Assignment 2

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

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)

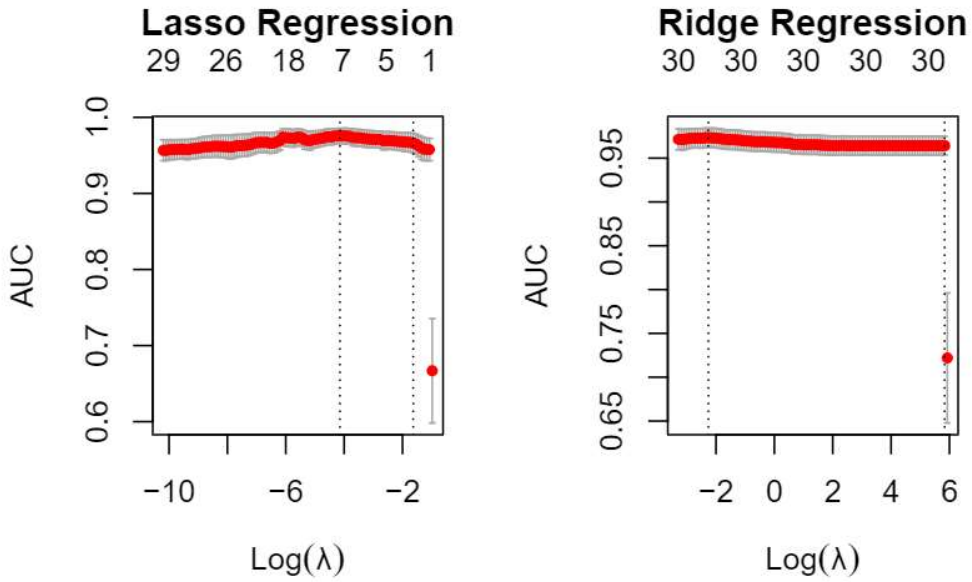
- Using package `caret`, create a data partition so that the training set contains 70% of the observations (set the random seed to 984065 beforehand).
- Fit both a ridge and Lasso regression model which use cross validation on the training set to diagnose the type of tumour from the 30 biomarkers.
- 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
# s2319494
# Set given seed value using the set.seed() function.
set.seed(984065)
# Read the dataset wdbc2.csv using fread().
wdbc.dt<-fread('wdbc2.csv')
# Convert given dataset to Data table
wdbc.dt<-setDT(wdbc.dt)
# Now it is time to do a train-test split of 70:30. This can be
# done using the createDataPartition() function specifying p=0.7.
# The data is split once and thus times=1 and we don't have to
# view the result as a list and hence list=FALSE.
t_seti<-createDataPartition(wdbc.dt$diagnosis,
                            p=0.7,
                            list = FALSE,
                            times = 1)
# t_seti returns a matrix of indicies of rows that collectively
# comprise the training set and these rows are assigned to tr_set.
# This training set will be used to train the models.
tr_set<-wdbc.dt[t_seti,]
# The remaining 30% of rows comprise the test set and these can be
# obtained by a simple complement or wdbc.dt[-t_seti,].
ts_set<-wdbc.dt[-t_seti,]
# The next step is to convert all categorical variable diagnosis to
# numeric by assigning 'benign'=1 and 'malignant'=2.
```

```r
tr_set$diagnosis<-c(benign=1,malignant=2)[tr_set$diagnosis]
ts_set$diagnosis<-c(benign=1,malignant=2)[ts_set$diagnosis]
# Convert resultant train data table to matrix using as.matrix()
tr_set<-as.matrix(tr_set)
# Now let us seperate dependant and independent variables.
# The second column is the independent variable and it is assigned
# to ytr_set.
ytr_set<-tr_set[,2]
# Columns 3 to 32 are the dependent variables and are is assigned
# to xtr_set.
xtr_set<-tr_set[,3:32]
# First we fit the Lasso Regression model (alpha=1) using
# cross-validation to the training data using the cv.glmnet()
# function.
fit.cv.lasso<-cv.glmnet(xtr_set,ytr_set,
                        type.measure = 'auc',
                        family='binomial')
# Next we fit the Ridge Regression model (alpha=0) in the
# same way.
fit.cv.ridge<-cv.glmnet(xtr_set,ytr_set,
                        alpha=0,
                        type.measure = 'auc',
                        family='binomial')
# Next, we split the screen into 2 halves using par() with
# mfrow=c(1,2). Margin sizes can also be customized using
# 'mar' argument.
par(mfrow=c(1,2),
    mar=c(4,4,5,2))
# Now we plot the Ridge and Lasso Regression objects.
plot(fit.cv.lasso,
     main='Lasso Regression')
plot(fit.cv.ridge,
     main='Ridge Regression')
```

Table 1: (1a)Lambda that maximises AUC

	Lambda
Lasso Regression	0.0158513
Ridge Regression	0.1042908




```
## Answer in this chunk
```

```
# The lambda values that maximise AUC for both Regression models
# (lambda.min) are tabulated in a matrix.
res<-matrix(c(fit.cv.lasso$lambda.min,
              fit.cv.ridge$lambda.min),
            nrow=2,
            dimnames = list(c('Lasso Regression',
                              'Ridge Regression'),
                            c('Lambda')))
# The matrix is returned using the kable() function.
kable(res,
      caption = '(1a)Lambda that maximises AUC')
```

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`.

```
# Store indicies for lambda.min and lambda.1se for both Ridge
# and Lasso Regression respectively.
i1<-fit.cv.ridge$index[1]
i2<-fit.cv.ridge$index[2]
i3<-fit.cv.lasso$index[1]
i4<-fit.cv.lasso$index[2]
# Create a data table for respective lambda values, coresponding
# AUC and Model size
d<-data.table(
  'Regression Type'=c('Ridge (lambda.min)',
                      'Ridge (lambda.1se)',
                      'Lasso (lambda.min)',
                      'Lasso (lambda.1se)'),
  'Lambda'=c(round(fit.cv.ridge$lambda[i1],3),
             round(fit.cv.ridge$lambda[i2],3),
             round(fit.cv.lasso$lambda[i3],3),
             round(fit.cv.lasso$lambda[i4],3)),
  'AUC'=c(round(fit.cv.ridge$cvm[i1],3),
             round(fit.cv.ridge$cvm[i2],3),
             round(fit.cv.lasso$cvm[i3],3),
             round(fit.cv.lasso$cvm[i4],3)),
  'Model Size'=c(fit.cv.ridge$nzero[i1],
                 fit.cv.ridge$nzero[i2],
                 fit.cv.lasso$nzero[i3],
                 fit.cv.lasso$nzero[i4])
)
# Pirnt data table using kable()
kable(d,caption = '(1b)Penalty Models Parameters')
```

```
# The first notable observation from the results is that the
# AUC in both Regression models are particularly bigger
# at lambda.min. This is actually because that is the value of
# lambda that maximizes AUC. The corresponding AUC that we see
# when lambda=lambda.min is the highest possible AUC you can get.
```

Table 2: (1b)Penalty Models Parameters

Regression Type	Lambda	AUC	Model Size
Ridge (lambda.min)	0.104	0.973	30
Ridge (lambda.1se)	341.505	0.964	30
Lasso (lambda.min)	0.016	0.976	7
Lasso (lambda.1se)	0.195	0.966	2

```
# Based on AUCs alone, Lasso at lambda.min appears to be the
# best model.
# Apart from this, the number of non-zero coefficients in Ridge
# Regression seem to be significantly bigger than Lasso Regression.
# This is because in Lasso Regression, some coefficients are shrunk
# to zero in the training process and thus gradually reducing the
# overall number of non-zero coefficients.
```

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***

```
# First we create a data table with all required columns form the
# training set. The first column is unnecessary so we select all
# columns from the second one [2:32].
Tr_set<-data.table(tr_set[,2:32])
# Now we fit a full model(diagnosis~.) in which outcome is dependent
# on all 30 biomarkers using lm().
lin_mod<-lm(diagnosis~.,
            data = Tr_set)
# Now we model stepwise backward selection. This can be done using
# the stepAIC() function with direction='backward'. We pass the
# full model (lin_mod) to it. The trace of the model explains every
# step in backward selection and for the moment, we will be skipping
```

Table 3: (1c)Model B: Variables Selected

x
radius
texture
perimeter
area
compactness
concavepoints
radius.stderr
concavity.stderr
radius.worst
area.worst
smoothness.worst
symmetry.worst
fractaldimension.worst

```
# that information by setting trace=FALSE.
sel.backward<-stepAIC(lin_mod,
                      direction = 'backward',
                      trace = FALSE
                      )
# Next, we report the variables selected. This can be done by
# passing the colnames() function on the new model. And since
# we are not interested in the outcome variable, we return
# all variables except that ([2:ncol(sel.backward$model)]).
# The result is returned using the kable() function.
kable(colnames(sel.backward$model)[2:ncol(sel.backward$model)],
      caption = '(1c)Model B: Variables Selected')
```

