Noah L. Schrick
1492657

Lab 5. Gene expression Part 3: Feature Selection. We will use the major depressive disorder (MDD) RNA-Seq data to extend our use of basic statistics to statistical learning to perform feature selection, which is a more advanced way to identify genes that are differentially expressed between phenotype groups (*e.g.*, between disease and healthy control). So far you have learned to normalize and transform data, filter genes based on variation, cluster subjects, perform differential expression using the t-test, and use a gene set enrichment analysis tool. In this lab, you will use the generalized linear model (GLM) called logistic regression, which allows you to include more sources of variation (covariates like sex) than the t-test. You will also use other machine learning feature selection methods.

Step 0. Process and filter data following previous labs.

```
# load gene expression data
load("sense.filtered.cpm.Rdata")  # setwd!

# load phenotype (mdd/hc) data
subject.attrs <- read.csv("Demographic_symptom.csv",
                          stringsAsFactors = FALSE)
library(dplyr)
# grab intersecting X (subject ids) and Diag (Diagnosis) from columns
phenos.df <- subject.attrs %>%
             filter(X %in% colnames(sense.filtered.cpm)) %>%
             dplyr::select(X, Diag)
mddPheno <- as.factor(phenos.df$Diag)

# Normalized and transform
library(preprocessCore)
mddExprData_quantile <- normalize.quantiles(sense.filtered.cpm)
mddExprData_quantileLog2 <- log2(mddExprData_quantile)
# attach phenotype names and gene names to data
colnames(mddExprData_quantileLog2) <- mddPheno
rownames(mddExprData_quantileLog2) <- rownames(sense.filtered.cpm)


# coefficient of variation filter sd(x)/abs(mean(x))
CoV_values <- apply(mddExprData_quantileLog2,1,
                    function(x) {sd(x)/abs(mean(x))})
# smaller threshold, the higher the experimental effect relative to the
# measurement precision
sum(CoV_values<.045)
# there is one gene that has 0 variation -- remove
sd_values <- apply(mddExprData_quantileLog2,1, function(x) {sd(x)})
rownames(mddExprData_quantileLog2)[sd_values==0]
# filter the data matrix
GxS.covfilter <- mddExprData_quantileLog2[CoV_values<.045 & sd_values>0,]
```

```
dim(GxS.covfilter)

# convert phenotype to factor
pheno.factor <- as.factor(colnames(GxS.covfilter))
pheno.factor
str(pheno.factor)
levels(pheno.factor)
```

A . Logistic regression. Logistic regression estimates the beta parameters that minimize the following model fit to the outcome data (phenotype class, C). We choose to encode the MDD status as C=1 and HC status as C=0. This makes HC the reference level (baseline) so the probability means probability of being in the MDD class. The predictor variable is a gene G with an expression level x.

$$\log\left(\frac{C=1 \vee G=x}{C=0 \vee G=x}\right)=\beta_o+\beta_1 x$$

In the glm function, R chooses the reference level to be the first level of the factor. In `pheno.factor`, we want to use `relevel` to make sure the reference level is HC.

```
# make sure HC is the reference level
pheno.factor.relevel <- relevel(pheno.factor,"HC")
levels(pheno.factor.relevel)
# also rename levels "0"/"1" from 1/2
levels(pheno.factor.relevel)[levels(pheno.factor.relevel)=="MDD"] <- 1
levels(pheno.factor.relevel)[levels(pheno.factor.relevel)=="HC"] <- 0
```

Run the following code to fit a logistic model of the first gene (`gene.row <- 2`) to the phenotype data. **1.** What are the coefficients for the slope and intercept of the model and their p-values? Based on the p-value of the slope, is this a good model?

```
gene.row <- 2
gene.name <-rownames(GxS.covfilter)[gene.row]
gene.expr <- GxS.covfilter[gene.row,]
gene.fit <- glm(pheno.factor.relevel~gene.expr, family=binomial)
summary(gene.fit)
coeff.mat <- coef(summary(gene.fit))
b0 <- coeff.mat[1,1]
b1 <- coeff.mat[2,1]
b1.pval <- coeff.mat[2,4]
```

1.

	Estimate	**Std. Error**	**z value**	**Pr(>|z|)**
(Intercept)	-11.114125	7.780399	-1.428478	0.1531544
gene.expr	1.990687	1.394874	1.427144	0.1535384

