BioA02

Wenyi Fang

3/24/2020

#Data and packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tibble)
library(dplyr)
library(tidyr)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(purrr)
library(ggplot2)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ISLR)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 3.0-2
```

```
library(arm)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: lme4
```

```
##
## arm (Version 1.10-1, built: 2018-4-12)
```

```
## Working directory is /Users/vanessafung/Desktop
```

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(stringr)
library(MASS)

setwd("/Users/vanessafung/Desktop")
results.dt = read.table(file = "assignment2/lipids.txt",header = T, sep = "\t")
lipid.class = read.table(file = "assignment2/lipid-classes.txt",header = F, sep = "\t")

wdbc2 = read.csv(file = "assignment2/wdbc2.csv")

gdm.dt = read.table("assignment2/GDM.raw.txt",header = T)
gdm.annot = read.table("assignment2/GDM.annot.txt",header = T,na.strings = "NA", sep = "\t")
gdm.test = read.table("assignment2/GDM.test.txt",header = T)
gdm.study2 = fread("assignment2/GDM.study2.txt",header = T)

nki = read.csv(file = "assignment2/nki.csv")
```

#Problem 01 ##01-(a)

```r
#divide lipid specious into 4 column to extract classes names out
results.dt[, c("v1", "v2", "v3", "v4")] = str_split_fixed(results.dt$lipid.species, " ", 4)
#no class infomation is included in v2,v4 therefore delete them
results.dt = results.dt[, -c(5, 7)]

#fix class name in results.dt to be the same as class name in lipid-classes list
for (i in c(4, 5)) {
  results.dt[, i][which(results.dt[, i] == "CER")] = "Cer"
  results.dt[, i][which(results.dt[, i] == "Dag")] = "DAG"
  results.dt[, i][which(results.dt[, i] == "Tag")] = "TAG"
  results.dt[, i][which(results.dt[, i] == "tag")] = "TAG"
}

#use merge() to annotate specious name to the results.dt
for (i in c(1, 3)) {
  results.dt = merge(
    results.dt,
    lipid.class,
    by.x = paste("v", i, sep = ""),
    by.y = c("V1"),
    all.x = T
  )
}

#tidy the results.dt
results.dt = results.dt %>%
  mutate(class.name = if_else(is.na(results.dt$V2.x), results.dt$V2.y, results.dt$V2.x)) %$%
  .[, -c(1, 2, 6, 7)]

#Count the number of lipids that fall in each class
cou.w.class = results.dt %>%
  group_by(class.name) %>%
  count()
cou.w.class
```

```
## # A tibble: 9 x 2
## # Groups:   class.name [9]
```

```
##   class.name                          n
##   <fct>                           <int>
## 1 Ceramides                          13
## 2 Cholesterol esters                  9
## 3 Diacylglycerols                    16
## 4 Lysophosphatidylcholines           13
## 5 Lysophosphatidylethanolamines       3
## 6 Phosphatidylcholines               42
## 7 Phosphatidylethanolamines          25
## 8 Phosphatidylserines                 8
## 9 Triacylglycerols                  147
```

##01-(b) The normal approximation is acceptable in this instance.Portion1 is the portion of norm.pvalue reject the null when t.pvalue reject the null,which is 1,Portion2 is the portion of norm.pvalue is insignificant when t.pvalue is insignificant, which is also 1. This means t.pvalue and norm.pvalue result in the same conclusions in this dataset, therefore the normal approximation is acceptable in this instance.

```
lipid.z.value = log(results.dt$oddsratio) / results.dt$se
results.dt = results.dt %>%
  mutate(
    t.p.value = 2 * pt(abs(lipid.z.value), 284, lower.tail = F),
    norm.p.value = 2 * pnorm(abs(lipid.z.value), lower.tail = F),
    zvalue = lipid.z.value
  )

#portion of pnorm and pt have the same significant conclution
portion1  = length(results.dt[which(results.dt$t.p.value <= 0.05 &
                                    results.dt$norm.p.value <= 0.05),]$lipid.species) / length(results
#portion of pnorm and pt have the same not significant conclution
portion2  = length(results.dt[which(results.dt$t.p.value > 0.05 &
                                    results.dt$norm.p.value > 0.05),]$lipid.species) / length(results

portion1#1
```

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

```
portion2#1
```

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

##01-(c)

```
holm.bonferroni = function(results.dt, alpha){
  parg.results = arrange(results.dt,t.p.value)
  k = 0
  while (parg.results$t.p.value[k+1]< alpha/(length(parg.results$t.p.value)+1-k-1)){
    k = k+1
  }
  print(k)
  return(parg.results[c(1:k),])
}
```

##01-(d)

```
benjamini.hochberg  =  function(results.dt, q){
  parg.results = arrange(results.dt,t.p.value)
  k = 0
  while (parg.results$t.p.value[k+1]<=(k+1)*q/length(parg.results$t.p.value)){
```

```
    k = k+1
  }
  print(k)
  return(parg.results[c(1:k),])
}
```

##01-(e)