```
# Finally, we report the coefficients of the model (stored in
# sel.backward$coefficients in increasing order of absolute values)
kable(
sel.backward$coefficients[order(abs(sel.backward$coefficients))],
      caption = '(1c)Model B: Coefficients')
```

```
# Now we fit a null model (diagnosis~1) in which outcome is only
# dependent on the intercept of the model using lm().
fwd_model<-lm(diagnosis~1,
              data=Tr_set)
```

Table 4: (1c)Model B: Coefficients

	x
area	-0.0004414
area.worst	-0.0008836
perimeter	-0.0059521
texture	0.0152970
radius	0.0481229
radius.worst	0.1540157
radius.stderr	0.3286845
symmetry.worst	0.7282311
concavity.stderr	1.2530096
(Intercept)	-1.5785245
smoothness.worst	2.7875483
compactness	-3.7735871
fractaldimension.worst	4.8277913
concavepoints	6.6934393

```
forward_select<-stepAIC(fwd_model,
                        scope = list(upper=lin_mod),
                        direction = 'forward',
                        trace=FALSE)
# Next, we report the variables selected. This can be done by passing
# the colnames() function on the new model. And since we are not
# interested in the outcome variable, we return all variables except
# that ([2:ncol(forward_select$model)]).
# The result is returned using the kable() function.
kable(
colnames(forward_select$model)[2:ncol(forward_select$model)],
      caption = '(1c)Model S: Variables Selected')
```

```
# Finally, we report the coefficients of the model (stored in
# forward_select$coefficients in increasing order of absolute values)
kable(
forward_select$coefficients[order(abs(forward_select$coefficients))],
      caption = '(1c)Model S: Coefficients')
```

```
# Backward selection starts with a full model and at each step
# eliminates a variable that has little or no significance to the
# prediction process. We see a small increase in the AIC value with
```

Table 5: (1c)Model S: Variables Selected

x
concavepoints.worst
radius.worst
area
texture
concavepoints
area.worst
perimeter.worst
radius.stderr
smoothness.worst
symmetry.worst
compactness
concavity.stderr
fractaldimension.worst
radius
perimeter

Table 6: (1c)Model S: Coefficients

	x
area	-0.0003773
area.worst	-0.0008454
perimeter.worst	-0.0048292
perimeter	-0.0049809
texture	0.0156029
radius	0.0448543
radius.worst	0.1713112
radius.stderr	0.3631839
symmetry.worst	0.7236488
concavepoints.worst	0.8301920
concavity.stderr	1.1258167
(Intercept)	-1.4836773
smoothness.worst	2.5027397
compactness	-3.4819636
fractaldimension.worst	4.4953955
concavepoints	5.6532651

Table 7: (1d)Pchisq-value

x
0.943856

```
# every step. Also expect a gradual decrease in prediction error
# with each step.
# Forward selection on the other hand starts with a null model.
# We start with trying to fit the best one predictor model that has
# lead to statistically significant reduction of AIC. The next step
# continues with all possible 2 predictor models and deciding the one
# that lead to significant decrease of AIC. This goes on and on
# until we get included required number of variables. Also expect a
# gradual decrease in prediction error with each step.trying to fit
# the best one predictor model that is statistically significant to
# outcome.
```

###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()`.

```
# Here we carry out the p-chi square test to test the goodness
# of fit of models B and S.
# First, we calculate degrees of freedom which is simply the
# absolute difference in number of variables of both models.
dfreedom<-abs(ncol(sel.backward$model)-ncol(forward_select$model))
# We calculate p-chisquare at "dfreedom" degrees of freedom.
p<-pchisq(deviance(sel.backward)-deviance(forward_select),
          df=dfreedom,
          lower.tail = FALSE)
# Return p-value using kable().
kable(p,caption = '(1d)Pchisq-value')
```

```
kable(c(summary(sel.backward)$adj.r.square,
summary(forward_select)$adj.r.square),
caption = '(1d)Adjusted R Squared values Models: B and S')
```

Table 8: (1d)Adjusted R Squared values Models: B and S

x
0.7288784
0.7287561

```
### Analysing Goodness of fit:
# Upon computing the pchisq-square value between models B and S,
# we observe that the value is very high (94.38%) than significance
# level alpha (0.05). This indicates that the models are not
# significantly different from each other and have more or less the
# same distribution.
# Furthermore, on comparing the adjusted r squared values, the
# backward selection model seems to have a slight edge over forward
# selection.
```

Problem 1.e (2 points)

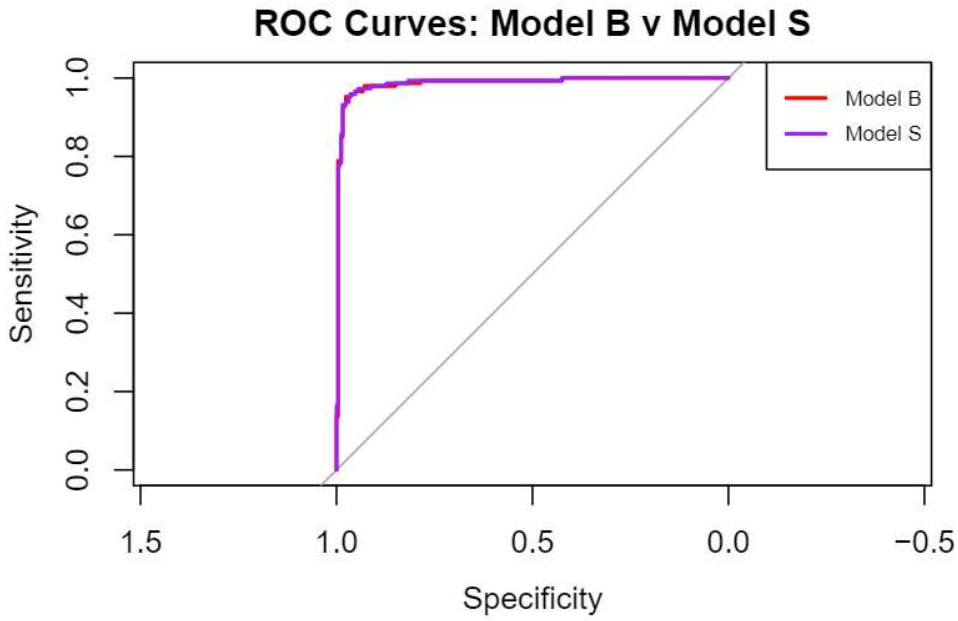
- Plot the ROC curve of the trained model for both **model B** and **model S**. Display with clear title, label and legend.
- Report AUC values in 3 significant figures for both **model B** and **model S** using `kable()`.
- Discuss which model has a better performance.

```
# Plot an ROC curve to compare both models.
suppressMessages(invisible({
  # First, create roc objects for both models (roc1
  # and roc2 respectively).
  roc1<-roc(Tr_set$diagnosis,
            sel.backward$fitted.values)
  roc2<-roc(Tr_set$diagnosis,
            forward_select$fitted.values)
  # Now we plot both roc curves with color of each plot specified
  # using the 'col' argument.
  plot(roc1,main='ROC Curves: Model B v Model S',col='red')
  # The add=TRUE argument ensures that this plot is added to the
  # first plot.
  plot(roc2,add=TRUE,
       col='purple')
  # Finally, we create a legend for the combined plot using the
```

Table 9: (1e)Stepwise models and their AUCs

Model	AUC
Model B	0.987
Model S	0.987

```
  # legend function.
  legend('topright',
         legend = c('Model B','Model S'),
         col=c('red','purple'),
         lty=c(1,1),lwd = c(2,2),cex=0.7)
}))
```




```
# The final objective is to report the AUC of both curves. This
# value is stored in the 'auc' attribute of the roc object. The
# round() function is used to round the value to 3 decimal places.
results<-data.table('Model'=c('Model B','Model S'),
                    'AUC'=c(round(roc1$auc,3),round(roc2$auc,3))
                           )
kable(results,
      caption='(1e)Stepwise models and their AUCs')
```