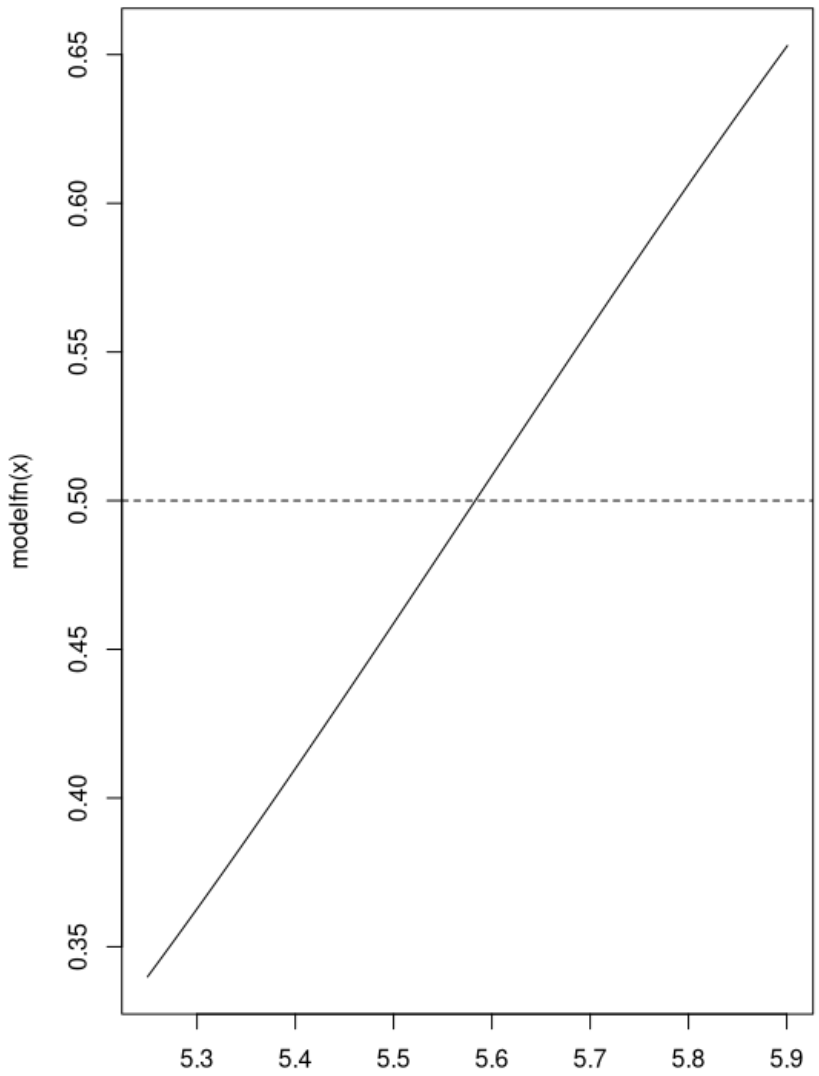
P-Value is > 0.05. This is not a good model.

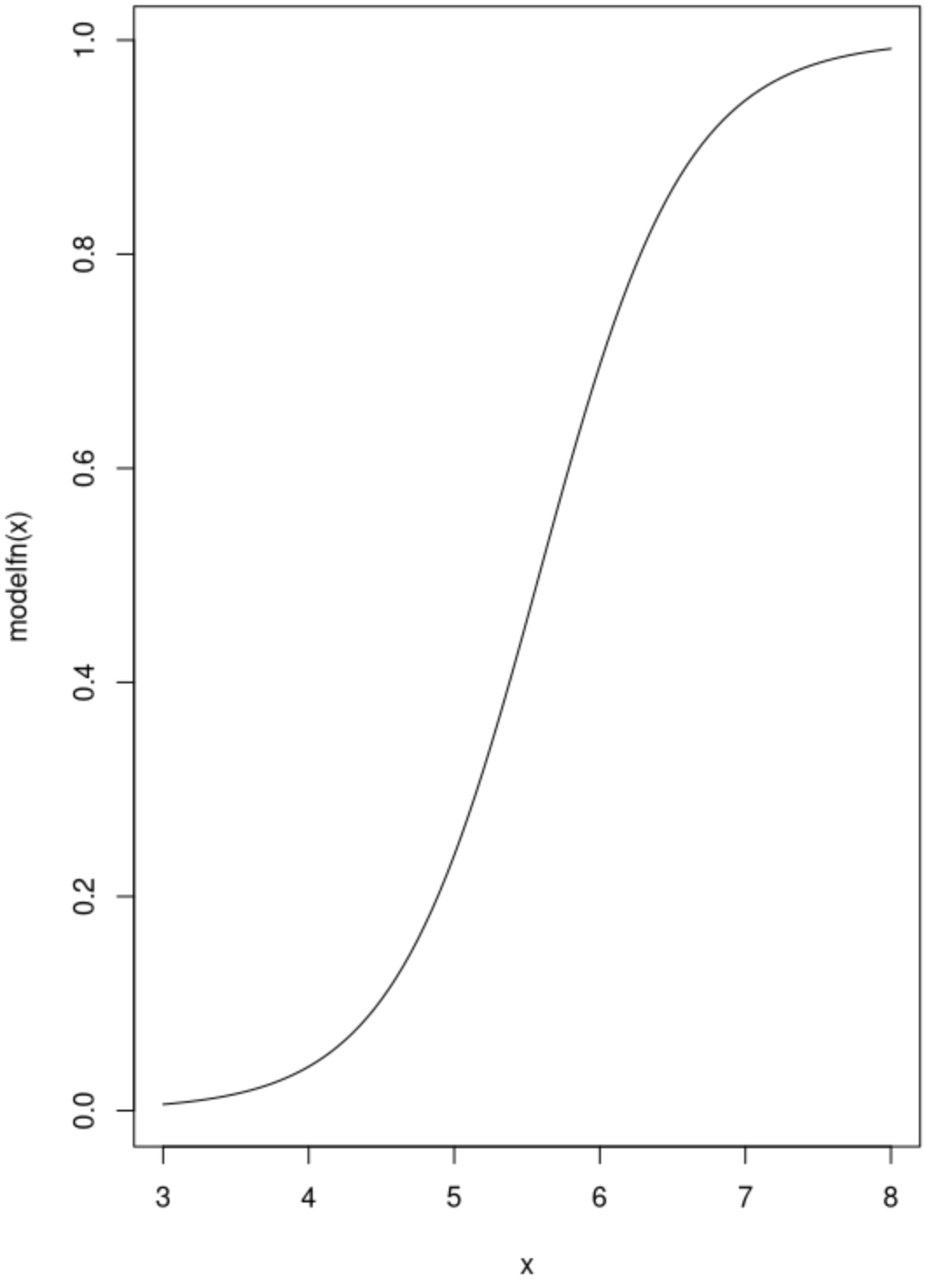
The equation below shows the logistic model that glm is fitting for the disease probability. Use the code below to show the probability predictions for this gene.