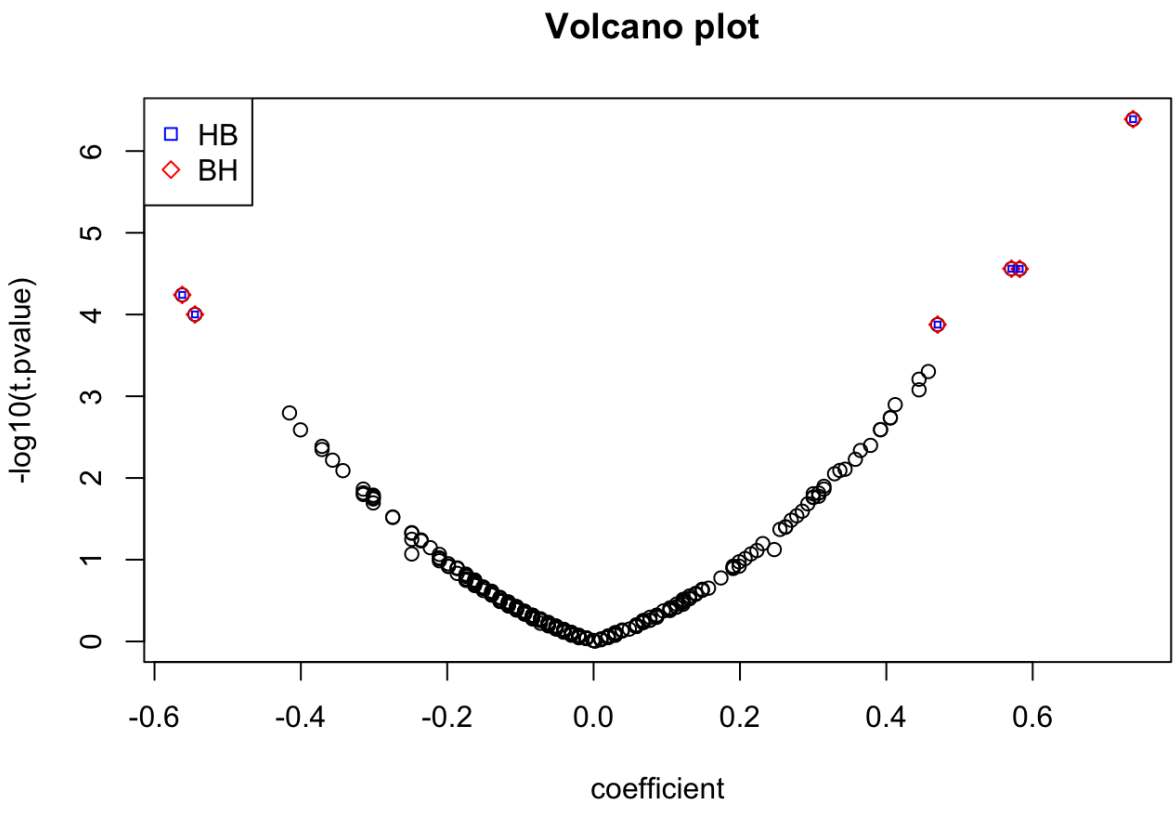
```
sub.df1 = holm.bonferroni(results.dt,0.05)
```

```
## [1] 6
```

```
sub.df2 = benjamini.hochberg(results.dt,0.01)
```

```
## [1] 6
```

```
{plot(log(results.dt$oddsratio),-log10(results.dt$t.p.value),col = "black" ,main = "Volcano plot ",xlab
points(log(sub.df2$oddsratio),-log10(sub.df2$t.p.value),col = "red",pch = 23)
points(log(sub.df1$oddsratio),-log10(sub.df1$t.p.value), pch=22, cex=0.5,col = "blue")
legend('topleft',legend = c("HB","BH"),col = c("blue","red"),pch = c(22,23),cex=1)}
```



##01-(f)

```
sub.df3 = benjamini.hochberg(results.dt,0.05)
```

```
## [1] 13
```

```
sub.df1;sub.df3
```

```
##   lipid.species oddsratio      se               class.name    t.p.value
## 1       PS 36:4      2.09 0.14212      Phosphatidylserines 4.071778e-07
## 2       PS 38:5      1.77 0.13395      Phosphatidylserines 2.752356e-05
## 3       PC 34:6      1.79 0.13662     Phosphatidylcholines 2.764683e-05
## 4      LPC 14:0      0.57 0.13763 Lysophosphatidylcholines 5.755634e-05
## 5      LPC 22:6      0.58 0.13801 Lysophosphatidylcholines 9.982223e-05
## 6       PC 34:2      1.60 0.12132     Phosphatidylcholines 1.329315e-04
##   norm.p.value    zvalue
## 1 2.138086e-07  5.186913
## 2 2.020331e-05  4.262632
## 3 2.029962e-05  4.261569
## 4 4.421444e-05 -4.084276
## 5 7.913247e-05 -3.947012
## 6 1.070274e-04  3.874082
```

```
##    lipid.species oddsratio      se               class.name    t.p.value
## 1        PS 36:4      2.09 0.14212      Phosphatidylserines 4.071778e-07
## 2        PS 38:5      1.77 0.13395      Phosphatidylserines 2.752356e-05
## 3        PC 34:6      1.79 0.13662     Phosphatidylcholines 2.764683e-05
## 4       LPC 14:0      0.57 0.13763 Lysophosphatidylcholines 5.755634e-05
## 5       LPC 22:6      0.58 0.13801 Lysophosphatidylcholines 9.982223e-05
## 6        PC 34:2      1.60 0.12132     Phosphatidylcholines 1.329315e-04
## 7        PC 36:2      1.58 0.12987     Phosphatidylcholines 4.985395e-04
## 8        PS 38:6      1.56 0.12846      Phosphatidylserines 6.194329e-04
## 9        PS 38:4      1.56 0.13168      Phosphatidylserines 8.350656e-04
## 10       PS 40:6      1.51 0.12657      Phosphatidylserines 1.266936e-03
## 11      LPC 17:0      0.66 0.13040 Lysophosphatidylcholines 1.600832e-03
## 12       PC 42:1      1.50 0.12885     Phosphatidylcholines 1.826160e-03
## 13       PC 34:0      1.50 0.12899     Phosphatidylcholines 1.846870e-03
##    norm.p.value    zvalue
## 1  2.138086e-07  5.186913
## 2  2.020331e-05  4.262632
## 3  2.029962e-05  4.261569
## 4  4.421444e-05 -4.084276
## 5  7.913247e-05 -3.947012
## 6  1.070274e-04  3.874082
## 7  4.280217e-04  3.522175
## 8  5.368397e-04  3.461668
## 9  7.327607e-04  3.377019
## 10 1.130009e-03  3.255982
## 11 1.440214e-03 -3.186468
## 12 1.650681e-03  3.146799
## 13 1.670066e-03  3.143384
```

#Problem 02 ##02-(a)

```
set.seed(1)
inTrain = createDataPartition(wdbc2$id, p = 0.7)#randomly split 70% data into training set
wdbc2Train = wdbc2[inTrain$Resample1,-1]#trim off the id column
wdbc2Test = wdbc2[-inTrain$Resample1,-1]

x_train = model.matrix(diagnosis ~ ., wdbc2Train)[,-1] # trim off the intercept column leaving only the
y_train = as.numeric(wdbc2Train$diagnosis)

x_test = model.matrix(diagnosis ~ ., wdbc2Test)[,-1]
```

```
y_test = as.numeric(wdbc2Test$diagnosis)

grid = 10 ^ seq(10, -2, length = 100)#a sequence of lambda

cv.out.ridge = cv.glmnet(
  x_train,
  y_train,
  alpha = 0,
  lambda = grid,
  family  = "binomial",
  type.measure = "auc"
) # Fit ridge regression model on training data

cv.out.lasso = cv.glmnet(
  x_train,
  y_train,
  alpha = 1,
  lambda = grid,
  family  = "binomial",
  type.measure = "auc"
) # Fit lasso regression model on training data

cv.out.ridge$lambda.min;cv.out.lasso$lambda.min
```

```
## [1] 0.01321941
```

```
## [1] 0.01
```

