bortB_analysis_final

Zhaoliang Zhou

4/25/2022

lib

```
####################################################
library("ridge")
library("sva")
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-39. For overview type 'help("mgcv-package")'.
```

```
## Loading required package: genefilter
```

```
## Loading required package: BiocParallel
```

```
library("car")
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:genefilter':
##
##     Anova
```

```
library("preprocessCore")
library("ROCR")
library("GEOquery")
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
```

```
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Setting options('download.file.method.GEOquery'='auto')

## Setting options('GEOquery.inmemory.gpl'=FALSE)
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-3
```

```
library(coefplot)
```

```
## Loading required package: ggplot2
```

```
library(ROCR)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:coefplot':
##
##     extractPath
```

```
library(nnet)
```

```
##
## Attaching package: 'nnet'

## The following object is masked from 'package:mgcv':
##
##     multinom
```

```
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:coefplot':
##
##     coefplot

## The following object is masked from 'package:stats':
##
##     loadings
```

```
#####################################################
### code chunk number 4: bortezomib.Snw:31-36
#####################################################
scriptsDir <- "C:/Users/leonz/Desktop/UMN/Spring 2022/PubH 7475 - Statistical Learnign and Data mining/
```

```
source(file.path(scriptsDir, "compute_phenotype_function.R"))
source(file.path(scriptsDir, "summarizeGenesByMean.R"))
source(file.path(scriptsDir, "homogenize_data.R"))
source(file.path(scriptsDir, "do_variable_selection.R"))

source("C:/Users/leonz/Desktop/UMN/Spring 2022/PubH 7475 - Statistical Learnign and Data mining/final p
```

load data and functions

```
##################################################
### code chunk number 5: bortezomib.Snw:42-44
##################################################
load("C:/Users/leonz/Desktop/UMN/Spring 2022/PubH 7475 - Statistical Learnign and Data mining/final pro
# bortezomib_mas5 <- getGEO("GSE9782") # uncomment this line to download the data directly from GEO.
```

```
##################################################
### code chunk number 8: bortezomib.Snw:76-77
##################################################
load(file="C:/Users/leonz/Desktop/UMN/Spring 2022/PubH 7475 - Statistical Learnign and Data mining/final


##################################################
### code chunk number 9: bortezomib.Snw:84-90
##################################################
sensBortezomib <- read.csv("C:/Users/leonz/Desktop/UMN/Spring 2022/PubH 7475 - Statistical Learnign and
as.is=TRUE)
bortic50s <- sensBortezomib$"IC.50"
names(bortic50s) <- sensBortezomib$"Cell.Line.Name"
tissue <- sensBortezomib$"Tissue"
names(tissue) <- sensBortezomib$"Cell.Line.Name"
```

```
##################################################
### code chunk number 10: bortezomib.Snw:96-104
##################################################
pData <- read.delim("C:/Users/leonz/Desktop/UMN/Spring 2022/PubH 7475 - Statistical Learnign and Data mi
pDataUnique <- pData[pData$Source.Name %in% names(which(table(pData$Source.Name) ==
1)), ]
rownames(pDataUnique) <- pDataUnique$Source.Name
commonCellLines <- rownames(pDataUnique)[rownames(pDataUnique) %in% names(bortic50s)]
pDataUniqueOrd <- pDataUnique[commonCellLines, ]
bortic50sOrd <- bortic50s[commonCellLines]
trainDataOrd <- gdsc_brainarray_syms[, pDataUniqueOrd$"Array.Data.File"]


##################################################
### code chunk number 11: bortezomib.Snw:110-111
##################################################
print(sum(grep("myeloma", sensBortezomib$Tissue), ignore.case=TRUE))
```

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

```
exprDataU133b <- cbind(exprs(bortezomib_mas5[[3]]), exprs(bortezomib_mas5[[4]]))

bortIndex <- c(which(pData(phenoData(bortezomib_mas5[[1]]))[,"characteristics_ch1.1"] == "treatment = P
```

```
dexIndex <- c(which(pData(phenoData(bortezomib_mas5[[1]]))[,"characteristics_ch1.1"] == "treatment = Dex
studyIndex <- c(as.character(pData(bortezomib_mas5[[1]])[, "characteristics_ch1"]), as.character(pData(b

library("hgu133b.db")
```

```
## Loading required package: AnnotationDbi
## Loading required package: stats4
## Loading required package: IRanges
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:Matrix':
##
##     expand, unname
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:nlme':
##
##     collapse
## The following object is masked from 'package:grDevices':
##
##     windows
## Loading required package: org.Hs.eg.db
##
##
```

```
x <- hgu133bSYMBOL
mapped_probes <- mappedkeys(x) # Get the probe identifiers that are mapped to a gene
names(mapped_probes) <- as.character(x[mapped_probes])
affy2sym <- as.character(x[mapped_probes])
sym2affy <- affy2sym
names(sym2affy) <- names(affy2sym)
rownames(exprDataU133b) <- sym2affy[rownames(exprDataU133b)]

respU133b <- c(as.character(pData(bortezomib_mas5[[3]])[, "characteristics_ch1.8"]),
as.character(pData(bortezomib_mas5[[4]])[, "characteristics_ch1.8"]))[bortIndex]
```

paper ridge

```
predictedSensitivityU133b <- calcPhenotype(exprDataU133b[, bortIndex], trainDataOrd, bortic50sOrd,
selection=1)
```

```
## Warning in if (class(testExprData) != "matrix") stop("ERROR: \"testExprData\"
## must be a matrix."): the condition has length > 1 and only the first element
```