$$\text{probability(sample=MDD|gene2=x)} = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

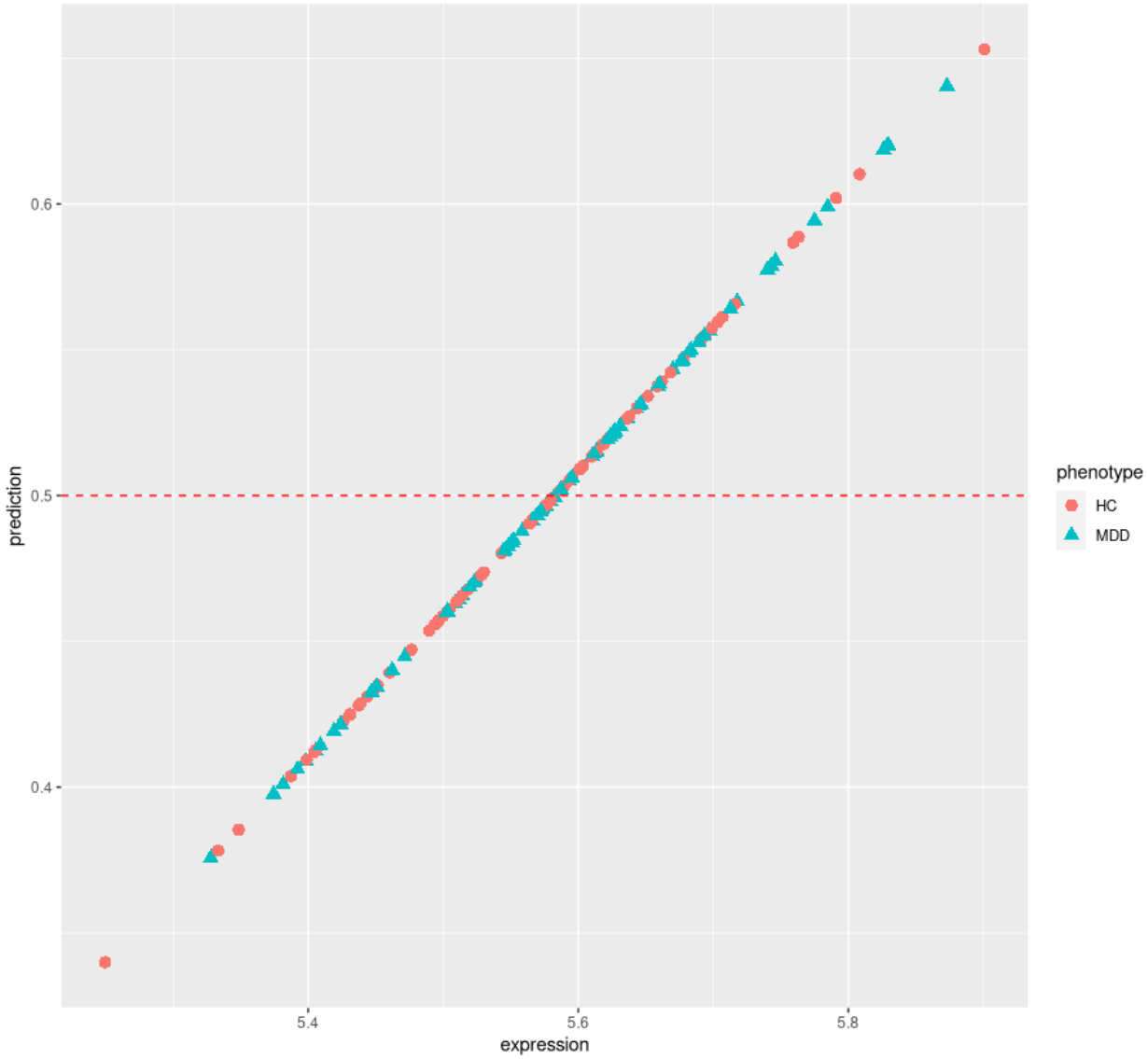
```
modelfn <- function(x){1/(1+exp(-(b0+b1*x)))}
g.min <- min(gene.expr)
g.max <- max(gene.expr)
curve(modelfn,g.min,g.max)  # plot for domain of actual gene's expression
abline(h=.5, lty=2)
curve(modelfn,3,8)     # replot with extended domain to see the S shape
abline(h=.5, lty=2)
```