##02-(b)

```
cv.auc.ridge = cv.out.ridge$cvm
auc.opt.ridge = cv.auc.ridge[which(cv.out.ridge$lambda == cv.out.ridge$lambda.min)]
auc.1se.ridge = cv.auc.ridge[which(cv.out.ridge$lambda == cv.out.ridge$lambda.1se)]

cv.auc.lasso = cv.out.lasso$cvm
auc.opt.lasso = cv.auc.lasso[which(cv.out.lasso$lambda == cv.out.lasso$lambda.min)]
auc.1se.lasso = cv.auc.lasso[which(cv.out.lasso$lambda == cv.out.lasso$lambda.1se)]

auc.df = data.frame(
  ridge.auc = c(auc.opt.ridge, auc.1se.ridge),
  lasso.auc = c(auc.opt.lasso, auc.1se.lasso),
  row.names = c("optimal.lambda", "1se.lambda")
)
auc.df
```

```
##                ridge.auc lasso.auc
## optimal.lambda 0.9856431 0.9809833
## 1se.lambda     0.9808945 0.9730718
```

##02-(c) Comments on results:model size of ridge is 30 but lasso is 8.This means lasso penalty more than ridge in this problem.Because ridge regression can't zero out coefficients; thus, it either end up including all the coefficients in the model, or none of them.In this problems, ridge model including all coefficeints. In contrast, the lasso does both parameter shrinkage and variable selection automatically.

```
sum.dt = data.frame(
  ridge.modelsize = c(cv.out.ridge$nzero[which(cv.out.lasso$lambda == cv.out.lasso$lambda.min)],
```

```
                      cv.out.ridge$nzero[which(cv.out.lasso$lambda == cv.out.lasso$lambda.1se)]),
  lasso.modelsize = c(cv.out.lasso$nzero[which(cv.out.lasso$lambda == cv.out.lasso$lambda.min)],
                      cv.out.lasso$nzero[which(cv.out.lasso$lambda == cv.out.lasso$lambda.1se)]),
  ridge.auc = c(auc.opt.ridge, auc.1se.ridge),
  lasso.auc =  c(auc.opt.lasso, auc.1se.lasso),
  row.names = c("optimal.lambda", "1se.lambda")
)
signif(sum.dt, 3)
```

```
##                ridge.modelsize lasso.modelsize ridge.auc lasso.auc
## optimal.lambda              30               8     0.986     0.981
## 1se.lambda                  30               4     0.981     0.973
```

##02-(d)

```
basemodel = glm(diagnosis ~ .,
                data = na.omit(wdbc2Train),
                family = binomial(link = 'logit'))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
modelB = stepAIC(basemodel,
                 direction = "back",
                 scale = T,
                 trace = F)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
coef.modelb = as.data.frame(coef(summary(modelB)))
signif.coef.modelb = coef.modelb[which(coef.modelb$`Pr(>|z|)` <= 0.05), ]#extract significant coefficie
signif.coef.modelb = signif.coef.modelb[sort(abs(signif.coef.modelb$Estimate),
                                             index.return = TRUE,
                                             decreasing = TRUE)$ix, ]
signif.coef.modelb
```

```
##                              Estimate   Std. Error   z value     Pr(>|z|)
## fractaldimension.stderr -1.469257e+03 6.414951e+02 -2.290363 2.200026e-02
## smoothness.stderr       -7.761337e+02 3.566880e+02 -2.175946 2.955932e-02
## fractaldimension.worst   1.815363e+02 8.457238e+01  2.146520 3.183154e-02
## concavepoints            1.335391e+02 5.766626e+01  2.315724 2.057336e-02
## smoothness.worst         1.159578e+02 4.605682e+01  2.517712 1.181199e-02
## compactness             -8.438774e+01 3.311870e+01 -2.548039 1.083305e-02
## (Intercept)             -6.750136e+01 1.585275e+01 -4.258023 2.062429e-05
## radius.stderr            2.369362e+01 5.960315e+00  3.975230 7.031117e-05
## concavity.worst          1.861314e+01 5.309572e+00  3.505583 4.556089e-04
## perimeter                8.628307e-01 3.600685e-01  2.396296 1.656171e-02
## texture.worst            3.324306e-01 8.535870e-02  3.894514 9.839598e-05
## perimeter.worst          3.294255e-01 1.428654e-01  2.305845 2.111929e-02
```

```
## area.worst              -2.772031e-02 7.114482e-03 -3.896322 9.766452e-05
```

##02-(e) These variables entered the model and was later on discarded: perimeter.worst, texture, area.stderr and area.

```
basenull = glm(diagnosis ~ 1, data = wdbc2Train, family = binomial)
modelS =  stepAIC(basenull,
                  scope = list(upper = basemodel),
                  direction = "both",
                  scale = T,
                  trace = F)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(modelS)
```

```
##
## Call:
## glm(formula = diagnosis ~ smoothness.worst + radius.stderr +
##     smoothness.stderr + perimeter + texture.worst + radius +
##     compactness + concavity + radius.worst + area.worst, family = binomial,
##     data = wdbc2Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6239  -0.0652  -0.0051   0.0129   3.4752
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -55.00405   10.34290  -5.318 1.05e-07 ***
## smoothness.worst   138.53123   34.77308   3.984 6.78e-05 ***
## radius.stderr       14.82743    4.06251   3.650 0.000262 ***
## smoothness.stderr -919.82381  332.92996  -2.763 0.005731 **
## perimeter            0.91899    0.67827   1.355 0.175450
## texture.worst        0.25093    0.06746   3.720 0.000199 ***
## radius              -5.87181    4.58093  -1.282 0.199914
## compactness        -82.25193   29.72351  -2.767 0.005653 **
## concavity           47.55444   13.60745   3.495 0.000475 ***
## radius.worst         3.49211    1.10408   3.163 0.001562 **
## area.worst          -0.02865    0.00666  -4.302 1.70e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 530.189  on 400  degrees of freedom
## Residual deviance:  75.925  on 390  degrees of freedom
## AIC: 97.925
##
## Number of Fisher Scoring iterations: 9
```

```
coef.models = as.data.frame(coef(summary(modelS)))
signif.coef.models = coef.models[which(coef.models$`Pr(>|z|)`<=0.05),]
signif.coef.models = signif.coef.models[sort(abs(signif.coef.models$Estimate),index.return=TRUE,decreas
signif.coef.models
```

```
##                      Estimate   Std. Error   z value     Pr(>|z|)
```

```
## smoothness.stderr -919.8238107 3.329300e+02 -2.762815 5.730529e-03
## smoothness.worst   138.5312276 3.477308e+01  3.983864 6.780372e-05
## compactness        -82.2519273 2.972351e+01 -2.767234 5.653408e-03
## (Intercept)        -55.0040473 1.034290e+01 -5.318051 1.048845e-07
## concavity           47.5544388 1.360745e+01  3.494735 4.745324e-04
## radius.stderr       14.8274297 4.062509e+00  3.649821 2.624235e-04
## radius.worst         3.4921051 1.104079e+00  3.162913 1.561991e-03
## texture.worst        0.2509271 6.745978e-02  3.719654 1.994956e-04
## area.worst          -0.0286493 6.660054e-03 -4.301662 1.695218e-05
```