```
### Which model has better performance?
# The AUCs in both models appear to be almost the same with Model S
# being slightly better than Model B (0.9872 v 0.9869). Typically,
# we can expect both models in this case to perform almost the same.
# Since the number of variables under consideration is large (30
# variables), forward selection seems to be a much better alternative
# than backward selection. Thus, we see that forward selection
# slightly has the edge.
```

Problem 1.f (6 points)

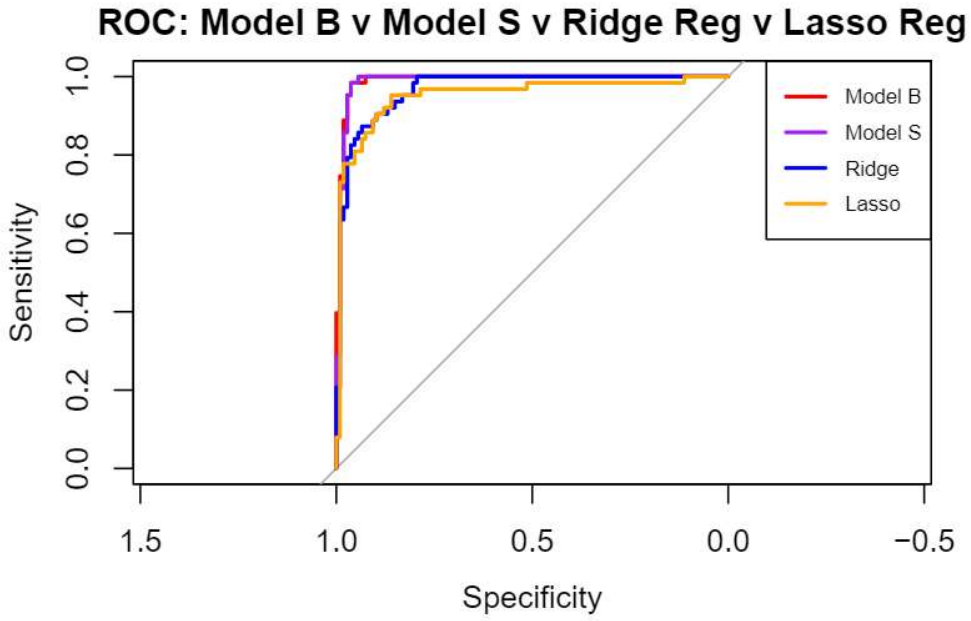
- Use the four models to predict the outcome for the observations in the test set (use the λ at 1 standard error for the penalised models).
- Plot the ROC curves of these models (on the sameplot, using different colours) and report their test AUCs.
- Display with clear title, label and legend.
- Compare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

```
# Convert test data table to matrix using as.matrix()
ts_set<-as.matrix(ts_set)
# Next we seperate the outcome variable and the predictor variables
# as yts_set and xts_set respectively.
# Columns 3 to 32 are the dependent variables and are is assigned
# to xtr_set.
xts_set<-ts_set[,3:32]
# The second column is the independent variable and it is assigned
# to yts_set.
yts_set<-ts_set[,2]
# Now we make predictions on outcome using the predict() function
# from the glmnet class and thus we see the 'newx' argument.
# Lambda is set at 1 standard error for the penalized models
# as follows:
pred1<-predict(fit.cv.ridge,
               newx = xts_set,
               type='response',
               s='lambda.1se')
pred2<-predict(fit.cv.lasso,
               newx = xts_set,
               type='response',
```

```
                s='lambda.1se')
# Convert xts_set to data frame to carry out predictions for
# the stepwise selection models.
xts_set<-as.data.frame(xts_set)
# Next we make predictions for stepwise linear models. This time
# the predict() function has the 'newdata' argument.
# Predicted values for stepwise backward selection.
pred3<-predict(sel.backward,
               newdata = xts_set)
# Predicted values for stepwise forward selection.
pred4<-predict(forward_select,
               newdata = xts_set)
# 'suppressMessages' helps to suppress any unwanted messages that
# may be returned.
suppressMessages(invisible({
  # Now we create roc() objects for all 4 models as follows:
  roc1<-roc(yts_set,pred3)
  roc2<-roc(yts_set,pred4)
  roc3<-roc(yts_set,pred1)
  roc4<-roc(yts_set,pred2)
  # Now we plot all 4 ROC curve by passing the objects to the
  # 'plot' function.To distinguish between plots, we give each plot
  # a colour using the 'col' argument.
  plot(roc1,
       main='ROC: Model B v Model S v Ridge Reg v Lasso Reg',
       col='red')
  # add=TRUE adds the current plot to the first plot. We add the
  # other plots in a similar way.
  plot(roc2,add=TRUE,col='purple')
  plot(roc3,add=TRUE,col='blue')
  plot(roc4,add=TRUE,col='orange')
  # Finally, we create a legend for the combined plot using the
  # legend() function.
  legend('topright',
         legend = c('Model B','Model S','Ridge','Lasso'),
         col=c('red','purple','blue','orange'),
         lty=c(1,1),lwd = c(2,2),cex=0.7)

}))
```

Table 10: (1f)AUCs of models

Model	AUC
Model B	0.990
Model S	0.989
Ridge Reg	0.968
Lasso Reg	0.952




```r
# The auc of each model can be found in the 'auc' attribute
# of the roc object. They are all tabulated into a data table
# as follows:
x<-data.table(
  'Model'=c('Model B','Model S','Ridge Reg','Lasso Reg'),
  'AUC' =c(round(roc1$auc,3),round(roc2$auc,3),
           round(roc3$auc,3),round(roc4$auc,3)
             ))
# Return the table using kable().
kable(x,caption = '(1f)AUCs of models')
```

```r
# Based on the train test AUCs, it can be observed that Ridge
# Regression (0.968 > 0.964), Model B (0.990>0.987) and Model S
# (0.989>0.987) show higher test AUCs compared to training AUCs.
```

```
#
# But on the whole, the train and test AUCs seem to be almost the
# same and this shows that these models are ideal fits and generalize
# well.
#
# Finally, we have Lasso Regression which gives which gives a lower
# Test AUC compared to Train AUC (0.952 < 0.966). Thus, this seems to
# generalize poorly compared to other models. Overall, Model B seems
# to have a better tradeoff between Train and Test AUC.
```

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()`.

```
# In order to impute all null values to median we first create
# an impute.to.median() function that takes the data table
# and imputes missing values as median.
impute.to.median <- function(x) {
                      # only apply to numeric/integer columns
                      if (is.numeric(x) || is.integer(x)){
                        # find which values are missing
                        na.idx <- is.na(x)
# replace NAs with the median computed over the observed values
                        x[na.idx] <- median(x, na.rm = TRUE)
                      }
                      # return the vector with imputed values
                      return(x)
}
# Read the GDM data using fread()
gdm.dt<-fread('GDM.raw.txt')
# Store the column names with numeric values
numcols <- gdm.dt %>%
  select_if(is.numeric) %>% colnames
# For all numeric columns, call impute.to.median()
gdm.dt <- gdm.dt %>%
                        .[, (numcols) := lapply(.SD,
#gdm.dt <- gdm.dt %>% copy %>%
 #                        .[, (numcols) := lapply(.SD,
# Return first 10 rows and first 7 columns using kable().
kable(head(gdm.dt,10)[,1:7],caption = '(2a)Imputed gdm.dt')
```

Table 11: (2a)Imputed gdm.dt

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

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 `SNPs`, `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.

```
# We create a function univ.glm.test() that takes a data table
# of SNPs (x), outcomes (y) and order flag as arguments and applies
# Logistic Regression to all 176 SNPs.
univ.glm.test<-function(x,y,order=FALSE){
   # Initialize a data frame 'res'.
   res<-NULL
  for(i in 1:ncol(x)){
    # For all columns:
    # Store ith column
    xi<-x[,i]
    # Fit Logistic Regression model to ith column or ith SNP.
    logistic_x<-glm(y~xi,family = binomial(link = 'logit'))
    # For each SNP, we return SNP names, regression coefficients,
    # odds ratios, standard errors and p-values.
    # The whole set of values for each SNP is appended to res
    # using rbind().
    res<-rbind(res,
```

```
                c(
                  colnames(x)[i],
                  logistic_x$coefficients[2],
                  exp(coef(logistic_x)[2]),
                  coef(summary(logistic_x))[-1, "Std. Error"],
                  coef(summary(logistic_x))[-1, "Pr(>|z|)"])
                )
  }
  # Assign column names to resultant values
   colnames(res)<-c('SNPName',
                    'Reg_Coefficient',
                    'Odds_Ratio',
                    'Std_Error',
                    'Pvalue')
  # Convert data frame to data table
   res<-data.table(res)
  # Check if order=TRUE
   if(order==TRUE){
   # Order rows by increasing order of Pvalues.
        res<-res[with(res,order(Pvalue))]
   }
   # Return res
  return(res)
}
```

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()`.