Another way to plot is to create the predicted probabilities from the model using the `predict` function. **2.** Show the plots.




modelfn(x)
0.0
0.2
0.4
0.6
0.8
1.0
3
4
5
6
7
8
x

```
Predicted.probs <- predict(gene.fit, gene.expr=gene.expr,
                                          type="response")
library(ggplot2)
phenotype <- pheno.factor # just rename for legend
gene.gg.df <- data.frame(expression=gene.expr,
                         prediction=predicted.probs)
# plot predicted probabilities versus gene expression
p <- ggplot(data=gene.gg.df)
p <- p + geom_point(aes(x=expression,
                        y=prediction,
                        color=phenotype,  # shape and color
                        shape=phenotype), # are true phenotype
                    size=3)
p <- p + geom_hline(yintercept = .5, linetype="dashed", color="red")
print(p)
```



B. Logistic regression as classification. Often logistic regression is used for inferential statistics, meaning it is used to find variables that are important for understanding something about the phenotype. However, it can also be used as a classifier (to predict whether a sample is in one group or another). The R function `predict`, used above, returns the probability (`predicted.probs`) of samples being Class=1 (MDD) given the fitted model parameters (`gene.fit`) and input expression levels of the samples (`gene.expr`).

Run the code below. **3.** What rule is used in the code to determine whether a sample is predicted to be MDD? How does the rule relate to the ggplot above? How many samples are predicted to be MDD? How many samples are correctly classified? **4.** Add code to calculate the classification accuracy (the number of correctly classified samples divided by the number of samples). Sum the correctly classified samples (`correct.classified`) and divide by the number of samples. Does this gene lead to a good classifier? (As a reference, a bad classifier would be flipping a coin, which would have 50% accuracy on average).

```
# vector of logistic output probabilities for this model (glm/eq. above)
# prob = .5 is the threshold for prediction class = 1 vs 0
predicted.class <- as.integer(predicted.probs >=.5)  # predicted class
pheno.factor.relevel  # true class
# vector of True (correctly predicted) and False (wrongly predicted)
correct.classified <- predicted.class == pheno.factor.relevel
# sum correct.classified
table(pheno.factor.relevel,predicted.class)
```

3. Part A computes a list of predicted probabilities. For this part, the R code uses the predicted probabilities, and checks each value. A boolean check is performed on each value to determine if it is greater than or equal to 0.5. This boolean check is then converted to an integer. If it's greater than or equal to 0.5, it returns a predicted class of "1" (MDD), otherwise it returns a predicted class of 0 (HC).

4.

```
# accuracy:
correct.acc <- sum(correct.classified == "TRUE") / length(correct.classified)
> correct.acc
[1] 0.5414013
```

This is slightly better than a bad classifier.

C. Logistic regression ranking of all genes. The code below writes a function to compute the glm statistics for a gene (row i) of the data matrix, then uses a for loop to create a data frame of results for all genes.

```
# logistic regression function for one gene row
lr.fn <- function(i){
  gene=rownames(GxS.covfilter)[i]
  gene.expr <- GxS.covfilter[i,]
  gene.fit <- glm(pheno.factor.relevel~gene.expr,
                  family=binomial)
  coeff.mat <- coef(summary(gene.fit))
  b1 <- coeff.mat[2,1]
  b1.pval <- coeff.mat[2,4]
  coefvec <- gene.fit$estimate # intercept, gene
  pvec <- gene.fit$p.value     # intercept, gene
  c(gene, b1, b1.pval)
}
```

Run the function for gene row 2. **5.** What does the function output?

```
lr.fn(2)
```

5.

> lr.fn(2)
[1] "AAK1" "1.99068675569476" "0.153538361756257"
Outputs the gene name, the intercept, and its p-value.