##02-(f) Employ AIC standard to choose model, model S performs better with a smaller AICAIC of model B is 103.6, AIC of model S is 97.9

```
modelB$aic;modelS$aic
```

```
## [1] 103.554
```

```
## [1] 97.9253
```

##02-(g) Training auc of model B is 0.995,Training auc of model S is 0.994

```
modelB.pred = predict(modelB, newdata = as.data.frame(x_train))
modelS.pred = predict(modelS, newdata = as.data.frame(x_train))

roc_B <- roc(y_train, modelB.pred)
```

```
## Setting levels: control = 1, case = 2
```

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

```
train.aucB = auc(roc_B)
roc_S <- roc(y_train, modelS.pred)
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
train.aucS = auc(roc_S)
train.aucB;train.aucS
```

```
## Area under the curve: 0.9951
```

```
## Area under the curve: 0.9938
```

##02-(h) All auc is over 0.9, this may indicates that they're all slightly overfiting. According to test auc,Ridge regression model has the best test performance, Lasso also performs well.Comparing trainig auc and test auc of Ridge and Lasso, there's no significant difference between them. ModelS performs the best in traning auc, however, ModelS's test auc is lower than Ridge and Lasso, which implies overfitting in training. According to ROC plot , test fitting performances: Ridge is better than Lasso,Lasso is better than Backward>Stepwise

```
pred.ridge = predict(cv.out.ridge,
                     newx = x_test,
                     type = "response",
                     s = cv.out.ridge$lambda.1se)
pred.lasso = predict(cv.out.lasso,
                     newx = x_test,
                     type = "response",
                     s = cv.out.lasso$lambda.1se)
pred.modelb = predict(modelB, newdata = as.data.frame(x_test))
pred.models = predict(modelS, newdata = as.data.frame(x_test))
```

```
roc_rid <- roc(y_test, pred.ridge)
```

```
## Setting levels: control = 1, case = 2
```

```
## Warning in roc.default(y_test, pred.ridge): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

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

```
roc_lasso <- roc(y_test, pred.lasso)
```

```
## Setting levels: control = 1, case = 2
```

```
## Warning in roc.default(y_test, pred.lasso): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

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

```
roc_B <- roc(y_test, pred.modelb)
```

```
## Setting levels: control = 1, case = 2
```

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

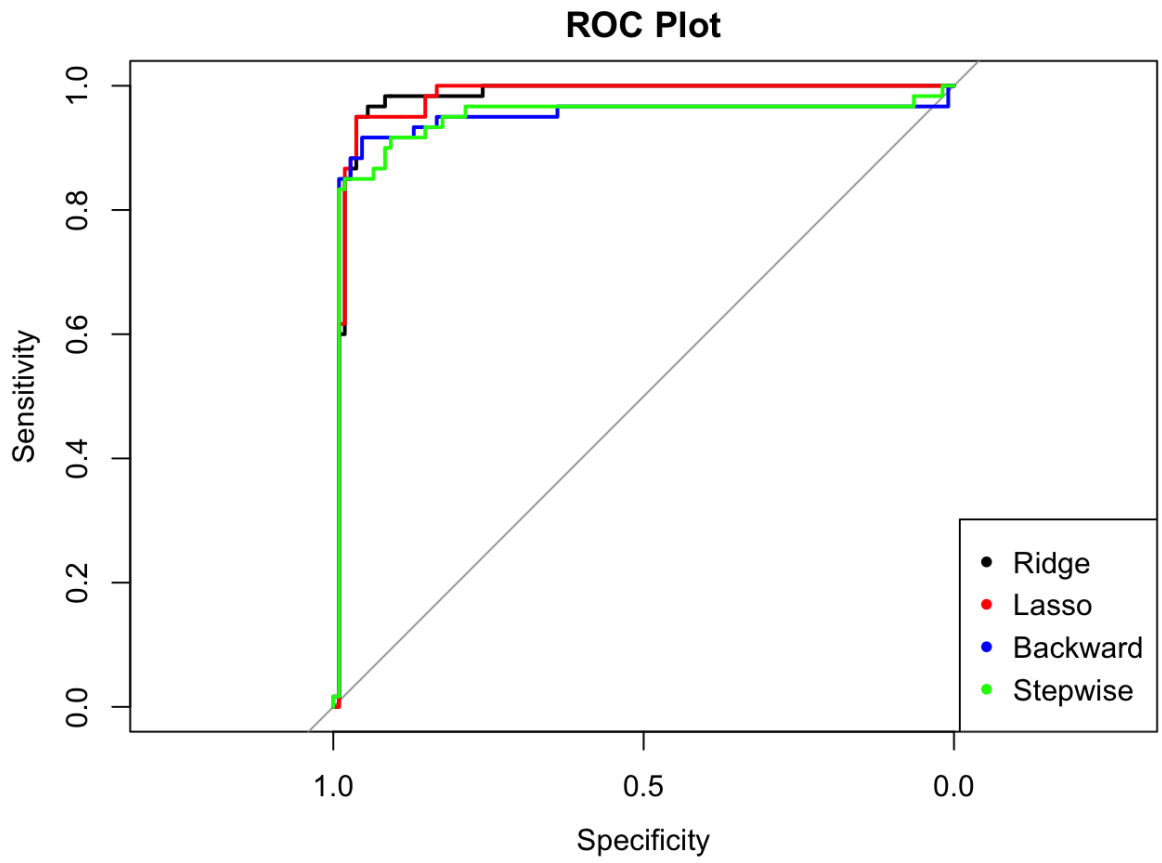
```
roc_S <- roc(y_test, pred.models)
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
test.aucR = auc(roc_rid)
test.aucL = auc(roc_lasso)
test.aucB = auc(roc_B)
test.aucS = auc(roc_S)

test.train.auc = data.frame(
  test.auc = c(test.aucR,test.aucL,test.aucB,test.aucS),
  train.auc = c(auc.opt.ridge,auc.opt.lasso,train.aucB,train.aucS),
  row.names = c("Ridge","Lasso","Backward","Stepwise"))
test.train.auc
```

```
##           test.auc train.auc
## Ridge    0.9799383 0.9856431
## Lasso    0.9790123 0.9809833
## Backward 0.9458333 0.9951394
## Stepwise 0.9458333 0.9937583
```

```
{plot(roc_rid,main = "ROC Plot")
lines(roc_lasso,col = "red")
lines(roc_B,col = "blue")
lines(roc_S,col = "green")
legend('bottomright',legend = c("Ridge","Lasso","Backward","Stepwise"),col = c("black","red","blue","gr
```



#Problem 03 ##03-(a)

```
snp.allele.dt = gdm.dt[, -c(1:3)]

#mean imputation
for (i in 4:179) {
  gdm.dt[, i][is.na(gdm.dt[, i])] <-
    mean(gdm.dt[, i], na.rm = TRUE)
}
```