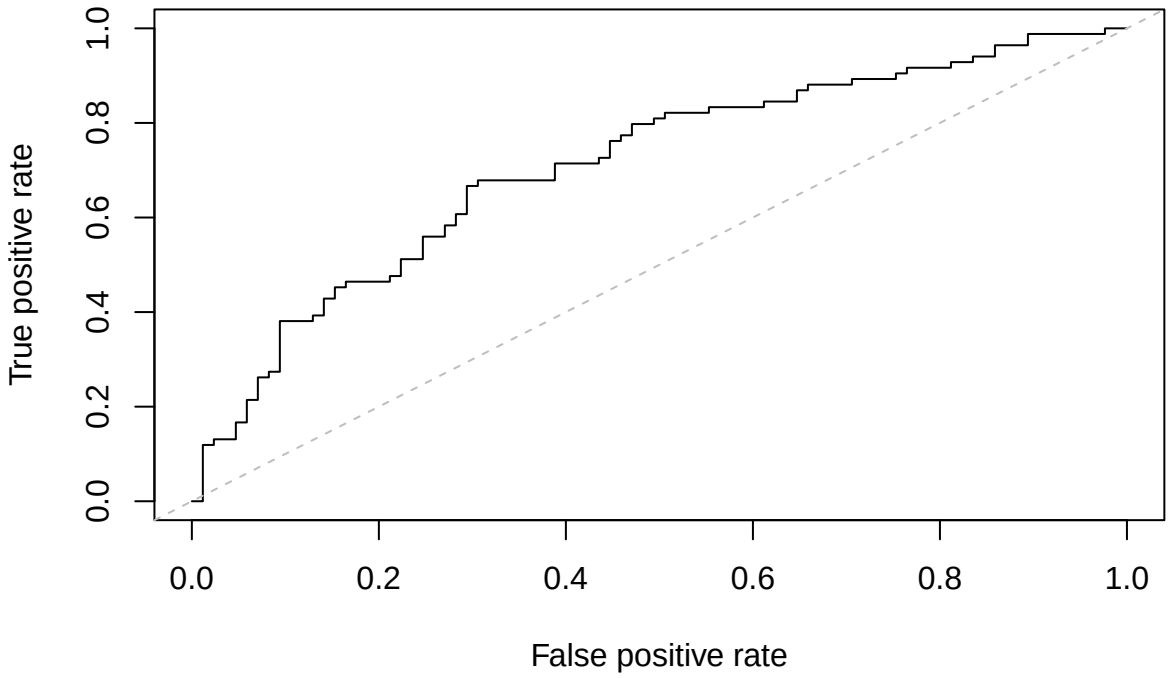
```
## will be used

## Warning in if (class(trainingExprData) != "matrix") stop("ERROR:
## \"trainingExprData\" must be a matrix."): the condition has length > 1 and only
## the first element will be used

##
##  3919  gene identifiers overlap between the supplied expression matrices...
##

## Found2batches

## Adjusting for0covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data

##
## Fitting Ridge Regression model... Done
##
## Calculating predicted phenotype...

## Warning in if (class(homData$test) == "numeric") {: the condition has length > 1
## and only the first element will be used

## Done
```

```
l_U133b <- list("Responder"=predictedSensitivityU133b[respU133b == "PGx_Responder = R"]
, "Non-responder"=predictedSensitivityU133b[respU133b == "PGx_Responder = NR"])
```

```
predB <- prediction(c(l_U133b[[1]], l_U133b[[2]]), c(rep("sens", length(l_U133b[[1]])),
rep("res", length(l_U133b[[2]]))),
label.ordering=c("sens", "res"))
perfB <- performance(predB, measure = "tpr", x.measure = "fpr")
print(paste("AUC:", performance(predB, measure = "auc")@"y.values"[[1]]))
```

```
## [1] "AUC: 0.710504201680672"
```

```
plot(perfB)
abline(0, 1, col="grey", lty=2)
```



ridge glmnet

```
predictedSensitivityU133b_ridge2 <- calcPhenotype_ridge_glmnet(exprDataU133b[, bortIndex], trainDataOrd
selection=1)
```

```
## Warning in if (class(testExprData) != "matrix") stop("ERROR: \"testExprData\"
## must be a matrix."): the condition has length > 1 and only the first element
## will be used
```

```
## Warning in if (class(trainingExprData) != "matrix") stop("ERROR:
## \"trainingExprData\" must be a matrix."): the condition has length > 1 and only
## the first element will be used
```

```
##
##  3919  gene identifiers overlap between the supplied expression matrices...
##
```

```
## Found2batches
```

```
## Adjusting for0covariate(s) or covariate level(s)
```

```
## Standardizing Data across genes
```

```
## Fitting L/S model and finding priors
```

```
## Finding parametric adjustments
```

```
## Adjusting the Data
```

```
##
## Fitting glmnet ridge Regression model... Done
##
```

```
## Calculating predicted phenotype...
## best lambda min is: 4.21746
```

```
## Warning in if (class(homData$test) == "numeric") {: the condition has length > 1
## and only the first element will be used
```

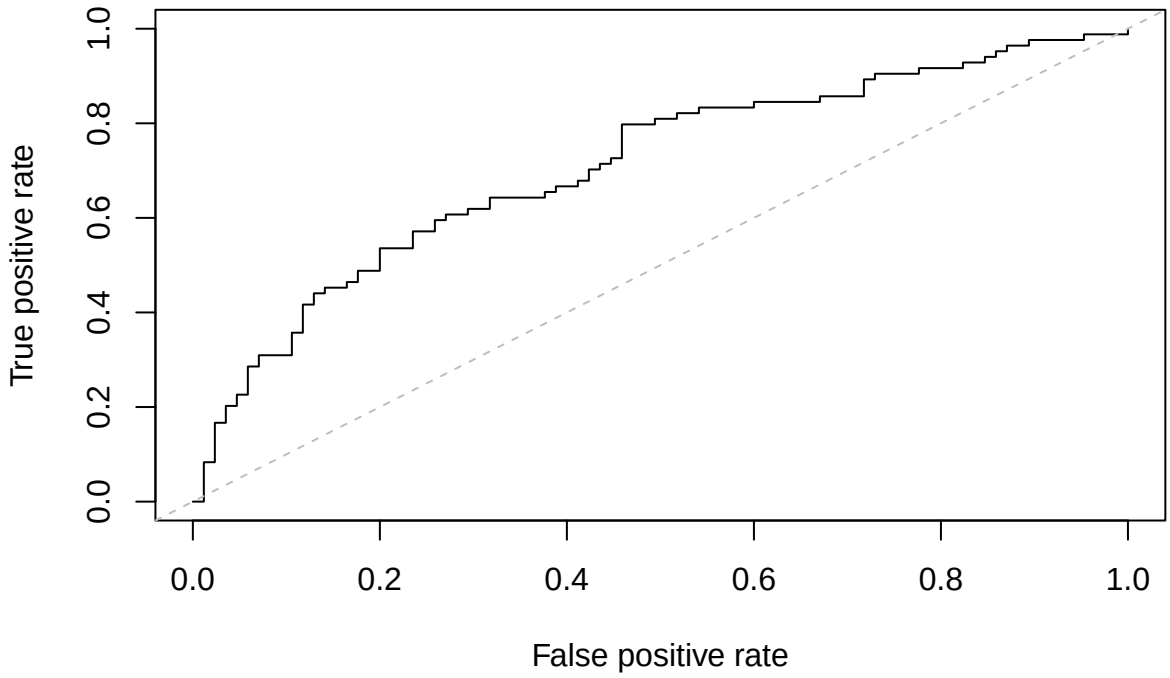
```
## Done
```

```
l_U133b_ridge2 <- list("Responder"=predictedSensitivityU133b_ridge2[respU133b == "PGx_Responder = R"]
, "Non-responder"=predictedSensitivityU133b_ridge2[respU133b == "PGx_Responder = NR"])
```

```
predB_ridge2 <- prediction(c(l_U133b_ridge2[[1]], l_U133b_ridge2[[2]]), c(rep("sens", length(l_U133b_ri
rep("res", length(l_U133b_ridge2[[2]]))),
label.ordering=c("sens", "res"))
perfB_ridge2 <- performance(predB_ridge2, measure = "tpr", x.measure = "fpr")
print(paste("AUC:", performance(predB_ridge2, measure = "auc")@"y.values"[[1]]))
```

```
## [1] "AUC: 0.709243697478992"
```

```
plot(perfB_ridge2)
abline(0, 1, col="grey", lty=2)
```



lasso

```
predictedSensitivityU133b_lasso <- calcPhenotype_lasso(exprDataU133b[, bortIndex], trainDataOrd, bortic
selection=1)
```

```
## Warning in if (class(testExprData) != "matrix") stop("ERROR: \"testExprData\"
```

```
## must be a matrix."): the condition has length > 1 and only the first element
## will be used
```

```
## Warning in if (class(trainingExprData) != "matrix") stop("ERROR:
## \"trainingExprData\" must be a matrix."): the condition has length > 1 and only
## the first element will be used
```

```
##
##  3919  gene identifiers overlap between the supplied expression matrices...
##
```

```
## Found2batches
```

```
## Adjusting for0covariate(s) or covariate level(s)
```

```
## Standardizing Data across genes
```

```
## Fitting L/S model and finding priors
```

```
## Finding parametric adjustments
```

```
## Adjusting the Data
```

```
##
## Fitting glmnet lasso Regression model... Done
##
## Calculating predicted phenotype...
## best lambda min is: 0.07543558
```

```
## Warning in if (class(homData$test) == "numeric") {: the condition has length > 1
## and only the first element will be used
```

```
## Done
```