Apply the function to all genes below and sort the results. **6.** Show the top 10 genes from `lr.results.sorted`.

```
# initialize results matrix
num.genes<-nrow(GxS.covfilter)
lr.results.mat <- matrix(0, nrow=nrow(GxS.covfilter), ncol=3)
# for loop the function to all genes
for (i in 1:num.genes){
  lr.results.mat[i,] <- lr.fn(i)
}
lr.results.df <- data.frame(lr.results.mat)
colnames(lr.results.df) <- c("gene", "b1", "p.val")

# sort results b1 coefficient p-value
library(dplyr)
lr.results.sorted <- lr.results.df %>%
  mutate_at("p.val", as.character) %>%
  mutate_at("p.val", as.numeric) %>%
  arrange(p.val)
lr.results.sorted[1:10,]
```

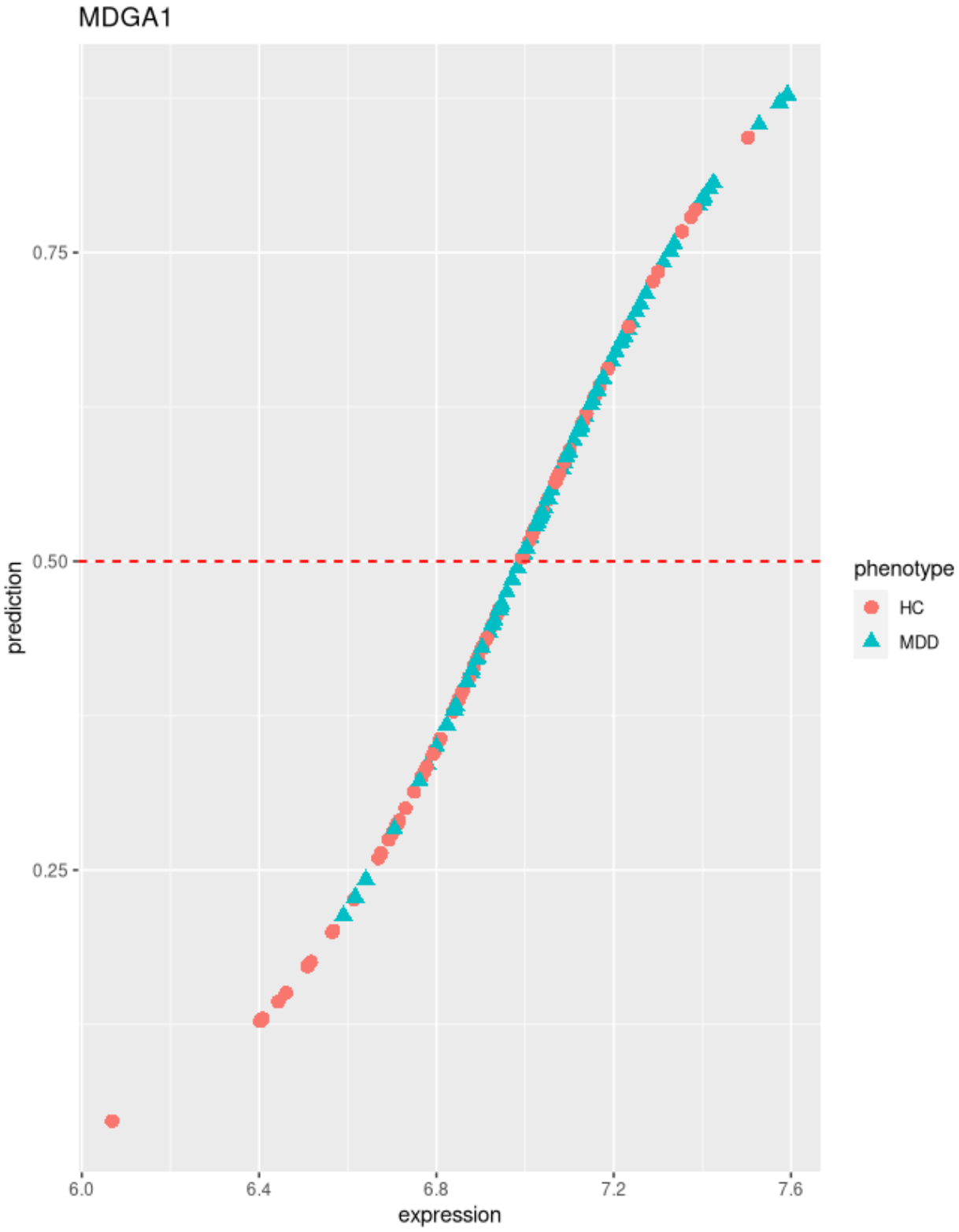
6.

	gene	b1	p.val
1	MDGA1	3.27222557173125	4.917554e-05
2	ZDHHC20	3.44752422025746	1.664413e-04
3	NPFF	-4.93598819959299	2.931886e-04
4	IRF2BPL	4.02807033481586	3.994416e-04
5	ARFGAP1	4.50243248968137	4.083146e-04
6	FAM138A	-3.90725563811716	4.308357e-04
7	UBD	2.02594802661092	4.757671e-04
8	BCL2L12	3.55042230110901	6.054376e-04
9	POGZ	-3.36092614792741	6.196730e-04
10	ZFP36L2	2.62332879572813	6.232671e-04

7. Use the code from A to create a ggplot scatter plot of the logistic regression model fit and compute the accuracy for the top gene. First we need to get the row index of the top gene:

```
gene.row <- which(rownames(GxS.covfilter)=="ENTER TOP GENE NAME HERE")
gene.expr <- GxS.covfilter[gene.row,]
gene.fit <- glm(pheno.factor.relevel~gene.expr,
                family=binomial)
predicted.probs <- predict(gene.fit, gene.expr=gene.expr,
type="response")
# Add Plotting Code
# Add code for accuracy
```

7.