```
# Store the SNP value columns into xtr in matrix form using
# as.matrix().
xtr<-as.matrix(gdm.dt[,4:ncol(gdm.dt)])
# Store outcome variable in ytr
```

```r
ytr<-gdm.dt$pheno
# The outputs of the function univ.glm.test() are all stored in
# data table gdm.as.dt.
gdm.as.dt<-univ.glm.test(xtr,ytr)
# When we are looking for the SNP that is most strongly associated
# to increased risk of gestational diabetes, we are looking for the
# SNP that maximizes the chance of disease and thus the SNP with the highest odds ratio.
max_likely <-gdm.as.dt[Odds_Ratio==max(Odds_Ratio),
                       SNPName]
# Returns "rs1423096_T"
# When we are looking for the SNP with most significant protective
# effect, we are looking for the SNP that minimizes the chance of
# disease and thus the SNP with the least odds ratio.
sig_pro <-gdm.as.dt[Odds_Ratio==min(Odds_Ratio),
                    SNPName]
# Returns "rs11575839_C"
# We convert xtr from matrix to dataframe using data.frame(). This
# is done in order to pass it to the logistic regression model.
xtr<-data.frame(xtr)
# Fit a logistic regression model with outcome variable (pheno) and
# the strong risk associated SNP
logi_str_risk<-glm(ytr~rs1423096_T,
                   data = xtr,
                   family = binomial(link = 'logit'))
# Fit a logistic regression model with outcome variable (pheno) and
# the lowest risk associated SNP
logi_pro_effect<-glm(ytr~rs11575839_C,
                   data = xtr,
                   family = binomial(link = 'logit'))
```

```r
# Return first 10 values of gdm.as.dt using head().
kable(head(gdm.as.dt,10),caption = 'gdm.as.dt')|>
  kable_styling(full_width = F,
                position = "center",
                latex_options = "hold_position")
```

```r
# Report summary statistics of the 2 focus SNPs
# (max_likely and sig_pro) using kable().
kable(coef(summary(logi_str_risk)),
      caption = "(2c)SNP with Strong Risk Associated")
```

Table 12: gdm.as.dt

SNPName	Reg_Coefficient	Odds_Ratio	Std_Error	Pvalue
rs7513574_T	0.00215749439605489	1.00215982346177	0.105137211018395	0.983627959659055
rs1627238_A	0.114637864262422	1.1214672406608	0.113822428263324	0.313855918679796
rs1171278_C	0.121409444710005	1.12908711648435	0.11380728528091	0.286062822255494
rs1137100_A	0.060104751291512	1.06194778112087	0.1104238492224	0.586228503617821
rs2568958_A	0.149379896281938	1.16111400832769	0.123380016627863	0.225998873574341
rs1514175_A	0.0562296257329725	1.05784056326456	0.105235896141762	0.593120277525613
rs1555543_C	-0.0777805378818058	0.925167443472291	0.105783590355313	0.462169103637855
rs10923931_C	-0.214774420994388	0.806723399515719	0.18251171399461	0.239287076198348
rs516636_A	0.0552265983291012	1.05678005214045	0.123343524188903	0.654336351046674
rs574367_G	0.0590072860418808	1.06078296962115	0.124480094024075	0.635478588347025

Table 13: (2c)SNP with Strong Risk Associated

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.0824139	0.0734054	1.122722	0.2615556
rs1423096_T	0.6510641	0.3166547	2.056069	0.0397758

```
kable(coef(summary(logi_pro_effect)),
      caption = "(2c)SNP with Most Significant Protective Effect")
```

```
# Initialize a dataframe to store 95% and 99% condifence intervals
# of the odds ratios of the 2 focus SNPs.
conf_ints<-NULL
# The first row will have 95% and 99% confidence limits of the
# strong risk SNP and this row is appended to conf_ints using rbind().
# The confidence limits are all are rounded to 3 decimal places
# using round().
conf_ints<-rbind(conf_ints,
            c('Strong Risk',
            round(exp(confint(logi_str_risk)),3)[2,],
            round(exp(confint(logi_str_risk,level=0.99)),3)[2,]
```

Table 14: (2c)SNP with Most Significant Protective Effect

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.1427220	0.0728282	1.959707	0.0500300
rs11575839_C	-0.6022542	0.3758156	-1.602526	0.1090394

```
                              ))
```

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

```
# The second row will have 95% and 99% confidence limits of the
# protective effect SNP () and this row is appended to conf_ints
# using rbind(). The confidence limits are all are rounded to 3
# decimal places using round().
conf_ints<-rbind(conf_ints,
           c('Protective Effect',
            round(exp(confint(logi_pro_effect)), 3)[2,],
            round(exp(confint(logi_pro_effect,level=0.99)), 3)[2,]
                              ))
```

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

```
# Assign column names to resultant dataframe.
colnames(conf_ints)<-c('SNP Type',
                       '95% CI:(2.5%)',
                       '95% CI:(97.5%)',
                       '99% CI:(0.5%)',
                       '99% CI:(99.5%)'
                       )
# Print conf_ints
kable(conf_ints,
      caption = '(2c)Odds Ratio Confidence Intervals (by SNP)')|>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 15: (2c)Odds Ratio Confidence Intervals (by SNP)

SNP Type	95% CI:(2.5%)	95% CI:(97.5%)	99% CI:(0.5%)	99% CI:(99.5%)
Strong Risk	1.05	3.67	0.874	4.567
Protective Effect	0.255	1.131	0.198	1.419

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 `SNP`s that have p-value $< 10^{-4}$ (`hit SNP`s) 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.***

```
# Read file GDM.annot.txt into GDM.annot
GDM.annot<-fread('GDM.annot.txt')
# Now from a given SNP (Ex: s1234_X), we need to separately extract
# the id (s1234) and allele (X).
# snp id can be extracted from SNP by removing the last 2 characters
# ("s1234"_X) and that is extracted (substr(x,1,length(x)-2))
# as follows:
snp<-sapply(gdm.as.dt[,1],
            function(i)
              substr(i,1,nchar(i)-2))
# The last character in the snp string is the allele and that is
# extracted (substr(x,length(x),length(x))) as follows:
allele<-sapply(gdm.as.dt[,1],
               function(i)
                 substr(i,nchar(i),nchar(i)))
# Create a copy of gdm.as.dt computed previously
gdm.as.dt_copy<-gdm.as.dt %>% copy
# We are going to replace SNPName column with two new columns
# and hence it is first set to null.
gdm.as.dt_copy[,SNPName:=NULL]
# The two new columns snp and allele are inserted into the
# data table
gdm.as.dt_copy[,`:=`(snp=snp,allele=allele)]
gdm.merged<-merge.data.table(gdm.as.dt_copy,
                             GDM.annot,
                             by.x='snp',
                             by.y='snp',
                             all = TRUE)
```

```r
# The hit SNPs are first filtered out using as.numeric(Pvalue)<1e-4
# condition.
# Then we specify required columns: snp id, affected allele,
# chromosome number, gene and pos and return first 10 rows using
# kable as follows:
kable(head(gdm.merged[as.numeric(Pvalue)<1e-4,
                c('snp','allele','chrom','gene','pos')],10),
                  caption = '2d:Hit SNPs')|>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 16: 2d:Hit SNPs

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

