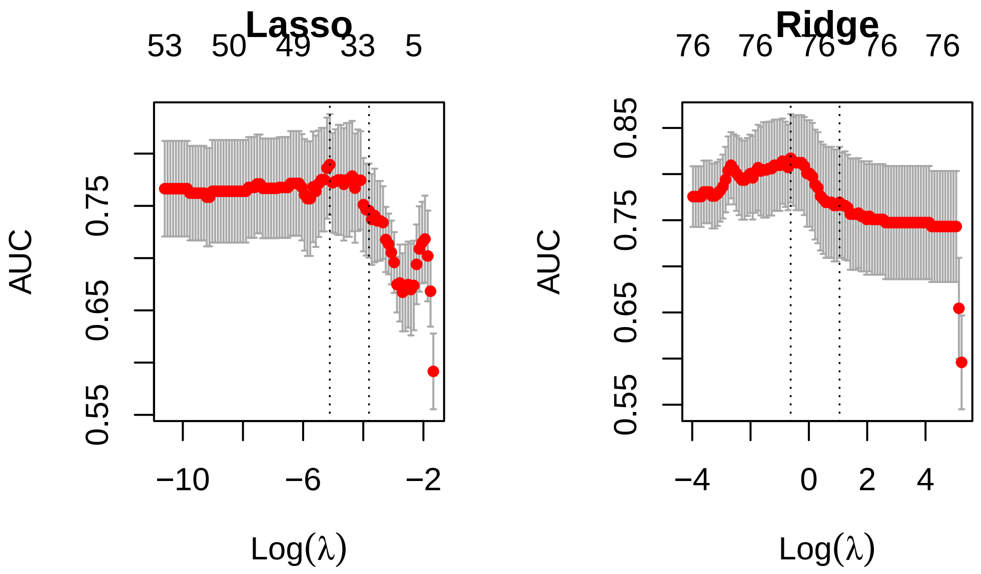
Note that, in the coming code cells, we will also fit a Lasso model to see the differences in performance, and justify why the Ridge model is chosen.

```r
# For reproducibility
set.seed(1)

# Obtain the rows that will take part of the training set
train.idx4 <- createDataPartition(nki$Event, p = 0.75)$Resample1

nki.prep <- prepare.glmnet(nki, formula=~ .)

# Define train and test sets, for x and y
nki.train <- nki.prep[train.idx4, ]
nki.test <- nki.prep[train.idx4, ]

nki.train.x <- nki.prep[train.idx4, -1]
nki.train.y <- nki.prep[train.idx4, 1]
nki.test.x <- nki.prep[-train.idx4, -1]
nki.test.y <- nki.prep[-train.idx4, 1]
```

Now, we fit the Ridge and Lasso models with the training dataset. We use `cv.glmnet()` to implicitly perform cross-validation (with 10 folds).

```r
# Fit models
fit.lasso4 <- cv.glmnet(nki.train.x, nki.train.y, family = "binomial",
                        type.measure = "auc")
fit.ridge4 <- cv.glmnet(nki.train.x, nki.train.y, alpha = 0, family = "binomial",
                        type.measure = "auc")
```

```r
# Plots
par(mfrow = c(1,2))
plot(fit.lasso4, main="Lasso")
plot(fit.ridge4, main="Ridge")
```



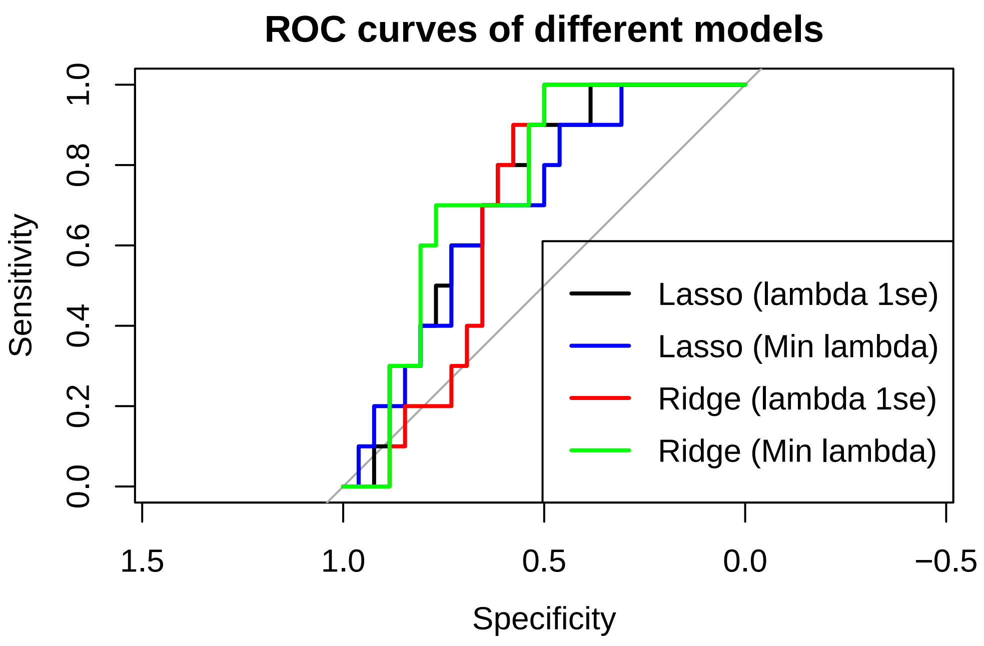
```r
cat("AUC score of Lasso regression using lambda min:",
    fit.lasso4$cvm[fit.lasso4$index][1],
    "\nAUC score of Ridge regression using lambda min:",
    fit.ridge4$cvm[fit.ridge4$index][1],
    "\nAUC score of Lasso regression using lambda 1se:",
    fit.lasso4$cvm[fit.lasso4$index][2],
    "\nAUC score of Ridge regression using lambda 1se:",
    fit.ridge4$cvm[fit.ridge4$index][2])
```

```
AUC score of Lasso regression using lambda min: 0.7895025
AUC score of Ridge regression using lambda min: 0.8171149
AUC score of Lasso regression using lambda 1se: 0.7454549
AUC score of Ridge regression using lambda 1se: 0.7693746
```