MDGA1
prediction
0.75
0.50
0.25
6.0
6.4
6.8
7.2
7.6
expression
phenotype
HC
MDD

Extra note. Here is how to compute a false discovery adjusted P-value from the original/raw P-values.

```
# adjusted p-value
lr.padj <- p.adjust(lr.results.sorted$p.val, method="BH")
lr.results.sorted$p.adj <- lr.padj
lr.results.sorted[1:45,]
```

> correct.acc
[1] 0.6242038

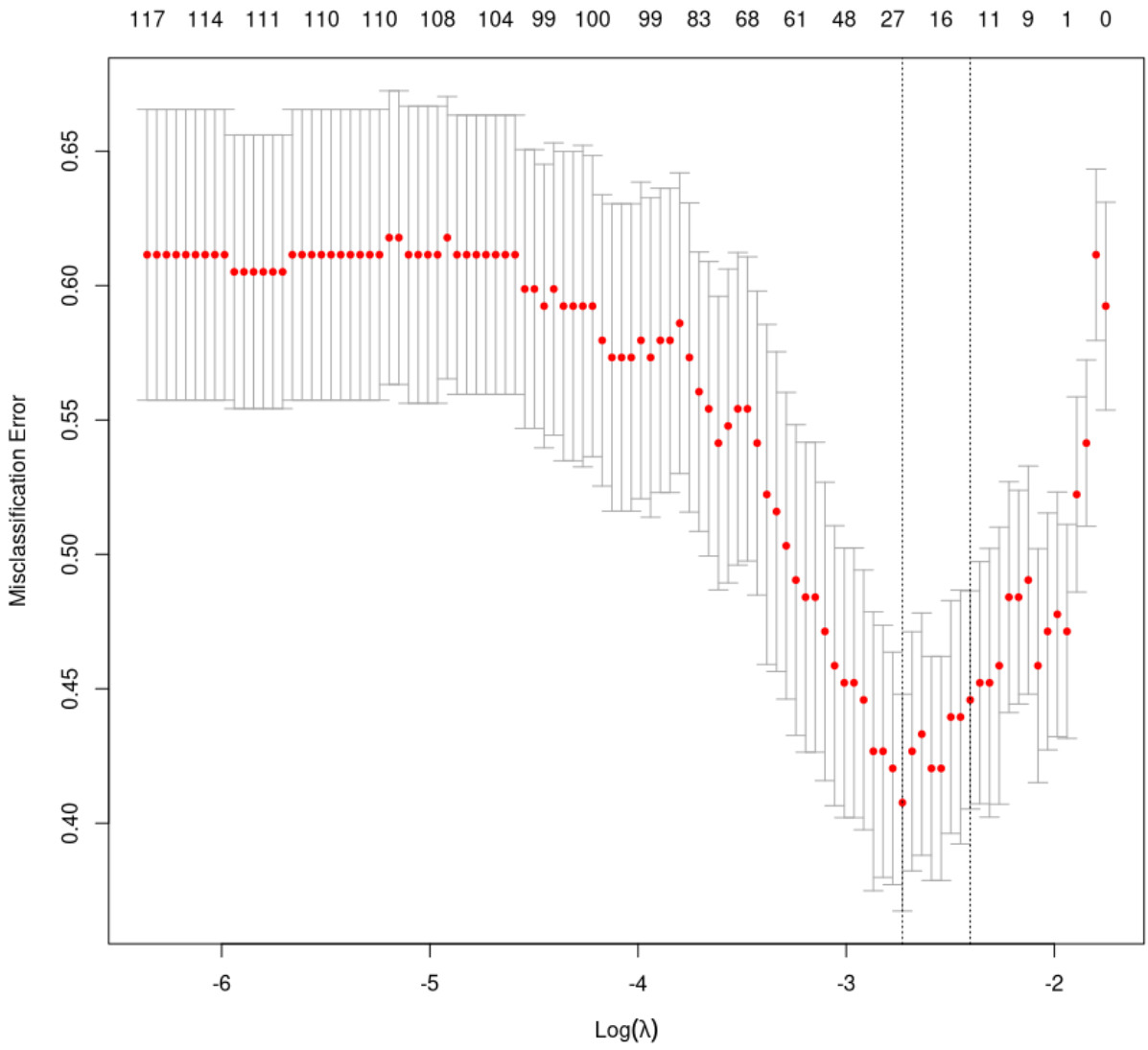
D. Statistical Learning. The following code uses GLMnet, which is a library for penalized logistic regression/classification. The regression model includes all of the genes at once and uses a penalty to help remove irrelevant variables. GLMnet combines both Ridge and Lasso when 0<alpha<1. GLMnet uses cross-validation (CV) to tune the amount of overall penalty (lambda). Penalized regression forces many of the coefficients to be zero.

```
library(glmnet) # install.packages("glmnet")

# alpha=0 means ridge, alpha=1 means lasso
# keep has to do with storing the results from cross-validiation
glmnet.model <- cv.glmnet(t(GxS.covfilter), pheno.factor.relevel, alpha=1,
                          family="binomial", type.measure="class", keep=T)
plot(glmnet.model) # plot of CV error vs lambda penalty
glmnet.model$lambda.min # lambda that minimizes the CV error
```

8. From the code above, show the CV plot. What is the lambda that minimizes the CV error? Show the genes with non-zero (important) coefficients and the classification accuracy. The accuracy is higher than regular glm logistic regression we used earlier because it is using many genes in the model and there is probably some overfitting.