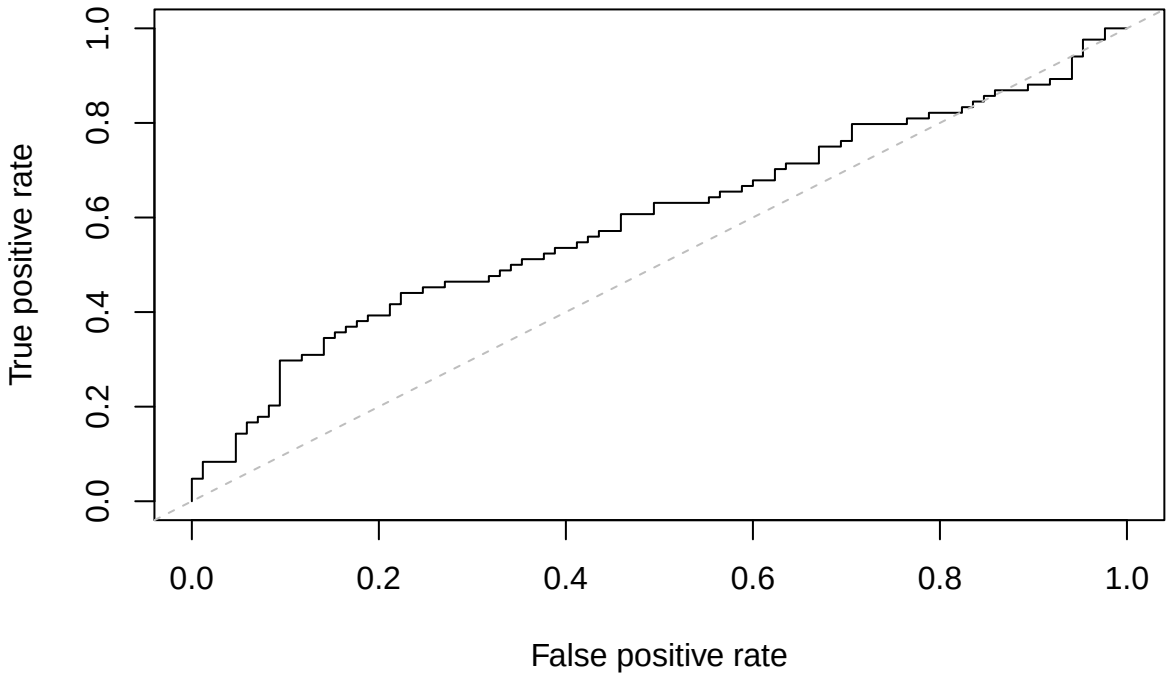
```
predictedSensitivityU133b_lasso1 <- predictedSensitivityU133b_lasso$preds
```

```
l_U133b_lasso  <- list("Responder"=predictedSensitivityU133b_lasso1[respU133b == "PGx_Responder = R"]
, "Non-responder"=predictedSensitivityU133b_lasso1[respU133b == "PGx_Responder = NR"])
```

```
predB_lasso  <- prediction(c(l_U133b_lasso[[1]], l_U133b_lasso[[2]]), c(rep("sens", length(l_U133b_lasso
rep("res", length(l_U133b_lasso [[2]]))),
label.ordering=c("sens", "res"))
perfB_lasso  <- performance(predB_lasso , measure = "tpr", x.measure = "fpr")
print(paste("AUC:", performance(predB_lasso , measure = "auc")@"y.values"[[1]]))
```

```
## [1] "AUC: 0.594677871148459"
```

```
plot(perfB_lasso)
abline(0, 1, col="grey", lty=2)
```



elestic net

```
predictedSensitivityU133b_enet <- calcPhenotype_enet(exprDataU133b[, bortIndex], trainDataOrd, bortic50s
selection=1)
```

```
## Warning in if (class(testExprData) != "matrix") stop("ERROR: \"testExprData\"
## must be a matrix."): the condition has length > 1 and only the first element
## will be used
```

```
## Warning in if (class(trainingExprData) != "matrix") stop("ERROR:
## \"trainingExprData\" must be a matrix."): the condition has length > 1 and only
## the first element will be used
```

```
##
##  3919  gene identifiers overlap between the supplied expression matrices...
##
```

```
## Found2batches
```

```
## Adjusting for0covariate(s) or covariate level(s)
```

```
## Standardizing Data across genes
```

```
## Fitting L/S model and finding priors
```

```
## Finding parametric adjustments
```

```
## Adjusting the Data
```

```
##
## Fitting glmnet enet Regression model... Done
##
## Calculating predicted phenotype...
## best lambda min is: 0.1374682
```

```
## Warning in if (class(homData$test) == "numeric") {: the condition has length > 1
## and only the first element will be used
```

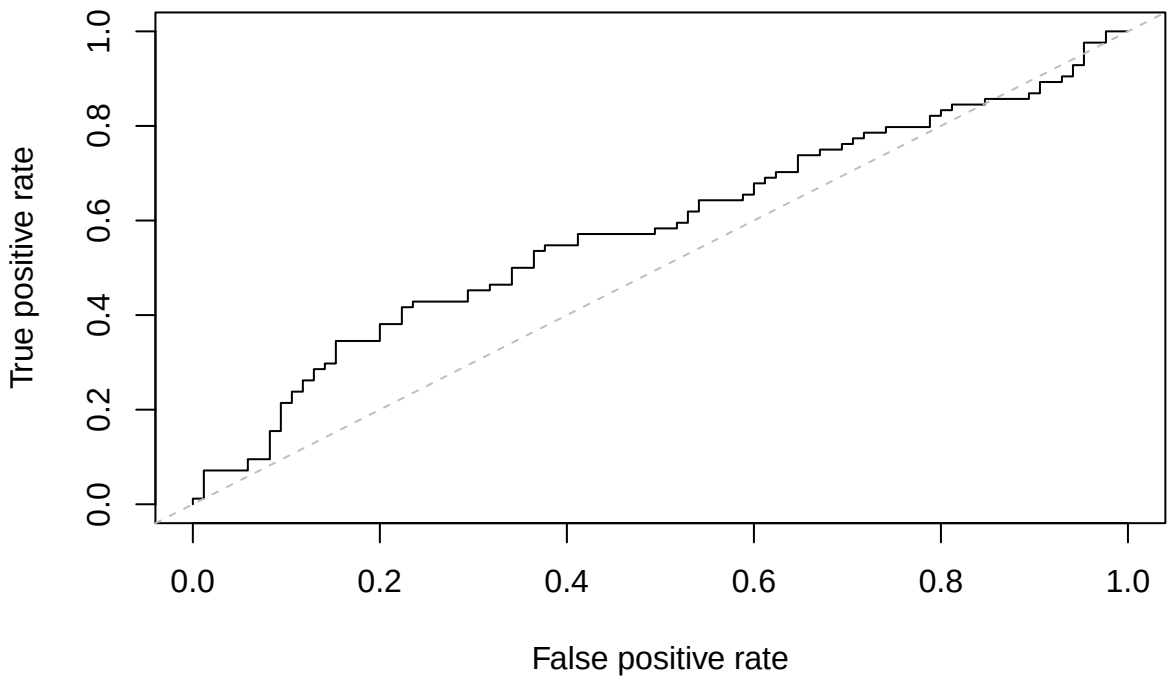
```
## Done
```

```
l_U133b_enet <- list("Responder"=predictedSensitivityU133b_enet[respU133b == "PGx_Responder = R"]
, "Non-responder"=predictedSensitivityU133b_enet[respU133b == "PGx_Responder = NR"])
```

```
predB_enet <- prediction(c(l_U133b_enet[[1]], l_U133b_enet[[2]]), c(rep("sens", length(l_U133b_enet[[1]]
rep("res", length(l_U133b_enet[[2]]))),
label.ordering=c("sens", "res"))
perfB_enet <- performance(predB_enet, measure = "tpr", x.measure = "fpr")
print(paste("AUC:", performance(predB_enet, measure = "auc")@"y.values"[[1]]))
```

```
## [1] "AUC: 0.580812324929972"
```

```
plot(perfB_enet)
abline(0, 1, col="grey", lty=2)
```