```r
# Next for each hit snp, we return genes (gene) that are within a
# 1Mb window from the snp position (pos) on the chromosome. This
# process can be automated using sapply() on the hit snp positions.
# Also note that there is a chance of repeated gene values that is
# eliminated by using unique().
# The result is stored in snp.hit.genes.
snp.hit.genes<-sapply(gdm.merged[as.numeric(Pvalue)<1e-4,pos],
  function(i)
unique(gdm.merged[as.numeric(Pvalue)<1e-4 & pos>=i-10^6 & pos<=i+10^6, gene]))
# Return the first 10 values using kable().
kable(head(snp.hit.genes,10),caption = '(2d)Genes (by hit SNP)')
```

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.***

Table 17: (2d)Genes (by hit SNP)

x
TCF7L2
KCNQ1

- 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
# First, let us store the rows that have Pvalue<10^-4 in SNPs_grs1.
SNPs_grs1<-gdm.as.dt[as.numeric(Pvalue)<1e-4]
# The SNPs that satisfy Pvalue<10^-4 in the 'SNPName' column
# of gdm.as.dt are all passed to SDcols. SDcols helps to select
# these columns alone and pass them to gdm_dt_snps1.
gdm_dt_snps1<- gdm.dt[, .SD,
                      .SDcols = gdm.as.dt[as.numeric(Pvalue) < 1e-4]$SNPName]
# Check ordering of SNPs using the statement below:
# stopifnot(colnames(gdm.dt)[-1] == gdm.as.dt$SNPName)
# Now we do a matrix product of gdm_dt_snps1 and Regression
# coefficients stored in SNPs_grs1 to get weighted score:
weighted.score1 <- as.matrix(gdm_dt_snps1) %*%
  as.numeric(SNPs_grs1$Reg_Coefficient)

# First, let us store the rows that have Pvalue<10^-3 in SNPs_grs2.
SNPs_grs2<-gdm.as.dt[as.numeric(Pvalue)<1e-3]
# The SNPs that satisfy Pvalue<10^-3 in the 'SNPName' column
# of gdm.as.dt are all passed to SDcols. SDcols helps to select
# these columns alone and pass them to gdm_dt_snps2.
gdm_dt_snps2 <- gdm.dt[, .SD,
                       .SDcols = gdm.as.dt[as.numeric(Pvalue)<1e-3]$SNPName]
# Check ordering of SNPs using the statement below:
# stopifnot(colnames(gdm.dt)[-1] == gdm.as.dt$SNPName)
# Now we do a matrix product of gdm_dt_snps2 and Regression
# coefficients stored in SNPs_grs2 to get weighted score:
weighted.score2 <- as.matrix(gdm_dt_snps2) %*%
  as.numeric(SNPs_grs2$Reg_Coefficient)

# First let us store the rows that have gene=='FTO' in SNPs_grs3.
SNPs_grs3<-gdm.merged[gene=='FTO']
# The SNPs that satisfy gene = 'FTO' in the 'SNPName' column
# of gdm.as.dt are all passed to SDcols. SDcols helps to select
# these columns alone and pass them to gdm_dt_snps3.
gdm_dt_snps3 <- gdm.dt[, .SD,
                .SDcols = gdm.merged[gene=='FTO',
                                     paste(snp,allele,sep='_')]]
# Check ordering of SNPs using the statement below:
```

```r
# stopifnot(colnames(gdm.dt)[-1] == gdm.merged$snp)
# Now we do a matrix product of gdm_dt_snps3 and Regression
# coefficients stored in SNPs_grs3 to get weighted score:
weighted.score3 <- as.matrix(gdm_dt_snps3) %*%
  as.numeric(SNPs_grs3$Reg_Coefficient)
# Lets add all of these weighted scores to gdm.dt
gdm.dt<-gdm.dt[,`:=`(wgrsscore1=weighted.score1,
                     wgrsscore2=weighted.score2,
                     wgrsscore3=weighted.score3)]
# Create Logistic Regression models with 'pheno' being the outcome
# variable and weighted score 1 being predictor.
mod.weighted1<-glm(pheno~wgrsscore1,
                   data = gdm.dt,
                   family = binomial(link = 'logit'))
# Create Logistic Regression models with 'pheno' being the outcome
# variable and weighted score 2 being predictor.
mod.weighted2<-glm(pheno~wgrsscore2,
                   data = gdm.dt,
                   family = binomial(link = 'logit'))
# Create Logistic Regression models with 'pheno' being the outcome
# variable and weighted score 3 being predictor.
mod.weighted3<-glm(pheno~wgrsscore3,
                   data = gdm.dt,
                   family = binomial(link = 'logit'))
# Now we store the odds ratio, 95% Confidence interval and P-value
# of the 3 weighted models in data table 'result_table'.
result_table<-data.table(
  'Weighted Score'=c(1,2,3),
  'Odds Ratio'=c(
      round(exp(coef(mod.weighted1)[2]),3),
      round(exp(coef(mod.weighted2)[2]),3),
      round(exp(coef(mod.weighted3)[2]),3)
                          ),
  '95% CI:(2.5%)'=c(
      round(confint(mod.weighted1)[2,1],3),
      round(confint(mod.weighted2)[2,1],3),
      round(confint(mod.weighted3)[2,1],3)
                          ),
  '95% CI:(97.5%)'=c(
      confint(mod.weighted1)[2,2],
      confint(mod.weighted2)[2,2],
```

Table 18: (2e)Weighted Scores Model Results

Weighted Score	Odds Ratio	95% CI:(2.5%)	95% CI:(97.5%)	Pvalue
1	2.729	0.655	1.3638064	0.000
2	1.452	0.248	0.5014574	0.000
3	1.414	-0.200	0.8971549	0.215

```
      confint(mod.weighted3)[2,2]
                            ),
  'Pvalue'=c(
    round(coef(summary(mod.weighted1))[-1, "Pr(>|z|)"],3),
    round(coef(summary(mod.weighted2))[-1, "Pr(>|z|)"],3),
    round(coef(summary(mod.weighted3))[-1, "Pr(>|z|)"],3)
    )
                          )
```

```
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...
```

```
# Print result_table using kable()
kable(result_table,
      caption = '(2e)Weighted Scores Model Results')
```

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 the test set using fread()
gdm.test<-fread('GDM.test.txt')
# Find the SNPs in gdm.merged[as.numeric(Pvalue)<1e-4] that are
# in gdm.test and store in 'x1'.
x1<-intersect(colnames(gdm.test)[4:179],
              gdm.merged[as.numeric(Pvalue)<1e-4,snp])
# Filter out the rows using as.numeric(Pvalue)<1e-4 and 'x1'
SNPs_grs4<-gdm.merged[as.numeric(Pvalue)<1e-4 & snp %in% x1]
# Extract the columns from 'x1'
gdm_dt_snps4<- gdm.test[, .SD,.SDcols = x1]
# Now we do a matrix product of gdm_dt_snps4 and Regression
# coefficients stored in SNPs_grs4 to get weighted score:
weighted.score4 <- as.matrix(gdm_dt_snps4) %*%
  as.numeric(SNPs_grs4$Reg_Coefficient)

# Find the SNPs in gdm.merged[as.numeric(Pvalue)<1e-3] that are in
# gdm.test and store in 'x2'.
x2<-intersect(colnames(gdm.test)[4:179],
              gdm.merged[as.numeric(Pvalue)<1e-3,snp])
# Filter out the rows using as.numeric(Pvalue)<1e-3 and 'x2'
SNPs_grs5<-gdm.merged[as.numeric(Pvalue)<1e-3 & snp %in% x2]
# Extract the columns from 'x2'
gdm_dt_snps5<- gdm.test[, .SD,.SDcols = x2]
# Now we do a matrix product of gdm_dt_snps5 and Regression
# coefficients stored in SNPs_grs5 to get weighted score:
weighted.score5 <- as.matrix(gdm_dt_snps5) %*%
  as.numeric(SNPs_grs5$Reg_Coefficient)

# Find the SNPs in gdm.merged[gene=='FTO'] that are in gdm.test
# and store in 'x3'.
x3<-intersect(colnames(gdm.test)[4:179],
              gdm.merged[gene=='FTO',snp])
# Filter out the rows using gene=='FTO' and 'x3'
SNPs_grs6<-gdm.merged[ gene=='FTO' & snp %in% x3]
# Extract the columns from 'x3'
gdm_dt_snps6<- gdm.test[, .SD,.SDcols = x3]
# Now we do a matrix product of gdm_dt_snps6 and Regression
# coefficients stored in SNPs_grs6 to get weighted score:
weighted.score6 <- as.matrix(gdm_dt_snps6) %*%
  as.numeric(SNPs_grs6$Reg_Coefficient)
# Lets add all of these weighted scores to gdm.test
```

Table 19: (2g)Weighted Models

Weighted Model	Log-likelihood
1	-407.774
2	-1074.865
3	-1036.955

```
gdm.test<-gdm.test[,`:=`(wgrsscore1=weighted.score4,
                         wgrsscore2=weighted.score5,
                         wgrsscore3=weighted.score6)]
```

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()`

```
# Use the previous models to make predictions  for the 3 genetic
# risk scores using the test data.
pred1<-predict(mod.weighted1,
               newdata=gdm.test)
pred2<-predict(mod.weighted2,
               newdata=gdm.test)
pred3<-predict(mod.weighted3,
               newdata=gdm.test)
# Next, calculate combined Log Likelihood for all 3 predictors. The
# dbinom() function is used for this purpose. We are interested in a
# single trial and hence size=1 and log=TRUE ensures that the
# probabilities
# are passed as# as
L<-c(sum(dbinom(gdm.dt$pheno,prob=pred1,size=1,log = TRUE),
         na.rm = TRUE),
sum(dbinom(gdm.dt$pheno,prob=pred2,size=1,log = TRUE),
    na.rm = TRUE),
sum(dbinom(gdm.dt$pheno,prob=pred3,size=1,log = TRUE)))
# Print data table using kable().
kable(data.table('Weighted Model'=c(1,2,3),'Log-likelihood'=L),
      caption = '(2g)Weighted Models')
```

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()`.

```
# Read summary statistics from GDM.study2.txt using fread()
gdm.study<-fread('GDM.study2.txt')
gdm.effect<- gdm.merged %>% copy
gdm.effect<-gdm.effect[with(gdm.effect,order(Pvalue))]
# Harmonise the variables of study
gdm.study<-gdm.study[snp %in% gdm.effect$snp]
gdm.effect<-gdm.effect[snp %in% gdm.study$snp]
```

```
gwas1 <- gdm.study[order(snp)]
gwas2 <- gdm.effect[order(snp)]
# all.equal(gdm.effect$snp,gdm.study$snp): TRUE

both.ok <-gwas2[as.numeric(Pvalue)<1e-4,allele] == gwas1$effect.allele
flipped <-gwas2[as.numeric(Pvalue)<1e-4,allele] == gwas1$other.allele

kable(
  table(both.ok, flipped)
      ,caption = "(2h)SNPs with Alleles Matching") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 20: (2h)SNPs with Alleles Matching