##03-(b)

```
univ.glm.test <- function(x, y, order = FALSE) {
  SNP.names = as.character()
  reg.coeff = as.numeric()
  odds = as.numeric()
  se = as.numeric()
  pvalue = as.numeric()

  for (j in 1:ncol(x)) {
    model = glm(y ~ x[, j], family = binomial(link = "logit"))
    SNP.names[j] = colnames(x)[j]
    reg.coeff[j] = model$coefficients[2]
    odds[j] = exp(model$coefficients[2])
    se[j] = coef(summary(model))[2, 2]
    pvalue[j] = coef(summary(model))[2, 4]
  }
```

```
  snp.logis.sum.dt = data.table(SNP.names, reg.coeff, odds, se, pvalue)

  if (order == T) {
    snp.logis.sum.dt = arrange(snp.logis.sum.dt, pvalue)
  }

  return(snp.logis.sum.dt)
}
```

##03-(c) select significant SNPs out, then extract the maximum and minimum coefficients from those significant snps to obtain summarized statistics and confidence interval.

```
gwas = univ.glm.test(x = gdm.dt[,-c(1:3)], y = gdm.dt$pheno, order = T)

signif.snp = gwas[which(gwas$pvalue <= 0.05), ]#extract significant SNPs out
max.risk = signif.snp[which(signif.snp$reg.coeff == max(signif.snp$reg.coeff)), ]
min.risk = signif.snp[which(signif.snp$reg.coeff == min(signif.snp$reg.coeff)), ]

max.int.95 = max.risk$reg.coeff+1.96 * c( - max.risk$se, max.risk$se)
min.int.99 = max.risk$reg.coeff+2.58 * c( - min.risk$se, min.risk$se)
min.int.95 = min.risk$reg.coeff+1.96 * c( - min.risk$se, min.risk$se)
max.int.99 = max.risk$reg.coeff+2.58 * c( - max.risk$se, max.risk$se)

summary.gwas = summary(gwas)
sum.ci.gwas = data.frame(
  row.names  = c(max.risk$SNP.names,"", min.risk$SNP.names,"."),
  odds.interval.95 = exp(c(max.int.95, min.int.95)),
  odds.interval.99 = exp(c(max.int.99, min.int.99)))
summary.gwas;sum.ci.gwas
```

```
##   SNP.names           reg.coeff             odds              se
##  Length:176         Min.   :-0.61121   Min.   :0.5427   Min.   :0.09651
##  Class :character   1st Qu.:-0.09681   1st Qu.:0.9077   1st Qu.:0.10545
##  Mode  :character   Median : 0.01990   Median :1.0201   Median :0.11742
##                     Mean   : 0.02067   Mean   :1.0407   Mean   :0.13714
##                     3rd Qu.: 0.12990   3rd Qu.:1.1387   3rd Qu.:0.14238
##                     Max.   : 0.65655   Max.   :1.9281   Max.   :0.37619
##      pvalue
##  Min.   :0.0000427
##  1st Qu.:0.1509309
##  Median :0.3426575
##  Mean   :0.4086077
##  3rd Qu.:0.6450566
##  Max.   :0.9955224
```

```
##             odds.interval.95 odds.interval.99
## rs1423096_T        1.0358472        0.8510165
##                    3.5890279        4.3685224
## rs2237897_T        0.5144129        1.4415783
## .                  0.8002147        2.5788989
```

##03-(d)

```
gwas[, c("snp", "allele")] = str_split_fixed(gwas$SNP.names, "_", 2)
gwas.annot = merge(gwas, gdm.annot,all.x = T)#merge gwas with gdm.annot by snp names
```

```
hit.snp = gwas.annot[which(gwas.annot$pvalue < 10 ^ -4), ][, c(2, 7, 8, 10)]

#genes with 1Mb window
gene.1Mb.range1 = c(gwas.annot[28,]$pos-1e06,gwas.annot[28,]$pos+1e06)#28 is the row number of the firs
gene.1Mb.range2 = c(gwas.annot[76,]$pos-1e06,gwas.annot[76,]$pos+1e06)#76 is the row number of the seco
gene.1Mb.hit1 = gwas.annot[which(gwas.annot$pos >=gene.1Mb.range1[1]&gwas.annot$pos <=gene.1Mb.range1[2
gene.1Mb.hit2 = gwas.annot[which(gwas.annot$pos >=gene.1Mb.range2[1]&gwas.annot$pos <=gene.1Mb.range2[2
as.data.frame(gene.1Mb.hit1$gene);as.data.frame(gene.1Mb.hit2$gene)
```

```
##   gene.1Mb.hit1$gene
## 1             TCF7L2
## 2             TCF7L2
## 3             TCF7L2
## 4             TCF7L2
```

```
##   gene.1Mb.hit2$gene
## 1                 TH
## 2              KCNQ1
## 3           CACNA2D4
## 4              KCNQ1
## 5              KCNQ1
## 6              KCNQ1
## 7               SMG6
## 8               SMG6
```

##03-(e)

```
gdm.dt = as.data.table(gdm.dt)
gwas = as.data.table(gwas)
gwas.annot = as.data.table(gwas.annot)

snp.p4 = gwas[which(gwas$pvalue <= 1e-4), ]
snp.p3 = gwas[which(gwas$pvalue <= 1e-3), ]
snp.fto = gwas.annot[which(gwas.annot$gene == "FTO"),]

gdm.p4 <- gdm.dt[, .SD, .SDcols = gwas[pvalue < 1e-4]$SNP.names]
gdm.p3 <- gdm.dt[, .SD, .SDcols = gwas[pvalue < 1e-3]$SNP.names]
gdm.fto <- gdm.dt[, .SD, .SDcols = gwas.annot[gene == "FTO"]$SNP.names]

score.p4 = as.matrix(gdm.p4) %*% snp.p4$reg.coeff
score.p3 = as.matrix(gdm.p3) %*% snp.p3$reg.coeff
score.fto = as.matrix(gdm.fto) %*% snp.fto$reg.coeff

gdm.dt[, "score1"] = score.p4
gdm.dt[, "score2"] = score.p3
gdm.dt[, "score3"] = score.fto

modelp4.score1 = glm(
  pheno~score1,
  data = gdm.dt,
  family = binomial,
  na.action = na.exclude
)
modelp3.score2 = glm(
  pheno~score2,
```

```
  data = gdm.dt,
  family = binomial,
  na.action = na.exclude
)
modelfto.score3 = glm(
  pheno~score3,
  data = gdm.dt,
  family = binomial,
  na.action = na.exclude
)

sum.mp4 = coef(summary(modelp4.score1))
sum.mp3 = round(coef(summary(modelp3.score2)),2)
sum.mfto = round(coef(summary(modelfto.score3)),2)

score.fit.sum = data.frame(
  oddratio = round(exp(c(sum.mp4[2, 1], sum.mp3[2, 1], sum.mfto[2, 1])),2),
  CI.95 = c(
    paste("(", round(sum.mp4[2, 1] - 1.96 * sum.mp4[2, 2],2), round(sum.mp4[2, 1] + 1.96 *
            sum.mp4[2, 2],2), ")"),
    paste("(", round(sum.mp3[2, 1] - 1.96 * sum.mp3[2, 2],2), round(sum.mp3[2, 1] + 1.96 *
            sum.mp3[2, 2],2), ")"),
    paste("(", round(sum.mfto[2, 1] - 1.96 * sum.mfto[2, 2],2), round(sum.mfto[2, 1] + 1.96 *
            sum.mfto[2, 2],2), ")")
  ),
  pvalue = c(sum.mp4[2, 4], sum.mp3[2, 4], sum.mfto[2, 4]),
  row.names = c("p<e-04","p<e-03","FTO")
)
score.fit.sum
```

```
##        oddratio          CI.95       pvalue
## p<e-04     2.72  ( 0.65 1.35 ) 2.342444e-08
## p<e-03     1.45  ( 0.25 0.49 ) 0.000000e+00
## FTO        1.40 ( -0.21 0.89 ) 2.300000e-01
```