svm

```
predictedSensitivityU133b_svm <- calcPhenotype_svm(exprDataU133b[, bortIndex], trainDataOrd, bortic50sO
selection=1)
```

```
## Warning in if (class(testExprData) != "matrix") stop("ERROR: \"testExprData\"
## must be a matrix."): the condition has length > 1 and only the first element
## will be used
```

```
## Warning in if (class(trainingExprData) != "matrix") stop("ERROR:
## \"trainingExprData\" must be a matrix."): the condition has length > 1 and only
## the first element will be used
```

```
##
##  3919  gene identifiers overlap between the supplied expression matrices...
##
```

```
## Found2batches
```

```
## Adjusting for0covariate(s) or covariate level(s)
```

```
## Standardizing Data across genes
```

```
## Fitting L/S model and finding priors
```

```
## Finding parametric adjustments
```

```
## Adjusting the Data
```

```
##
## Fitting svm Regression model... Done
##
## Calculating predicted phenotype... best gamma 0.0003244646 best cost 1
```

```
## Warning in if (class(homData$test) == "numeric") {: the condition has length > 1
## and only the first element will be used
```

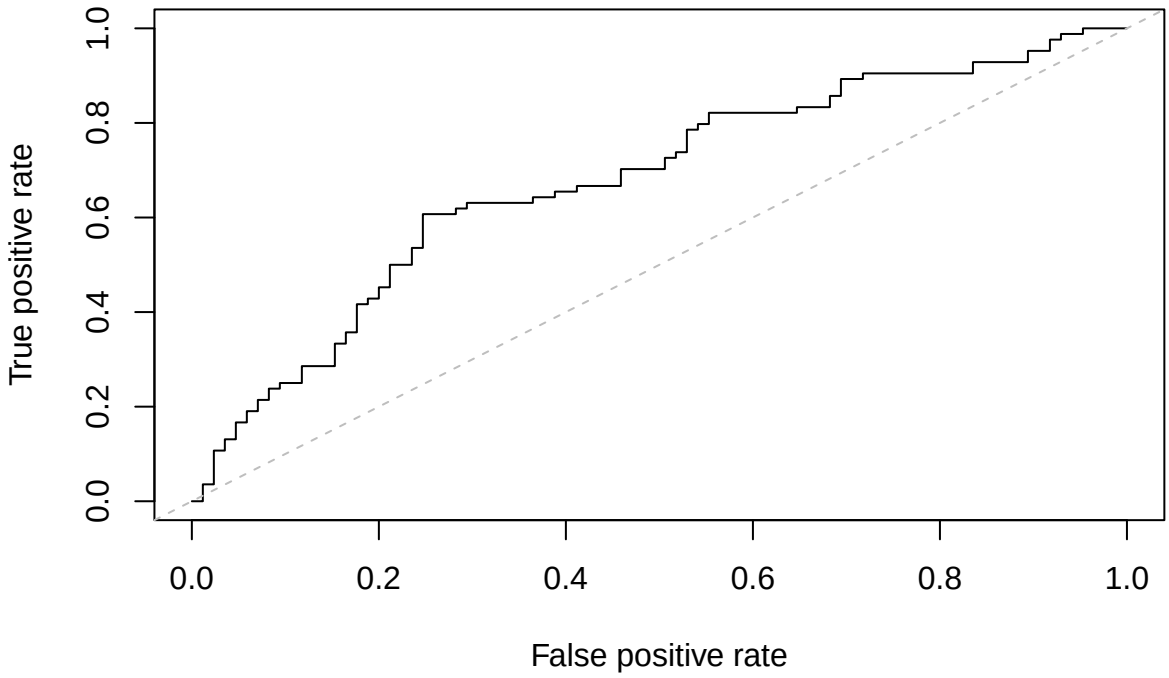
```
## Done
```

```
l_U133b_svm  <- list("Responder"=predictedSensitivityU133b_svm[respU133b == "PGx_Responder = R"]
, "Non-responder"=predictedSensitivityU133b_svm[respU133b == "PGx_Responder = NR"])
```

```
predB_svm  <- prediction(c(l_U133b_svm[[1]], l_U133b_svm[[2]]), c(rep("sens", length(l_U133b_svm[[1]])),
rep("res", length(l_U133b_svm[[2]]))),
label.ordering=c("sens", "res"))
perfB_svm  <- performance(predB_svm , measure = "tpr", x.measure = "fpr")
print(paste("AUC:", performance(predB_svm , measure = "auc")@"y.values"[[1]]))
```

```
## [1] "AUC: 0.677310924369748"
```

```
plot(perfB_svm )
abline(0, 1, col="grey", lty=2)
```