	FALSE	TRUE
FALSE	81	44
TRUE	51	0

```
# Initialize effect sizes
Beta1 <- as.numeric(gdm.effect$Reg_Coefficient)
Beta2 <- as.numeric(gdm.study$beta)
Beta2[flipped] <- -Beta2[flipped]
# Calculate inverse variance weights
```

```r
wgwas1 <- 1 / as.numeric(gdm.effect$Std_Error)^2
wgwas2 <- 1 / gdm.study$se^2
# Print weights of first GWAS
kable(t(head(wgwas1)),
      caption = "(2h)Weight of 1st GWAS (wgwas1)") |>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 21: (2h)Weight of 1st GWAS (wgwas1)

81.38295	55.17323	52.35296	55.4531	49.45205	29.02107

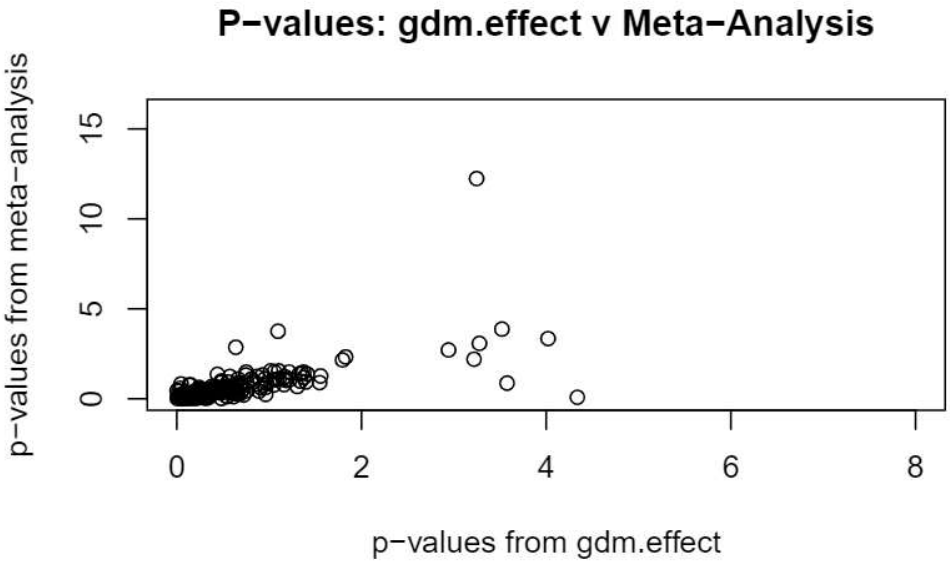
```r
# Print weights of second GWAS
kable(t(head(wgwas2)),
      caption = "(2h)Weight of 2nd GWAS (wgwas2)") |>
  kable_styling(full_width = F, position = "center", latex_options = "hold_position")
```

Table 22: (2h)Weight of 2nd GWAS (wgwas2)

34.82747	8.361486	1.397415	38.53447	11.27683	9.14147

```r
# Upon observation, Study 1 looks more empowered
```

```r
# Meta-analysis effect size (beta.ma) is a weighted sum of the
# effect sizes.
beta.ma <- (wgwas1 * Beta1 + wgwas2 * Beta2) /
  (wgwas1 + wgwas2)
# Meta Analysis Standard Error
se.ma <- sqrt(1 / (wgwas1 + wgwas2))
# Meta Analysis P-values
pval.ma <- 2 * pnorm(abs(beta.ma / se.ma), lower.tail = FALSE)
# Plot P-values from 1st study v P-values from meta analysis
plot(-log10(as.numeric(gdm.effect$Pvalue)), -log10(pval.ma),
     xlim = c(0, 8), ylim = c(0, 16),
     xlab = "p-values from gdm.effect",
     ylab = "p-values from meta-analysis",
     main = "P-values: gdm.effect v Meta-Analysis"
     )
```


P−values: gdm.effect v Meta−Analysis
p−values from meta−analysis
0
5
10
15
0
2
4
6
8
p−values from gdm.effect

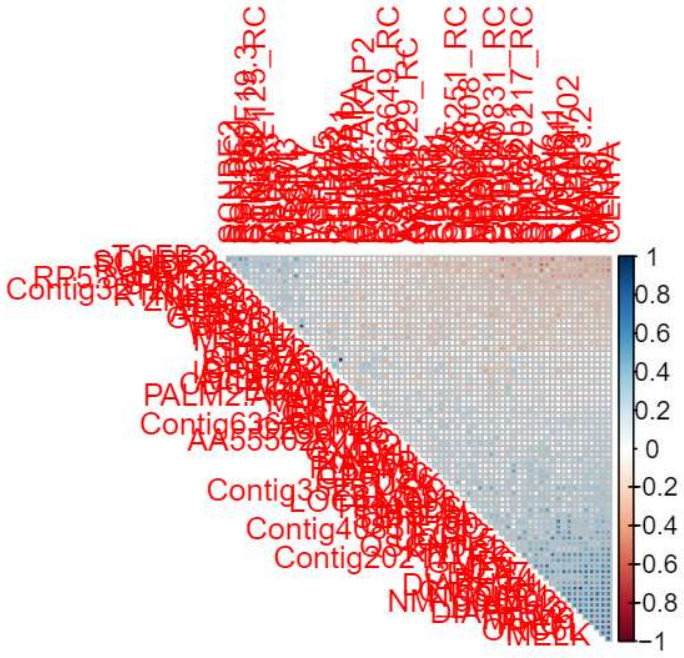
Problem 3 (33 points)

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 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.

```
# Read the data file using fread()
nki.dt<-fread('nki.csv')
# Create a new data table genes that contains all gene expression
# columns of nki.dt
genes<-data.table(nki.dt[,7:76])
# Store the list of names of numeric columns
numcols <- sapply(genes, is.numeric)
# Subset of numeric columns
cor.genes <- genes[, ..numcols] %>% #subset of numeric columns
             cor(use="pairwise.complete")
# Plot a Correlation plot of the 70 gene expressions
corrplot(cor.genes, method = 'square',
         order = 'FPC', type = 'upper', diag = FALSE)
```




```
### What we observe?
# The method that we use to compute the coreelation coefficient
# is the default one (Pearson). Since Corr(A,B) = Corr(B,A),
# we can skip one triangle of values in the matrix. (It should
# also be noted that the correlation of a variable with itself
# is 1 and thus the diagonals are all blues based on the color
# marker to the right.)
# In our case, we will be skipping the lower triangle using
# type= 'upper'. The positive correlations are displayed in blue
# while the negative correlations are displayed red. As we specified
# method = 'square', each cell in the matrix will have a sqaure in it,
# the size of which is dependent on the corresponding correlation
# coefficient.
```

```
### Identify variable pairs with cor > 0.8
K <- which(abs(cor.genes)>0.8 & row(cor.genes)<col(cor.genes),
           arr.ind=TRUE)
# Reconstruct names from positions
Corr80 <- matrix(colnames(cor.genes)[K],ncol=2)
# Bind the columns into Corr80
Corr80<-cbind(Corr80,cor.genes[K])
```

Table 23: (3a)Variable pairs with Correlation Coefficient > 0.8

Var1	Var2	Corr_Value
DIAPH3	DIAPH3.1	0.803136819514865
DIAPH3	DIAPH3.2	0.833859118546205
DIAPH3.1	DIAPH3.2	0.88687406089682
PECI	PECI.1	0.869783649528927
IGFBP5	IGFBP5.1	0.97750296924928
NUSAP1	PRC1	0.829835551228507
PRC1	CENPA	0.817542383966119

```
# Assign column names
colnames(Corr80)<-c('Var1','Var2','Corr_Value')
# Print the pairs with corr(A,B)>0.8
kable(Corr80,
      caption = '(3a)Variable pairs with Correlation Coefficient > 0.8')
```

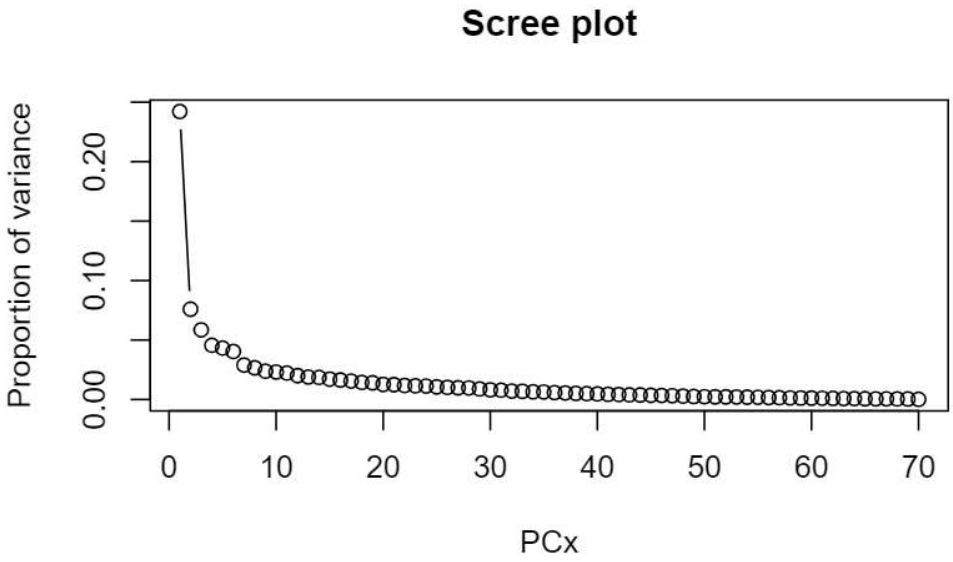
Problem 3.b (8 points)

- 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.