##03-(f)

```
gwas.test = univ.glm.test(x=gdm.test[,-c(1:3)],y = gdm.test$pheno,order = T)
gwas.annot.test = merge(gwas.test, gdm.annot,by.x = "SNP.names",by.y = "snp")#merge gwas with gdm.annot

gwas.test = as.data.table(gwas.test)
gdm.test = as.data.table(gdm.test)
gwas.annot.test = as.data.table(gwas.annot.test)

gdm.p4.test <- gdm.test[, .SD, .SDcols = gwas[pvalue < 1e-4]$snp]
gdm.p3.test <- gdm.test[, .SD, .SDcols = gwas[pvalue < 1e-3]$snp]
gdm.fto.test <- gdm.test[, .SD, .SDcols = gwas.annot[gene == "FTO"]$snp]

score.p4.test = as.matrix(gdm.p4.test) %*% snp.p4$reg.coeff
score.p3.test = as.matrix(gdm.p3.test) %*% snp.p3$reg.coeff
score.fto.test = as.matrix(gdm.fto.test) %*% snp.fto$reg.coeff

gdm.test[, "score1"] = score.p4.test
gdm.test[, "score2"] = score.p3.test
gdm.test[, "score3"] = score.fto.test
```

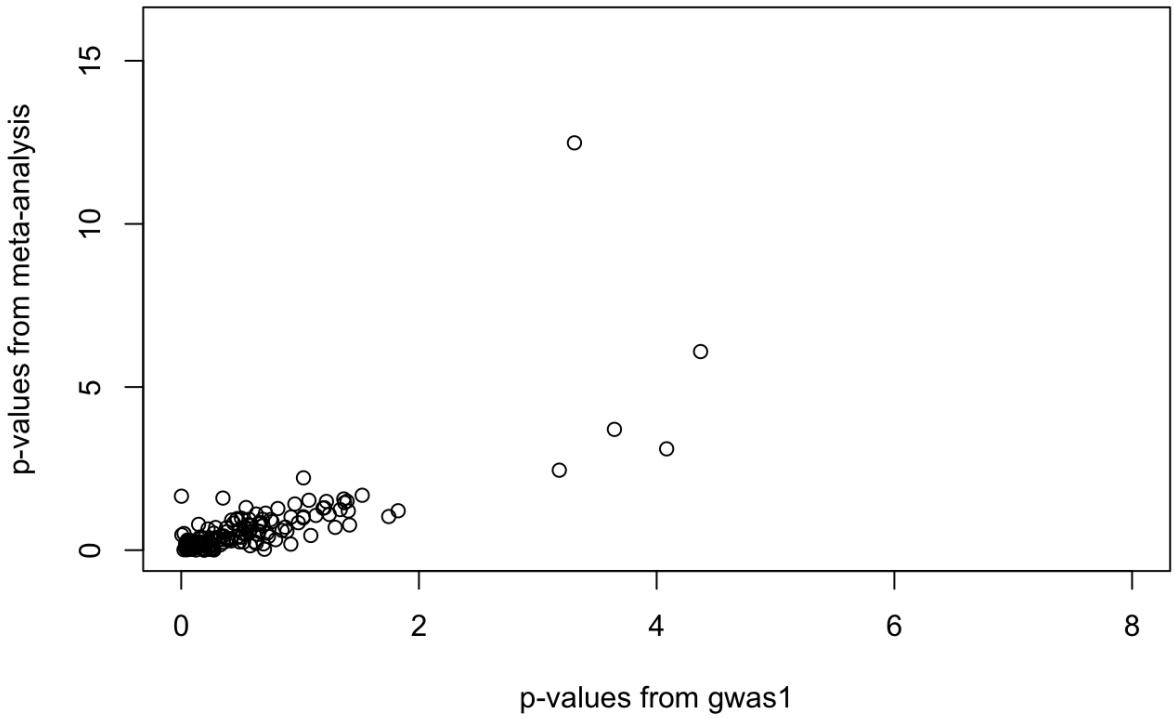
##03-(g) The test log-likelihood for the predicted probabilities from the three genetic risk score models.

```
#predicted outcomes from models fitted at point (e)
pred.p4 = predict(modelp4.score1, newdata = gdm.test)
pred.p3 = predict(modelp3.score2, newdata = gdm.test)
pred.fto = predict(modelfto.score3, newdata = gdm.test)

#define a function to calculate test log likelihood
test.loglik <- function(s_model,pred,score) {
    log_lik <- sum(pred*s_model$coefficients[2]%*%score-log(1+exp(s_model$coefficients[2]*score)))
    return(log_lik)
}

# Compute the test log-likelihood for the predicted probabilities from the three genetic risk score mod
log.likeli.p4 = test.loglik(modelp4.score1,pred.p4,gdm.test$score1)
log.likeli.p3 = test.loglik(modelp3.score2,pred.p3,gdm.test$score2)
log.likeli.fto = test.loglik(modelfto.score3,pred.fto,gdm.test$score3)

loglik = data.frame(
  loglikeli = c(log.likeli.p4,log.likeli.p3,log.likeli.fto),
  row.names = c("p<e04","p<e03","FTO")
)
loglik
```

```
##       loglikeli
## p<e04 -17.75933
## p<e03 -19.81281
## FTO   -28.50983
```