ann

```
# lasso selected variables
lasso_select_vars2B <- c(predictedSensitivityU133b_lasso$coeffs$features)

# remove intercept
lasso_select_vars2B <-  lasso_select_vars2B[-1]
lasso_select_vars2B
```

```
##  [1] "EGR1"     "COPZ1"    "SSPN"     "ANXA1"    "PARP6"    "GTF2A1"
##  [7] "GLI3"     "ZNF12"    "CLP1"     "SOCS7"    "FKBP14"   "PCDH7"
## [13] "GLIPR1"   "RIPK1"    "AHCTF1"   "DDA1"     "ST8SIA3"  "HELZ"
## [19] "RBM25"    "RSU1"     "NFASC"    "ZNF638"   "ZCCHC2"   "ALKBH1"
## [25] "ARID5B"   "SERPINE2" "LDB3"     "RBM12B"
```

```
length(lasso_select_vars2B)
```

```
## [1] 28
```

```
trainDataOrd_subset2B <- trainDataOrd[rownames(trainDataOrd) %in% lasso_select_vars2B, ]  # Extract row
dim(trainDataOrd_subset2B)                                                                # Print data frame subset
```

```
## [1]  28 280
```

```
predictedSensitivityU133b_ann <- calcPhenotype_ann(exprDataU133b[, bortIndex], trainDataOrd_subset2B, b
selection=1)
```

```
## Warning in if (class(testExprData) != "matrix") stop("ERROR: \"testExprData\"
## must be a matrix."): the condition has length > 1 and only the first element
## will be used

## Warning in if (class(trainingExprData) != "matrix") stop("ERROR:
## \"trainingExprData\" must be a matrix."): the condition has length > 1 and only
## the first element will be used

##
##  28  gene identifiers overlap between the supplied expression matrices...
##

## Found2batches

## Adjusting for0covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data

##
## Fitting ANN Regression model... # weights:  241
## initial  value 3479.806504
## iter  10 value 79.952006
## iter  20 value 79.913795
## iter  30 value 74.468418
## iter  40 value 57.242496
## iter  50 value 56.640142
## iter  60 value 54.951019
## iter  70 value 54.378602
## iter  80 value 53.590538
## iter  90 value 52.720242
## iter 100 value 51.517848
## iter 110 value 50.087492
## iter 120 value 49.441833
## iter 130 value 48.567900
## iter 140 value 48.200768
## iter 150 value 48.166070
## iter 160 value 47.678034
## iter 170 value 46.622919
## iter 180 value 45.511664
## iter 190 value 44.410423
## iter 200 value 43.768777
## iter 210 value 43.372564
## iter 220 value 43.196001
## iter 230 value 43.041319
## iter 240 value 42.824571
## iter 250 value 42.527378
## iter 260 value 42.308697
## iter 270 value 41.956137
## iter 280 value 40.784568
## iter 290 value 38.948088
## iter 300 value 37.074150
## iter 310 value 36.144601
```

```
## iter 320 value 34.723801
## iter 330 value 33.460392
## iter 340 value 32.411282
## iter 350 value 32.043091
## iter 360 value 31.862154
## iter 370 value 31.740095
## iter 380 value 31.667969
## iter 390 value 31.599917
## iter 400 value 31.543485
## iter 410 value 31.509143
## iter 420 value 31.459238
## iter 430 value 31.403502
## iter 440 value 31.362518
## iter 450 value 31.333843
## iter 460 value 31.313164
## iter 470 value 31.305009
## iter 480 value 31.281467
## iter 490 value 31.251980
## iter 500 value 31.208384
## iter 510 value 31.132576
## iter 520 value 30.963434
## iter 530 value 30.847699
## iter 540 value 30.793681
## iter 550 value 30.758855
## iter 560 value 30.684238
## iter 570 value 30.600920
## iter 580 value 30.554916
## iter 590 value 30.527487
## iter 600 value 30.492976
## iter 610 value 30.468402
## iter 620 value 30.456356
## iter 630 value 30.446793
## iter 640 value 30.424179
## iter 650 value 30.406746
## iter 660 value 30.400000
## iter 670 value 30.394574
## iter 680 value 30.388355
## iter 690 value 30.378804
## iter 700 value 30.368644
## iter 710 value 30.349797
## iter 720 value 30.347013
## iter 730 value 30.343057
## iter 740 value 30.334871
## iter 750 value 30.320523
## iter 760 value 30.298823
## iter 770 value 30.280573
## iter 780 value 30.259653
## iter 790 value 30.240551
## iter 800 value 30.235713
## iter 810 value 30.227327
## iter 820 value 30.207703
## iter 830 value 30.186142
## iter 840 value 30.152690
## iter 850 value 30.135806
```

```
## iter 860 value 30.129712
## iter 870 value 30.125556
## iter 880 value 30.089250
## iter 890 value 30.082766
## iter 900 value 30.077357
## iter 910 value 30.070809
## iter 920 value 30.056551
## iter 930 value 30.043947
## iter 940 value 30.038181
## iter 950 value 30.024479
## iter 960 value 29.994069
## iter 970 value 29.951481
## iter 980 value 29.898613
## iter 990 value 29.791852
## iter1000 value 29.506111
## iter1010 value 29.101731
## iter1020 value 28.430380
## iter1030 value 27.874130
## iter1040 value 27.182353
## iter1050 value 26.949808
## iter1060 value 26.692566
## iter1070 value 26.377456
## iter1080 value 25.944440
## iter1090 value 25.573690
## iter1100 value 25.207287
## iter1110 value 24.999429
## iter1120 value 24.499714
## iter1130 value 24.113325
## iter1140 value 23.860473
## iter1150 value 23.591381
## iter1160 value 23.131446
## iter1170 value 22.653182
## iter1180 value 22.345695
## iter1190 value 22.073092
## iter1200 value 21.735177
## iter1210 value 21.487886
## iter1220 value 21.284288
## iter1230 value 21.177166
## iter1240 value 21.046668
## iter1250 value 20.895419
## iter1260 value 20.862483
## iter1270 value 20.685725
## iter1280 value 20.443266
## iter1290 value 20.236351
## iter1300 value 20.025354
## iter1310 value 19.868341
## iter1320 value 19.714293
## iter1330 value 19.653605
## iter1340 value 19.585578
## iter1350 value 19.461159
## iter1360 value 19.367546
## iter1370 value 19.304913
## iter1380 value 19.269917
## iter1390 value 19.216811
```

```
## iter1400 value 19.180964
## iter1410 value 19.123017
## iter1420 value 19.004274
## iter1430 value 18.830798
## iter1440 value 18.708515
## iter1450 value 18.531397
## iter1460 value 18.410969
## iter1470 value 18.277763
## iter1480 value 18.186892
## iter1490 value 18.133940
## iter1500 value 18.107274
## iter1510 value 18.091359
## iter1520 value 18.074730
## iter1530 value 18.059660
## iter1540 value 18.045001
## iter1550 value 18.031591
## iter1560 value 18.023136
## iter1570 value 17.995637
## iter1580 value 17.984194
## iter1590 value 17.960200
## iter1600 value 17.948045
## iter1610 value 17.927195
## iter1620 value 17.908203
## iter1630 value 17.883904
## iter1640 value 17.850912
## iter1650 value 17.816329
## iter1660 value 17.793668
## iter1670 value 17.767043
## iter1680 value 17.751094
## iter1690 value 17.726508
## iter1700 value 17.707616
## iter1710 value 17.694590
## iter1720 value 17.684339
## iter1730 value 17.676453
## iter1740 value 17.666765
## iter1750 value 17.657032
## iter1760 value 17.630420
## iter1770 value 17.619914
## iter1780 value 17.598792
## iter1790 value 17.565001
## iter1800 value 17.436741
## iter1810 value 17.249180
## iter1820 value 17.165168
## iter1830 value 17.088410
## iter1840 value 17.032953
## iter1850 value 16.986007
## iter1860 value 16.927800
## iter1870 value 16.851173
## iter1880 value 16.789495
## iter1890 value 16.759617
## iter1900 value 16.728945
## iter1910 value 16.679466
## iter1920 value 16.639044
## iter1930 value 16.613176
```

```
## iter1940 value 16.592888
## iter1950 value 16.556068
## iter1960 value 16.545930
## iter1970 value 16.530521
## iter1980 value 16.526185
## iter1990 value 16.521457
## iter2000 value 16.517313
## iter2010 value 16.506544
## iter2020 value 16.496321
## iter2030 value 16.483463
## iter2040 value 16.475236
## iter2050 value 16.470153
## iter2060 value 16.462522
## iter2070 value 16.454454
## iter2080 value 16.445320
## iter2090 value 16.439454
## iter2100 value 16.435095
## iter2110 value 16.429892
## iter2120 value 16.418561
## iter2130 value 16.379405
## iter2140 value 16.375150
## iter2150 value 16.368737
## iter2160 value 16.360495
## iter2170 value 16.344172
## iter2180 value 16.316175
## iter2190 value 16.296299
## iter2200 value 16.285172
## iter2210 value 16.274685
## iter2220 value 16.261491
## iter2230 value 16.253792
## iter2240 value 16.249792
## iter2250 value 16.246865
## iter2260 value 16.242241
## iter2270 value 16.238813
## iter2280 value 16.235784
## iter2290 value 16.232247
## iter2300 value 16.220668
## iter2310 value 16.020172
## iter2320 value 15.863292
## iter2330 value 15.633048
## iter2340 value 15.483987
## iter2350 value 15.369185
## iter2360 value 15.329872
## iter2370 value 15.309154
## iter2380 value 15.278569
## iter2390 value 15.243529
## iter2400 value 15.230957
## iter2410 value 15.209607
## iter2420 value 15.189483
## iter2430 value 15.181800
## iter2440 value 15.169781
## iter2450 value 15.158730
## iter2460 value 15.149259
## iter2470 value 15.145152
```

```
## iter2480 value 15.140717
## iter2490 value 15.137325
## iter2500 value 15.135114
## iter2510 value 15.131633
## iter2520 value 15.129905
## iter2530 value 15.065013
## iter2540 value 15.014925
## iter2550 value 14.997835
## iter2560 value 14.985396
## iter2570 value 14.973130
## iter2580 value 14.948823
## iter2590 value 14.916859
## iter2600 value 14.893395
## iter2610 value 14.880554
## iter2620 value 14.877082
## iter2630 value 14.871644
## iter2640 value 14.866346
## iter2650 value 14.863311
## iter2660 value 14.861026
## iter2670 value 14.859173
## iter2680 value 14.856607
## iter2690 value 14.855234
## iter2700 value 14.854003
## iter2710 value 14.852448
## iter2720 value 14.851224
## iter2730 value 14.847269
## iter2740 value 14.844749
## iter2750 value 14.843048
## iter2760 value 14.841207
## iter2770 value 14.840204
## iter2780 value 14.839183
## iter2790 value 14.829024
## iter2800 value 14.820490
## iter2810 value 14.811741
## iter2820 value 14.808684
## iter2830 value 14.802724
## iter2840 value 14.717279
## iter2850 value 14.643398
## iter2860 value 14.622546
## iter2870 value 14.605913
## iter2880 value 14.597582
## iter2890 value 14.588366
## iter2900 value 14.574285
## iter2910 value 14.561543
## iter2920 value 14.556579
## iter2930 value 14.552538
## iter2940 value 14.546331
## iter2950 value 14.537772
## iter2960 value 14.532220
## iter2970 value 14.529376
## iter2980 value 14.527407
## iter2990 value 14.525886
## iter3000 value 14.525036
## iter3010 value 14.524230
```

```
## iter3020 value 14.523460
## iter3030 value 14.521655
## iter3040 value 14.513312
## iter3050 value 14.507122
## iter3060 value 14.504806
## iter3070 value 14.503170
## iter3080 value 14.502472
## iter3090 value 14.502155
## iter3100 value 14.501913
## iter3110 value 14.378354
## iter3120 value 14.365129
## iter3130 value 14.360717
## iter3140 value 14.356098
## iter3150 value 14.355724
## iter3160 value 14.355487
## iter3170 value 14.355316
## iter3180 value 14.355292
## final  value 14.355290
## converged
## Done
##
## Calculating predicted phenotype... /nbest size is: 10
```

```
## Warning in if (class(homData$test) == "numeric") {: the condition has length > 1
## and only the first element will be used
```