8.


117 114 111 110 110 108 104 99 100 99 83 68 61 48 27 16 11 9 1 0
Misclassification Error
0.40
0.45
0.50
0.55
0.60
0.65
-6
-5
-4
-3
-2
Log(λ)

lambda that minimizes CV error: 0.06521466

genes with non-zero coefficients:
“Easy” way with

```
if (!require("coefplot")) install.packages("coefplot")
library(coefplot)
extract.coef(glmnet.model)
```

```
> extract.coef(glmnet.model)
                    Value        SE       Coefficient
(Intercept) 15.32081869 NA (Intercept)
ALAS1        0.11502152 NA       ALAS1
ATXN7L2      0.07594971 NA     ATXN7L2
C16orf74     0.50393920 NA    C16orf74
DLEC1       -0.55011200 NA       DLEC1
ELP3        -0.58275685 NA        ELP3
EXOSC3      -0.10808390 NA      EXOSC3
FAM138A     -0.40964781 NA     FAM138A
IRF2BPL      0.06336807 NA     IRF2BPL
MDGA1        1.03735782 NA      MDGA1
MEN1        -0.43343899 NA       MEN1
NPFF        -0.48181494 NA       NPFF
OSBPL3      -1.86605893 NA     OSBPL3
PANX1       -1.11155374 NA      PANX1
POGZ        -0.01244924 NA       POGZ
PPP2R5C     -0.02285152 NA    PPP2R5C
PSMG3       -0.03391447 NA      PSMG3
RASA1       -0.62975277 NA      RASA1
TUSC3        0.68148987 NA      TUSC3
UBD          0.33308961 NA        UBD
UPK1A       -0.38450494 NA      UPK1A
VDAC3       -0.36955140 NA      VDAC3
YAP1        -0.03868904 NA       YAP1
ZCWPW1       0.58062073 NA     ZCWPW1
ZDHHC20      0.33022555 NA    ZDHHC20
```

Or, from given code:

```
> glmnet.nonzero.coeffs
 [1] "(Intercept)" "ALAS1"      "ATXN7L2"    "C16orf74"   "DLEC1"
 [6] "ELP3"        "EXOSC3"     "FAM138A"    "IRF2BPL"    "MDGA1"
[11] "MEN1"        "NPFF"       "OSBPL3"     "PANX1"      "POGZ"
[16] "PPP2R5C"     "PSMG3"      "RASA1"      "TUSC3"      "UBD"
[21] "UPK1A"       "VDAC3"      "YAP1"       "ZCWPW1"     "ZDHHC20"
```

classification accuracy: 0.8152866

```
# get the penalized regression coefficients
glmnet.coeffs <- predict(glmnet.model,type="coefficients",
                         s=glmnet.model$lambda.min)

# get the coefficients that are not zero (these are the selected variables)
#glmnet.nonzero.coeffs <- #glmnet.coeffs@Dimnames[[1]][which(glmnet.coeffs!
=0)]
#glmnet.nonzero.coeffs

top_glmnet <- data.frame(as.matrix(glmnet.coeffs))      %>%
            tibble::rownames_to_column(var = "features") %>%
            filter(s1!=0, features!="(Intercept)")

top_glmnet_features <- top_glmnet %>% pull(features)

# apply the glmnet model to the data to get class predictions
glmnet.predicted <- predict(glmnet.model,
                            s=glmnet.model$lambda.min, # lambda to use
                            type="class",              # classify
                            newx=t(GxS.covfilter))     # apply to original
glmnet.accuracy <- mean(factor(glmnet.predicted)==pheno.factor.relevel)
glmnet.accuracy
table(glmnet.predicted,pheno.factor.relevel) # confusion matrix
```

Optional. LASSO does not use a P-value to rank important genes. However, it is multivariate, does automatic feature selection and reduces the number of correlated variables. How does the number of genes selected by LASSO compare with a false discovery rate adjusted P-value of the univariate P-values? Use R's `p.adjust` to selected the top group of adjusted P-values from the univariate P-values, `lr.results.sorted`.