##03-(h)meta-analysis Because gwas in (c) has only one ellele,therefore only check whether snp and effect allele is corresponded or not in these two gwas results.

```
#Combine snp and effect allele in gdm.study2 into a new column "Snp.names"
gdm.study2[,"Snp.names"]<-paste(gdm.study2$snp,gdm.study2$effect.allele,sep = "_")


#check snp and effect allele is corresponded or not
gwas1= gwas[SNP.names %in% gdm.study2$Snp.names]
gwas2 = gdm.study2[Snp.names %in% gwas$SNP.names]

beta1 <- gwas1$reg.coeff
beta2 <- gwas2$beta


weight.gwas1 <- 1 / gwas1$se^2
weight.gwas2 <- 1 / gwas2$se^2

beta.ma <- (weight.gwas1 * beta1 + weight.gwas2 * beta2) / (weight.gwas1 + weight.gwas2)
se.ma <- sqrt(1 / (weight.gwas1 + weight.gwas2))

pval.ma <- 2 * pnorm(abs(beta.ma / se.ma), lower.tail=FALSE)
plot(-log10(gwas1$pvalue), -log10(pval.ma), xlim=c(0, 8), ylim=c(0, 16), xlab="p-values from gwas1", yla
```



#Problem 04 ##04-(a)

```
cor.matrix = cor(nki[, -(1:6)])

flattenCorrMatrix <- function(cormat) {
  ut = upper.tri(cormat)
  data.frame(
    variable1 = rownames(cormat)[row(cormat)[ut]],
    variable2 = rownames(cormat)[col(cormat)[ut]],
    cor = (cormat)[ut]
  )
}

nki.pair.df = flattenCorrMatrix(cor.matrix)
nki.pair.df[which(abs(nki.pair.df$cor) > 0.8), ]
```

```
##      variable1 variable2       cor
## 31      DIAPH3  DIAPH3.1 0.8031368
## 58      DIAPH3  DIAPH3.2 0.8338591
## 64    DIAPH3.1  DIAPH3.2 0.8868741
## 1117      PECI    PECI.1 0.8697836
## 1768    IGFBP5  IGFBP5.1 0.9775030
## 1957    NUSAP1      PRC1 0.8298356
## 2144      PRC1     CENPA 0.8175424
```

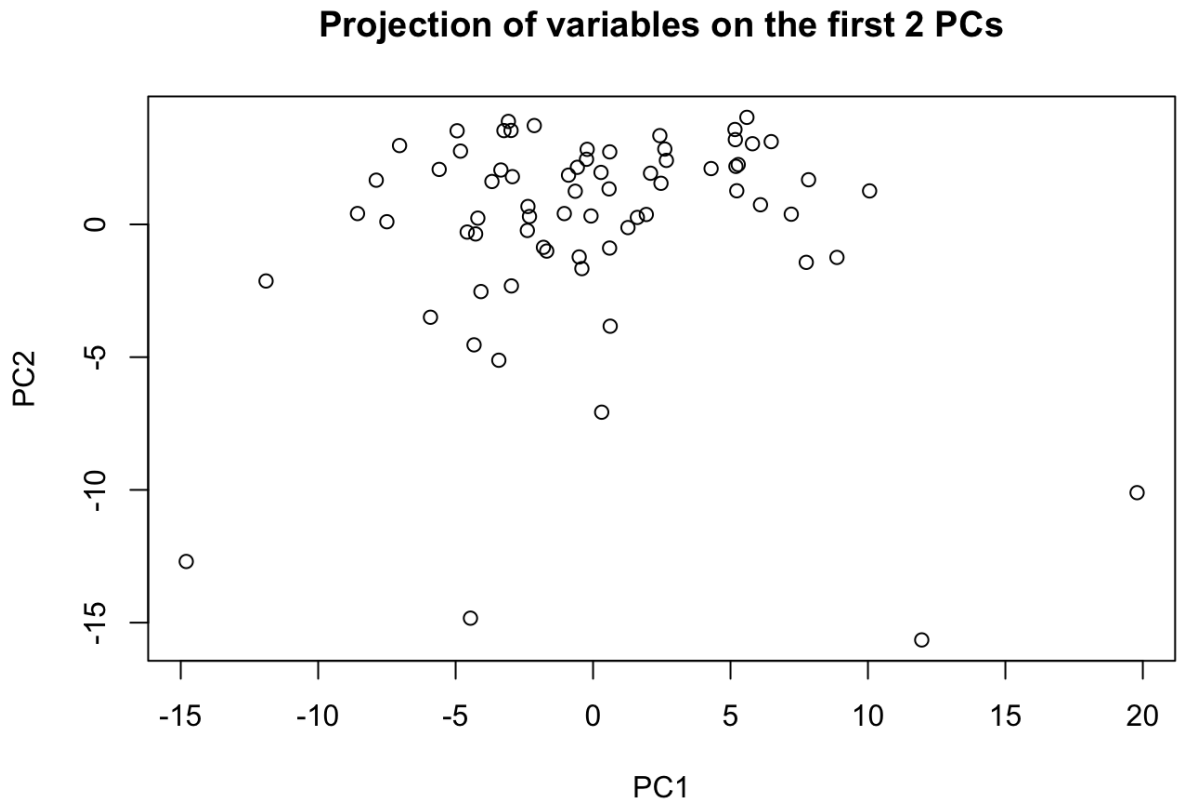
##04-(b) The percentage of variance explained by the first two components is 33.16%. Rule of iden-