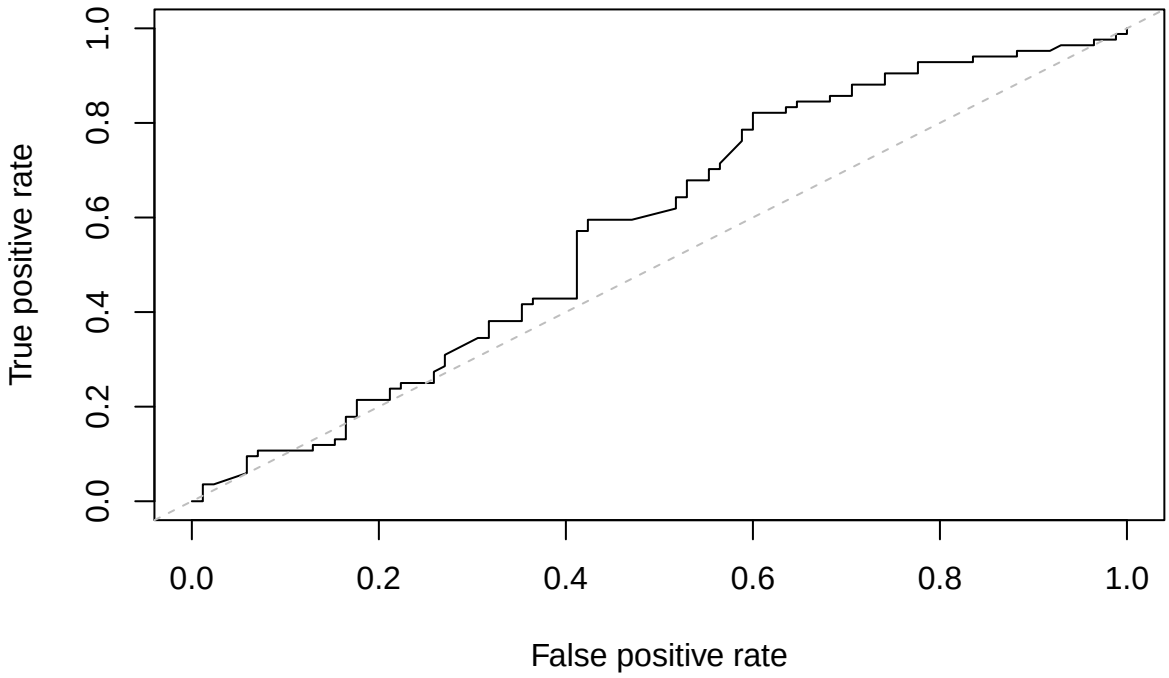
```
## Done
```

```
l_U133b_ann  <- list("Responder"=predictedSensitivityU133b_ann[respU133b == "PGx_Responder = R"]
, "Non-responder"=predictedSensitivityU133b_ann[respU133b == "PGx_Responder = NR"])
```

```
predB_ann  <- prediction(c(l_U133b_ann[[1]], l_U133b_ann[[2]]), c(rep("sens", length(l_U133b_ann[[1]]))
rep("res", length(l_U133b_ann[[2]]))),
label.ordering=c("sens", "res"))
perfB_ann  <- performance(predB_ann , measure = "tpr", x.measure = "fpr")
print(paste("AUC:", performance(predB_ann , measure = "auc")@"y.values"[[1]]))
```

```
## [1] "AUC: 0.578501400560224"
```

```
plot(perfB_ann )
abline(0, 1, col="grey", lty=2)
```



pcr

```
predictedSensitivityU133b_pcr <- calcPhenotype_pcr(exprDataU133b[, bortIndex], trainDataOrd, bortic50sO
,Ncomp = 90 ,selection=1)
```

```
## Warning in if (class(testExprData) != "matrix") stop("ERROR: \"testExprData\"
## must be a matrix."): the condition has length > 1 and only the first element
## will be used
```

```
## Warning in if (class(trainingExprData) != "matrix") stop("ERROR:
## \"trainingExprData\" must be a matrix."): the condition has length > 1 and only
## the first element will be used
```

```
##
##  3919  gene identifiers overlap between the supplied expression matrices...
##
```

```
## Found2batches
```

```
## Adjusting for0covariate(s) or covariate level(s)
```

```
## Standardizing Data across genes
```