```
# Apply the prcomp() function which does required PCA on 'genes'
pca.v<-prcomp(genes,center = T,scale. = T)
# Fraction of variance covered by each variable
perc.expl <- pca.v$sdev^2 / sum(pca.v$sdev^2)

# Plot a scree plot to look and decide on which principle components
# to keep
plot(perc.expl, main="Scree plot",
          type = 'b',
          xlab='PCx',
          ylab='Proportion of variance'
          )
```

Table 24: (3b)Proportion of variance covered

x
0.9039862




```
# Proportion of variance covered
kable(sum(perc.expl[1:33]),
      caption = '(3b)Proportion of variance covered')
```

```
# Around 90.4%
```

```
### Which principle components to keep?
# Upon looking at the scree plot, the curve flattens after 40.
# This means that the first 40 components cover a major portion
# of the variance (around 94.3%). So it would suffice to choose
# the first 30 to 40 components. We set our threshold to components
# that cover atleast 90% of the variance. The first 33 components
# are just enough to satisfy this criteria (around 90.4%). And thus,
# that is the new dimensional space. We have cut down 70 components
# to just 33.
```

```r
# Create a copy of nki.dt
nki.dt.new<-nki.dt %>% copy
# Convert the categorical variables to numeric.
nki.dt.new$EstrogenReceptor<-c(Positive=1,
                               Negative=2)[nki.dt.new$EstrogenReceptor]
nki.dt.new$Grade<-c(Intermediate=1,
                    `Well diff`=2,
                    `Poorly diff`=3)[nki.dt.new$Grade]
# Adjusted Model: Adjusted with Age, EstrogenReceptor and Grade as  covariates.
nki.dt.new.lr<-glm(Event~Age+EstrogenReceptor+Grade,
                   data = nki.dt.new,
                   family = binomial(link = 'logit'))
# View the required coefficients in the summary of the model.
kable(coef(summary(nki.dt.new.lr)),
      caption = 'Adjusted Model')|>
  kable_styling(full_width = F, position = "center",
                latex_options = "hold_position")
```

Table 25: Adjusted Model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.2884817	1.6465208	0.7825480	0.4338926
Age	-0.0676552	0.0344309	-1.9649576	0.0494191
EstrogenReceptor	0.6209388	0.4728793	1.3131021	0.1891485
Grade	0.1268021	0.2282015	0.5556586	0.5784443

```r
# The unadjusted models are simply the models obtained by
# modelling the relationship between Event and each Principle
# Component seperately (For each PCx, Event ~ PCx).
# Conveniently, the univ.glm.test() from 2b, does this for us.
Unadj.mod<-univ.glm.test(pca.v$x[,1:33],
                         nki.dt.new$Event,
                         order=TRUE)
# Let's change the column name 'SNPName' to 'Principle_Componentx'
# for better understanding
colnames(Unadj.mod)[1] <- "Principle_Componentx"
# Now let us sort out the models in which the Principle
# Components are statistically significant in the prediction
# process. This models have their Pvalue less than level of
# significance (0.05).

```

```
# Unadj.mod[as.numeric(Pvalue)<0.05, Principle_Componentx]:
# PC19,PC3,PC1,PC23,PC11,PC14
```

```
### Adjusted and Non-Adjusted models:
# If we have a look at the adjusted model, only the age column seems
# to be statistical significant in the whole prediction process. This
# can be concluded as the other 2 variables,  EstrogenReceptor and
# Grade have much higher P values.

# In case of the Non-adjusted models, there are only 6 models in
# which the predictor variables are of statistical significance.
# The major difference that we need to observe is the in case of the
# adjusted model, the results are based on an adjusted combination
# of more than one variable (atleast one covariate in addition)
# (Age, EstrogenReceptor and Grade variables in our case). Whereas,
# in case of Non-adjusted models, there are no additional covariates
# and the modelling is only based on 1 variable.
```

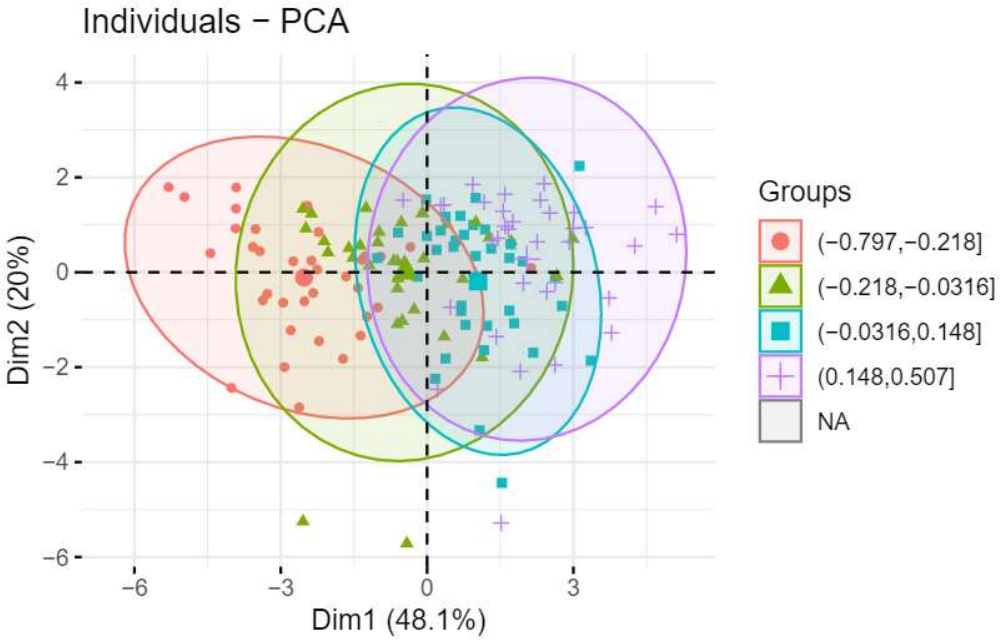
Problem 3.c (8 points)

- Use PCA plots to compare the main drivers with 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.

```
# Retrieve column names of the main drivers.
x<-c(unique(colnames(cor.genes)[K]))
# Apply pca on those drivers
pca.drivers<-prcomp(nki.dt[,..x],
                    center = T,
                    scale. = T)
# Now let us return all pca plots
# Graph of individuals
fviz_pca_ind(pca.drivers, geom = 'point',
             habillage = cut(nki.dt$NUSAP1,
                             #colour by NUSAP1
                            quantile(nki.dt$NUSAP1)),
              addEllipses = T)
```