tify 4 genes: According to scatter plot of PC1 vs PC2, I found there're 4 "outliers" around the bottom,therefore I ordered PC2 in increasing order, and exrtact the first 4 gene as the most different group,they are ZNF533,MMP9,CDCA7,SCUBE2.

```
nki.pca = prcomp(t(nki[, -(1:6)]),
                 center = TRUE,
                 scale  = TRUE)
summary(nki.pca)
```

```
## Importance of components:
##                            PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     5.6053 4.0415 3.10751 2.96830 2.33569 2.27737 2.21892
## Proportion of Variance 0.2182 0.1134 0.06706 0.06119 0.03789 0.03602 0.03419
## Cumulative Proportion  0.2182 0.3316 0.39868 0.45986 0.49775 0.53376 0.56796
##                            PC8     PC9    PC10    PC11   PC12   PC13    PC14
## Standard deviation     2.18096 1.95856 1.88658 1.82950 1.7637 1.6928 1.63315
## Proportion of Variance 0.03303 0.02664 0.02472 0.02324 0.0216 0.0199 0.01852
## Cumulative Proportion  0.60099 0.62763 0.65234 0.67559 0.6972 0.7171 0.73561
##                           PC15    PC16   PC17    PC18    PC19    PC20    PC21
## Standard deviation     1.5599 1.49993 1.4551 1.42807 1.38852 1.32131 1.30382
## Proportion of Variance 0.0169 0.01562 0.0147 0.01416 0.01339 0.01212 0.01181
## Cumulative Proportion  0.7525 0.76813 0.7828 0.79700 0.81039 0.82251 0.83431
##                           PC22    PC23    PC24    PC25   PC26   PC27    PC28
## Standard deviation     1.22059 1.20118 1.15058 1.11382 1.0864 1.0528 1.03360
## Proportion of Variance 0.01035 0.01002 0.00919 0.00862 0.0082 0.0077 0.00742
## Cumulative Proportion  0.84466 0.85468 0.86387 0.87249 0.8807 0.8884 0.89580
##                           PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     1.00050 0.95764 0.92774 0.88901 0.85398 0.83937 0.83031
## Proportion of Variance 0.00695 0.00637 0.00598 0.00549 0.00506 0.00489 0.00479
## Cumulative Proportion  0.90275 0.90912 0.91510 0.92059 0.92565 0.93055 0.93533
##                           PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.8135 0.79396 0.78385 0.76291 0.73847 0.72343 0.68790
## Proportion of Variance 0.0046 0.00438 0.00427 0.00404 0.00379 0.00363 0.00329
## Cumulative Proportion  0.9399 0.94431 0.94857 0.95261 0.95640 0.96004 0.96332
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.65318 0.63742 0.62923 0.57779 0.56366 0.54771 0.54506
## Proportion of Variance 0.00296 0.00282 0.00275 0.00232 0.00221 0.00208 0.00206
## Cumulative Proportion  0.96629 0.96911 0.97186 0.97417 0.97638 0.97846 0.98053
##                           PC50   PC51    PC52    PC53    PC54    PC55    PC56
## Standard deviation     0.52697 0.5097 0.49822 0.49047 0.47105 0.42915 0.41866
## Proportion of Variance 0.00193 0.0018 0.00172 0.00167 0.00154 0.00128 0.00122
## Cumulative Proportion  0.98246 0.9843 0.98598 0.98765 0.98920 0.99047 0.99169
##                           PC57    PC58    PC59    PC60    PC61    PC62    PC63
## Standard deviation     0.40366 0.38706 0.37110 0.34104 0.33078 0.32220 0.30170
## Proportion of Variance 0.00113 0.00104 0.00096 0.00081 0.00076 0.00072 0.00063
## Cumulative Proportion  0.99282 0.99386 0.99482 0.99563 0.99639 0.99711 0.99774
##                           PC64    PC65    PC66    PC67    PC68    PC69
## Standard deviation     0.26977 0.25759 0.24448 0.23374 0.19423 0.18478
## Proportion of Variance 0.00051 0.00046 0.00042 0.00038 0.00026 0.00024
## Cumulative Proportion  0.99825 0.99871 0.99912 0.99950 0.99976 1.00000
##                           PC70
## Standard deviation     2.489e-15
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

```r
#scatter plot of PC1 and PC2
plot(nki.pca$x[, 1:2], main = "Projection of variables on the first 2 PCs")
```




```r
#identify 4 genes
PC2 = nki.pca$x[,2]
sort(PC2)[1:4]
```

```
##    ZNF533      MMP9     CDCA7    SCUBE2
## -15.65053 -14.83138 -12.69578 -10.10422
```

##04-(c) I use correaltion to test 3 principle components and outcomes are associated or not. PC1 and PC2 are almost equally associate with unadjusted and adjusted model according to correlation. PC3 is more associated with unadjusted model (cor = 0.173), correlation between adjusted model outcomes and PC is 0.079.

```r
patient.nki.pca = prcomp(nki[, -(1:6)], scale. = T)

unadj.moel = glm(Event ~ ., data = nki[, -(2:6)], family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
adj.modle = glm(Event ~ ., data = nki, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
out.unadj = predict(unadj.moel)
out.adj = predict(adj.modle)

#Use cor to test whether principle components 1 to 3 is associated with models ourtcomes
cor.pc.adjout = NULL
cor.pc.unadjout = NULL
for (i in 1:3) {
  cor.pc.adjout[i] = cor(patient.nki.pca$x[, i], out.adj)
  cor.pc.unadjout[i] = cor(patient.nki.pca$x[, i], out.unadj)
}

cor.pc.outcome = data.frame(cor.pc.unadjout,
                            cor.pc.adjout,
                            row.names = c("PC1", "PC2", "PC3"))
cor.pc.outcome
```

```
##     cor.pc.unadjout cor.pc.adjout
## PC1      0.28356234    0.26852994
## PC2     -0.06563661   -0.05296604
## PC3      0.17257000    0.07930190
```

##04-(d) AUC of Model penalized all variables is 0.80 and auc of Model only penalize gene is 0.75,which is out of my expectation,beacuse modle with less covariates being penalized should have a larger auc.In this problem, model penalized all variables is probably overfitted. Besides, model penalized all variables has a smaller model size(35 for optimal lambda and 4 for 1se lambda) than model only penalize gene(55 for optimal lambda and 7 for 1se lambda).Generally, lasso might perform better in a situation where some of the predictors have large coefficients, in this problem, perhaps gene predicators in the second model have very small coefficients.

```
set.seed(1)
x_nki = prepareX(nki[,-1])#transform original design matrix to fit the glmnet object

model.all.lasso = cv.glmnet(
  x_nki,
  nki$Event,
  family = "binomial",
  alpha = 1,
  type.measure = "auc"
)

model.penal.lasso = cv.glmnet(
  x_nki,
  nki$Event,
  family = "binomial",
  alpha = 1,
  penalty.factor = c(rep(0, 7), rep(1, 70)),
  type.measure = "auc"
)#penalty factor = 0 repeat 7 times because 5 non-gene variables are transformed into 7 dummies in x_nk

model.all.lasso;model.penal.lasso
```

```
##
## Call:  cv.glmnet(x = x_nki, y = nki$Event, type.measure = "auc", family = "binomial",      alpha = 1
##
## Measure: AUC
```

```
##
##      Lambda Measure      SE Nonzero
## min 0.01974  0.7854 0.05022      35
## 1se 0.07969  0.7363 0.04714       4
```

```
##
## Call:  cv.glmnet(x = x_nki, y = nki$Event, type.measure = "auc", family = "binomial",      alpha = 1
##
## Measure: AUC
##
##      Lambda Measure      SE Nonzero
## min 0.00360  0.7466 0.04696      55
## 1se 0.08515  0.6997 0.05645       7
```

```
#calculate AUC
all.pred = predict(model.all.lasso,x_nki)
penal.pred = predict(model.penal.lasso,x_nki)
all.roc = roc(nki$Event,all.pred)
```

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

```
## Warning in roc.default(nki$Event, all.pred): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

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

```
penal.roc = roc(nki$Event,penal.pred)
```

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

```
## Warning in roc.default(nki$Event, penal.pred): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

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

```
all.auc = auc(all.roc)
penal.auc = auc(penal.roc)
all.auc;penal.auc
```

```
## Area under the curve: 0.8012
```

```
## Area under the curve: 0.7451
```