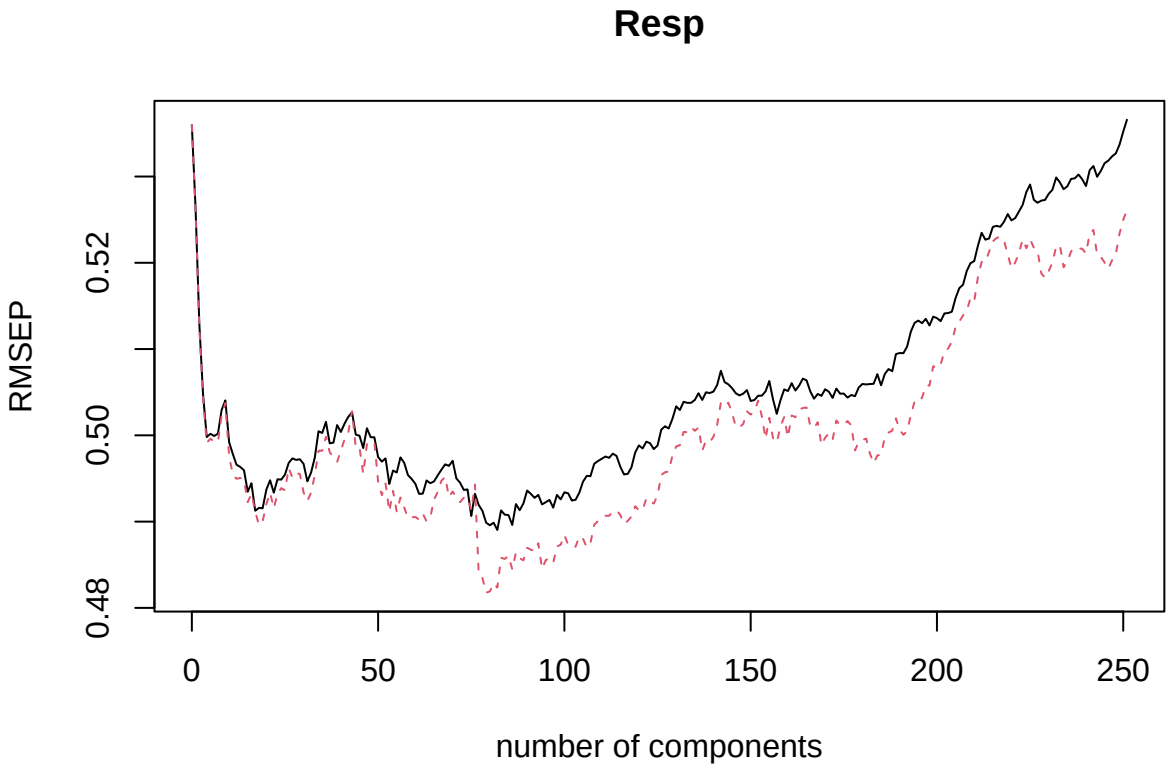
```
## Fitting L/S model and finding priors
```

```
## Finding parametric adjustments
```

```
## Adjusting the Data
```

```
##
## Fitting PCR Regression model... Done
##
## Calculating predicted phenotype... /n
```

```
## Warning in if (class(homData$test) == "numeric") {: the condition has length > 1
## and only the first element will be used
```



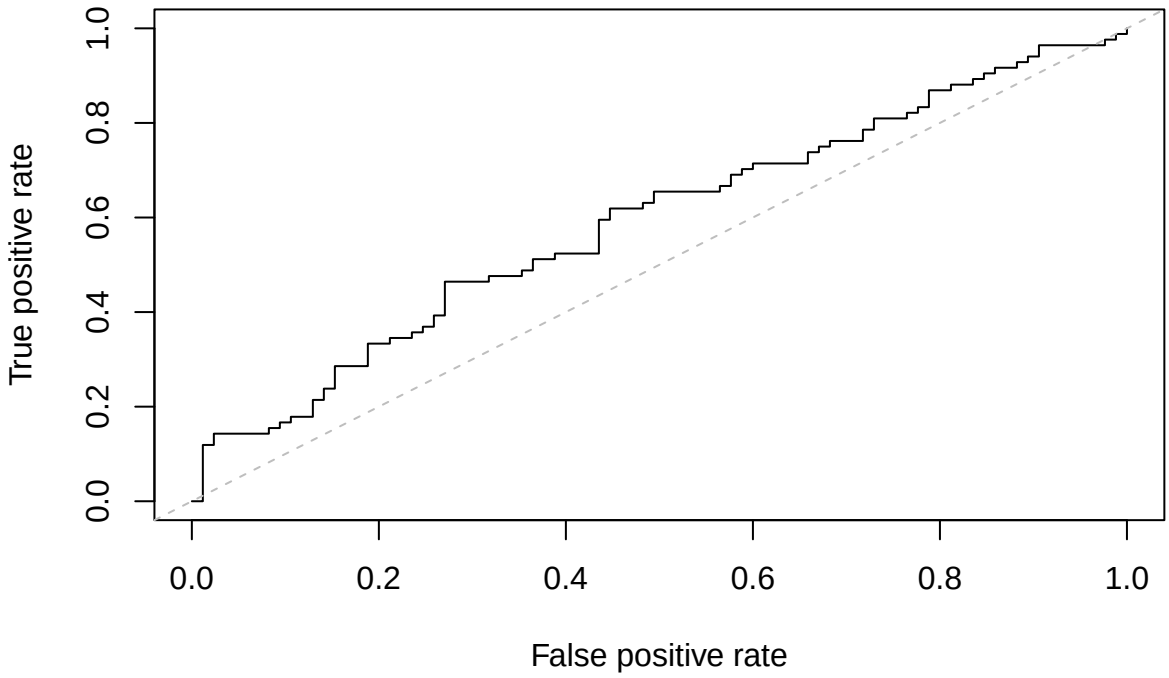

```
## Done
```

```
l_U133b_pcr  <- list("Responder"=predictedSensitivityU133b_pcr[respU133b == "PGx_Responder = R"]
, "Non-responder"=predictedSensitivityU133b_pcr[respU133b == "PGx_Responder = NR"])
```

```
predB_pcr  <- prediction(c(l_U133b_pcr[[1]], l_U133b_pcr[[2]]), c(rep("sens", length(l_U133b_pcr[[1]]))
rep("res", length(l_U133b_pcr[[2]]))),
label.ordering=c("sens", "res"))
perfB_pcr  <- performance(predB_pcr , measure = "tpr", x.measure = "fpr")
print(paste("AUC:", performance(predB_pcr , measure = "auc")@"y.values"[[1]]))
```

```
## [1] "AUC: 0.590896358543417"
```