```
Too few points to calculate an ellipse
```

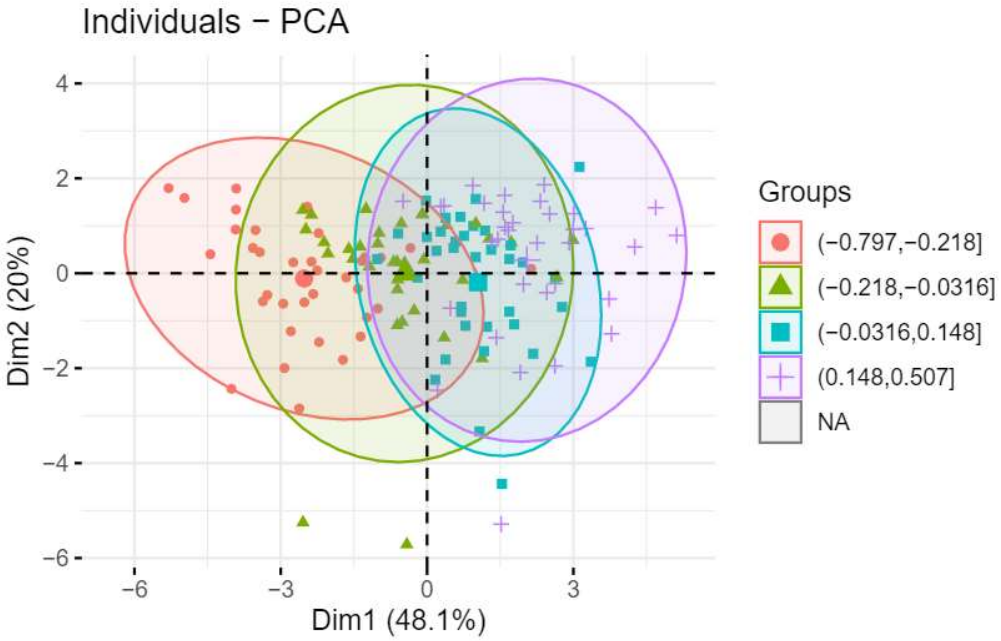
```
Warning: Removed 1 rows containing missing values (`geom_point()`).
Removed 1 rows containing missing values (`geom_point()`).
```



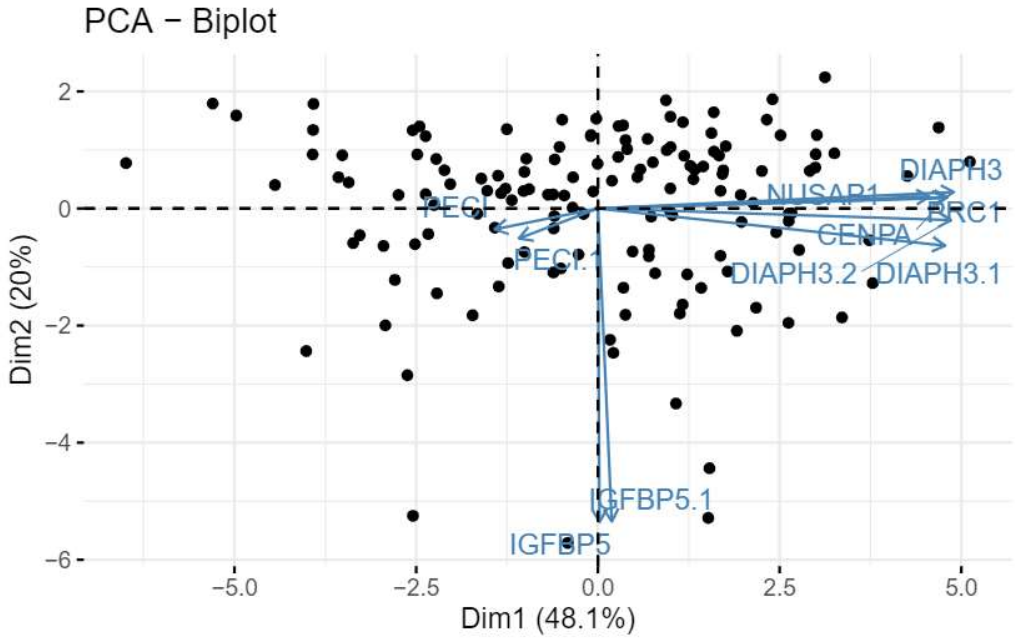

```r
# Graph of individuals with data points projected along
# the second and third Principle Components.
fviz_pca_ind(pca.drivers, geom = 'point',
             habillage = cut(nki.dt$NUSAP1, axes=c(2,3),
                             #colour by NUSAP1
                            quantile(nki.dt$NUSAP1)),
              addEllipses = T)
```

```
Too few points to calculate an ellipse
```

```
Warning: Removed 1 rows containing missing values (`geom_point()`).
Removed 1 rows containing missing values (`geom_point()`).
```




```
# Biplot
fviz_pca_biplot(pca.drivers, geom='point', repel = T)
```



```
### Observations:
# From the first 3 plots we observe that the ellipses overlap with
# each other. The centroids (shown with much bigger data markers)
# are also separated in the right order along the x-axis although
# this seperation is not seen along the y-axis.

# In the 2nd plot, we can look at the projections of variables drawn
# on a 2d plane. Upon looking at the extreme values (Values far
# left or far right of the axes), along Y-axis, we can observe that
# IGFBP5 and IGFBP5.1 have been assigned the highest magnitude by
# Principle Component 2. Similarly, along X-axis, we can observe that
# PECI and DIAPH3, DIAPH3.1 and PRC1 have been assigned the highest
# weights by Principle Component 1. This can also be observed in
# the correlation plot we did in 3a.

# From 3b, we know that only 6 variables (Out of 70) are associated
# with outcome (Event).This sums up the need to eliminate variables
# that are unassociated with outcome. An other idea might be to use
# the box plot for outlier detection.
```

Problem 3.d (11 points)

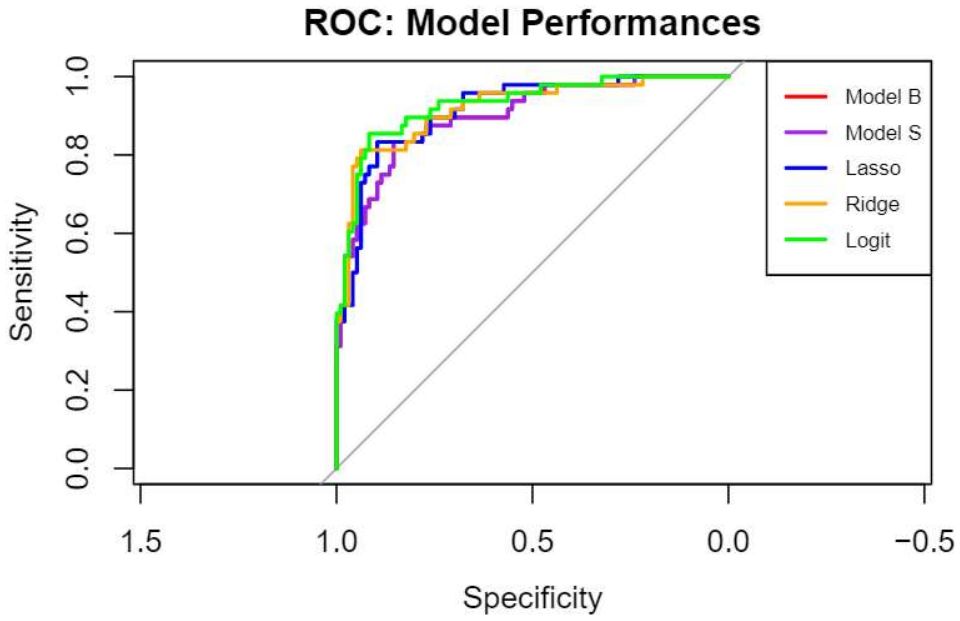
- Based on the models we examined in the labs, fit an appropriate model with the aim to provide the most accurate prognosis you can for patients.
- Discuss and justify your decisions with several experiments and evidences.

```
PCx<-data.frame(pca.v$x[,1:33])
# Create a Null Model with just theintercept as independent variable
Null.mod<-lm(nki.dt$Event~1,data=PCx)
# Create a Full Model with all independent variables
Full.mod<-lm(nki.dt$Event~.,data=PCx)
# Now we use these models to fit Stepwise selection models
# Stepwise backward selection model
backward.sel<-stepAIC(Full.mod,
                      direction = 'backward',
                      scope = formula(Full.mod),
                      trace = FALSE)
# Predictions based on above model
pred_1<-predict(backward.sel,newdata= PCx)

```

```r
# Stepwise forward selection model
forward.sel<- stepAIC(Null.mod,
                      direction = 'forward',
                      scope = formula(Full.mod),
                      trace = FALSE)
# Predictions based on above model
pred_2<-predict(forward.sel,newx= PCx)

Lasso.reg<-cv.glmnet(pca.v$x[,1:33],nki.dt$Event,
                     type.measure = 'auc',
                     family='binomial')
# Predictions based on above model
pred_3<-predict(Lasso.reg,newx=pca.v$x[,1:33])

# Ridge Regression
Ridge.reg<-cv.glmnet(pca.v$x[,1:33],nki.dt$Event,
                     alpha=0,
                     type.measure = 'auc',
                     family='binomial')
pred_4<-predict(Ridge.reg,newx= pca.v$x[,1:33])

# Logistic Regression: Most accurate model
log_reg<-glm(nki.dt$Event~.,
             data = PCx,
             family = binomial(link='logit'))
pred_5<-predict(log_reg,newdata= PCx)
```

```r
suppressMessages(invisible({
  # Now we create roc() objects for all 4 models as follows:
  roc1<-roc(nki.dt$Event,pred_1)
  roc2<-roc(nki.dt$Event,pred_2)
  roc3<-roc(nki.dt$Event,pred_3)
  roc4<-roc(nki.dt$Event,pred_4)
  roc5<-roc(nki.dt$Event,pred_5)
  # Now we plot all 5 ROC curve by passing the objects to the
  # 'plot' function.To distinguish between plots, we give each plot
  # a colour using the 'col' argument.
  plot(roc1,main='ROC: Model Performances',col='red')
  # add=TRUE adds the current plot to the first plot. We add the other
  # plots in a similar way.
  plot(roc2,add=TRUE,col='purple')
```

```
  plot(roc3,add=TRUE,col='blue')
  plot(roc4,add=TRUE,col='orange')
   plot(roc5,add=TRUE,col='green')
  # Finally, we create a legend for the combined plot using the
  # legend() function.
  legend('topright',
         legend = c('Model B','Model S','Lasso','Ridge','Logit'),
         col=c('red','purple','blue','orange','green'),
         lty=c(1,1),lwd = c(2,2),cex=0.7)

}))
```




```
# From observing the set of AUC curves from the above plot,
# Logistic Regression is the model that maximises AUC
# (AUC=0.9286)
```