From the output above, we see that the Ridge models (using either λ_{min} or λ_{1se}) obtain higher

AUC scores than the Lasso models. Now, we proceed to analyse the forecasting power of each model, using the test model.

```r
# Predictions for Lasso and Ridge models, using lambda min and 1se
pred.lasso4 <- predict(fit.lasso4, nki.test.x, s = "lambda.1se", type = "response")
pred.ridge4 <- predict(fit.ridge4, nki.test.x, s = "lambda.1se", type = "response")

pred.lasso4min <- predict(fit.lasso4, nki.test.x, s = "lambda.min",
                          type = "response")
pred.ridge4min <- predict(fit.ridge4, nki.test.x, s = "lambda.min",
                          type = "response")

# ROC curves
suppressMessages(invisible({
lasso4.test1se <- roc(nki.test.y,
                      pred.lasso4, plot = T,
                      silent = TRUE,
                      col = "black",
                      main = "ROC curves of different models")
lasso4.testmin <- roc(nki.test.y,
                      pred.lasso4min, plot = T,
                      silent = TRUE,
                      col = "blue",
                      add = TRUE)

ridge4.test1se <- roc(nki.test.y,
                      pred.ridge4, plot = T,
                      silent = TRUE,
                      col = "red",
                      add = TRUE)
ridge4.testmin <- roc(nki.test.y,
                      pred.ridge4min, plot = T,
                      silent = TRUE,
                      col = "green",
                      add = TRUE)
legend("bottomright",
       legend = c("Lasso (lambda 1se)", "Lasso (Min lambda)",
                  "Ridge (lambda 1se)", "Ridge (Min lambda)"),
       col = c("black", "blue", "red", "green"), lwd = 2)
}))
```



In the plot above we see the ROC curves of each model. We clearly see that the Ridge models perform better, in general, that the Lasso.

```r
cat("Lasso model using lambda 1se:",
    lasso4.test1se$auc,
    "\nLasso model using lambda min:",
    lasso4.testmin$auc,
    "\nRidge model using lambda 1se:",
    ridge4.test1se$auc,
    "\nRidge model using lambda min:",
    ridge4.testmin$auc
)
```

```
Lasso model using lambda 1se: 0.7192308
Lasso model using lambda min: 0.6923077
Ridge model using lambda 1se: 0.6807692
Ridge model using lambda min: 0.7423077
```

Again, if we check the AUC of each model, we observe that the Ridge models outperforms Lasso. The difference between using λ_{min} against λ_{1se} is small, although λ_{1se} obtains a slightly better score. However, patients, as well as doctors, are ultimately interested in the outcome of the predictions ,whether there will be post-operation difficulties or not. Therefore, we move on to analyse the proportion of predictions that are correct, for each model.

```r
# Create vector with element = 1 if the predicted probability
# was larger than 0.5, and 0 otherwise; for each model
pred.lasso4.bin <- ifelse(pred.lasso4 > 0.5, 1, 0)
pred.ridge4.bin <- ifelse(pred.ridge4 > 0.5, 1, 0)
pred.lasso4min.bin <- ifelse(pred.lasso4min > 0.5, 1, 0)
pred.ridge4min.bin <- ifelse(pred.ridge4min > 0.5, 1, 0)

# Calculate proportions
cat("Proportion of correct predictions of Lasso model, using lambda 1se:",
    round(sum(pred.lasso4.bin == nki.test.y)/length(nki.test.y), 3),
    "\nRidge model, using lambda 1se:",
    round(sum(pred.ridge4.bin == nki.test.y)/length(nki.test.y),3),
    "\nLasso model, using lambda min:",
    round(sum(pred.lasso4min.bin == nki.test.y)/length(nki.test.y), 3),
    "\nRidge model, using lambda min:",
    round(sum(pred.ridge4min.bin == nki.test.y)/length(nki.test.y), 3)
)
```

```
Proportion of correct predictions of Lasso model, using lambda 1se: 0.694
Ridge model, using lambda 1se: 0.667
Lasso model, using lambda min: 0.639
Ridge model, using lambda min: 0.694
```

Once again, Ridge models outperfom Lasso. In this case, however, λ_{min} performs better than λ_{1se}.

With all these information taken into account, we have decided to use Ridge as our final model. Furthermore, the chosen submodel that would be given to doctors/used in the professional sector would be the Ridge model using λ_{min}. As we have seen through the last few code outputs, all the metrics are similar whether λ_{min} or λ_{1se} is used, but ultimately what matters in practice is the number of correct predictions, and in this aspect the model with λ_{min} performed a bit better.

Using the chosen model, we could expect about 85% - 90% of correct forecasts on post-surgery complications. The value of $\lambda_{min} = 0.02288$.