D. Nearest Neighbor Projected Distance Regression (NPDR) is a feature selection algorithm that uses nearest neighbors in the full gene space to identify important genes that may influence due to interactions with other genes. It also has a pseudo p-value for statistical significance. **9.** Install and run NPDR and show the top genes with false discovery rate adjusted p-value less than .05.

```
# for devtools, install Rtools (Windows program) and restart Rstudio
install.packages("devtools")
library(devtools)
install_github("insilico/npdr")


library(npdr)  #https://github.com/insilico/npdr
SxG.dat <- t(GxS.covfilter)
npdr.MDD.results <- npdr(pheno.factor, SxG.dat,
                         regression.type="binomial",
                         attr.diff.type="numeric-abs",
                         nbd.metric = "manhattan",
                         knn=knnSURF.balanced(pheno.factor, sd.frac = .5),
                         dopar.nn = F, dopar.reg=F,
                         padj.method="bonferroni", verbose = T)
colnames(npdr.MDD.results)  # column names of the output

library(dplyr)
# gets genes (attributes/att) with FDR-adjusted p-value<.05
top.p05.npdr <- npdr.MDD.results %>% filter(pval.adj<.05) %>% pull(att)
```

9.

```
[1] "PRPF6"    "ZDHHC20"  "NPFF"     "CREG1"    "UBD"      "WDYHV1"   "GPBAR1"
"TAB3"     "ATXN7L2"  "BANF2"
[11] "SS18L2"   "ZNF622"   "TRAM1"    "NFATC3"   "DLEC1"    "RBBP5"
"ORMDL2"   "RAB25"    "PRDM5"    "CLK2P"
[21] "C6orf118" "PANX1"    "KLHL22"   "EXOC2"    "HNRNPUL2" "HNRNPDL"
"CBL"      "BCL2L12"  "LTB4R2"   "ARID2"
[31] "ANGPT1"   "HMGN3-AS1" "N4BP2L2"  "DNM1P35"  "TSPAN15"  "UBA6-AS1"
"STIM2"    "SRFBP1"   "SLC35B4"  "GOLGA6A"
[41] "SRSF5"    "DYNC2H1"  "ARFGAP1"  "CIAPIN1"  "LOC148413" "VPS33A"
"BRD3"     "EXOSC2"   "LUC7L2"   "CLIP1-AS1"
[51] "STK38L"   "IMMP1L"   "ZCWPW1"   "GNL3"     "H3F3C"    "VMAC"
"KPNA2"    "SRSF12"   "MITF"     "MDGA1"
[61] "MAPK8IP1" "HNRNPA3P1" "NGRN"     "MFSD12"
```

10. Use the code below to print the top 200 genes and see what pathways are enriched (MSigDB Reactome web service). What are the top pathways? **In MSigDB, change "with FDR q-value less than" from .05 to 1.**

```
# grab top 200, remove NA, remove "", get att col
top.npdr <- npdr.MDD.results %>% dplyr::slice(1:200) %>%
  filter(att!="") %>% pull(att)

write.table(top.npdr,row.names=F,col.names=F,quote=F)
```

10. top.npdr <- npdr.MDD.results %>% dplyr::slice(1:200) %>%
filter(pval.adj<1) %>% pull(att)

PRPF6
ZDHHC20
NPFF
CREG1
UBD
WDYHV1
GPBAR1
TAB3
ATXN7L2
BANF2
SS18L2
ZNF622
TRAM1
NFATC3
DLEC1
RBBP5
ORMDL2
RAB25
PRDM5
CLK2P
C6orf118
PANX1
KLHL22
EXOC2
HNRNPUL2
HNRNPDL
CBL
BCL2L12
LTB4R2
ARID2

ANGPT1
HMGN3-AS1
N4BP2L2
DNM1P35
TSPAN15
UBA6-AS1
STIM2
SRFBP1
SLC35B4
GOLGA6A
SRSF5
DYNC2H1
ARFGAP1
CIAPIN1
LOC148413
VPS33A
BRD3
EXOSC2
LUC7L2
CLIP1-AS1
STK38L
IMMP1L
ZCWPW1
GNL3
H3F3C
VMAC
KPNA2
SRSF12
MITF
MDGA1
MAPK8IP1
HNRNPA3P1
NGRN
MFSD12
MRPS6
ITM2C
TUSC3
SMARCC2
LEPROTL1

ZMYM6NB
ARL6IP6
TATDN3
DGKE
INO80B
NR1H2
FAM193A
WASF3
TAF1D
ZNF347
EYA4
RBM5-AS1
ZCCHC9
DNAJC21
MAPKAPK2
FOXRED2
EGLN2
NCOA7
TBC1D15
IQSEC2
ANKRD42
TBCD
TCF21
TBC1D7
RAB11B
ZSWIM8
MCTS2P
SREK1IP1
MLLT1
EIF2D
SRSF10
RNF31
YAP1
AHSG
RPAP2
OIP5
CRX
C2orf15
TNFAIP8L2

PRPF31
ZNF658
TARM1
METAP1D
SLC3A1
ADIRF
AVP
SUPT5H
ZBTB11-AS1
U2AF1L4
KAT6B
RPL36
SNX29
NDUFB1
ZNF610
SAP30L
CCAR2
JOSD2
TRAF4
TMEM167A
CAMSAP2
LENG9
FEM1A
AP1G2
WI2-2373I1.2
MKRN1
RAB11FIP5
HIRIP3
COQ2