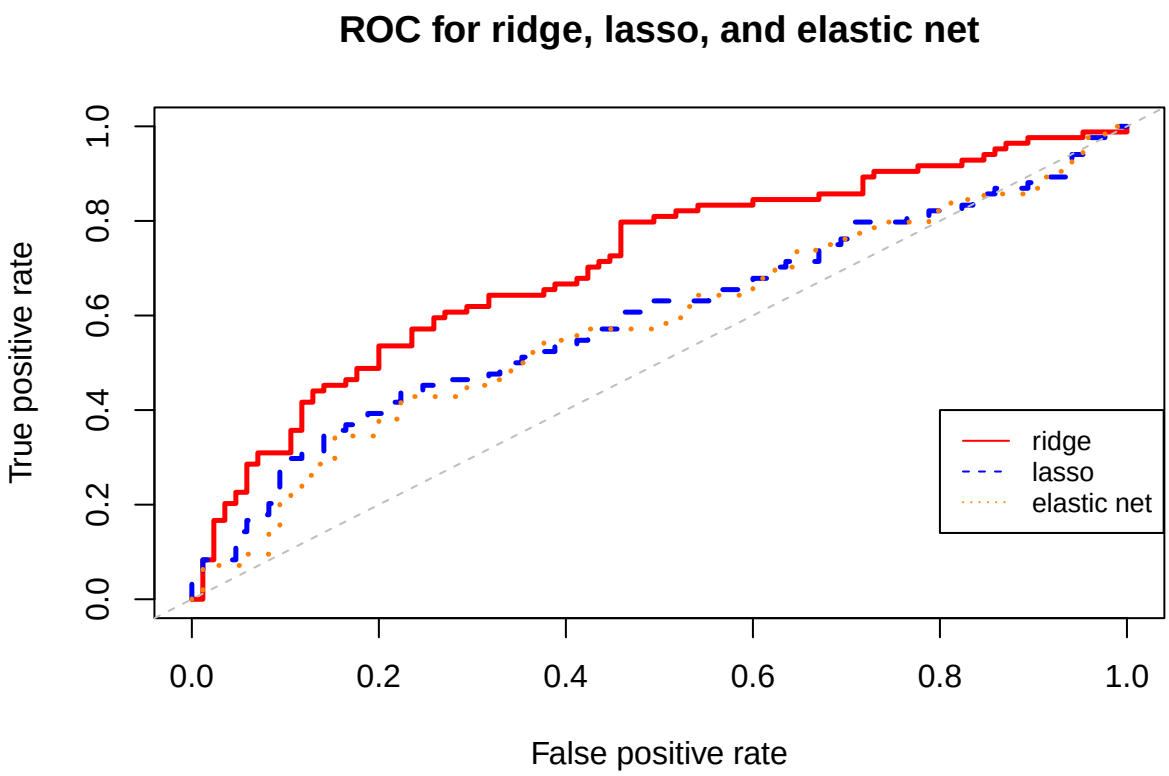
```
plot(perfB_pcr )
abline(0, 1, col="grey", lty=2)
```



roc curves

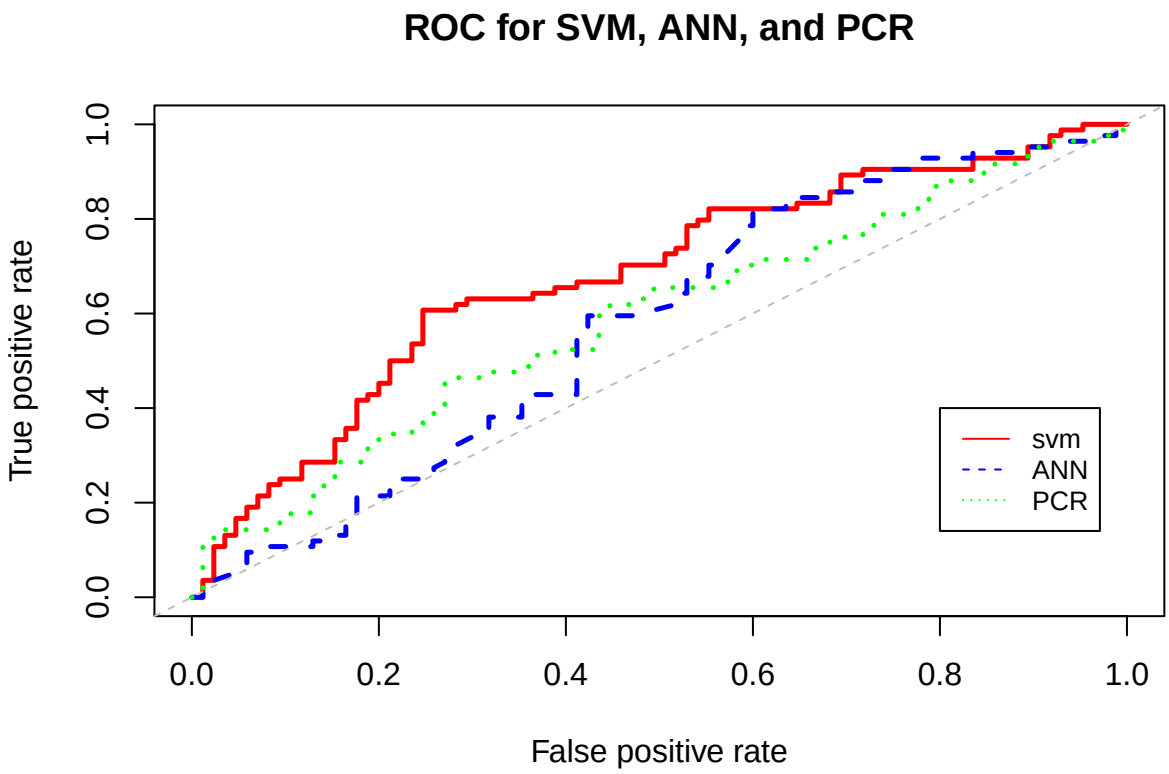
1st set of plots: ridge, lasso, elestic net

```
plot(perfB_ridge2 ,main="ROC for ridge, lasso, and elastic net", col="red", lty=1,lwd = 2.5)
lines(perfB_lasso@x.values[[1]], perfB_lasso@y.values[[1]], col="blue", lty=2,lwd = 2.5)
lines(perfB_enet@x.values[[1]], perfB_enet@y.values[[1]], col="darkorange1", lty=3,lwd = 2.5)
abline(0, 1, col="grey", lty=2)
legend(0.8, 0.4, legend=c("ridge", "lasso","elastic net"),
       col=c("red", "blue","darkorange1"), lty=1:3, cex=0.8)
```



part 2: svm, ann, pcr

```
plot(perfB_svm ,main="ROC for SVM, ANN, and PCR", col="red", lty=1,lwd = 2.5)
lines(perfB_ann@x.values[[1]], perfB_ann@y.values[[1]], col="blue", lty=2,lwd = 2.5)
lines(perfB_pcr@x.values[[1]], perfB_pcr@y.values[[1]], col="green", lty=3,lwd = 2.5)
abline(0, 1, col="grey", lty=2)
legend(0.8, 0.4, legend=c("svm", "ANN","PCR"),
       col=c("red", "blue","green"), lty=1:3, cex=0.8)
```


ROC for SVM, ANN, and PCR
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
svm
ANN
PCR