**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) than the univariate 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 ALL status as C=1 and NoL status as C=0. This makes NoL the reference level (baseline) so the probability means probability of being in the ALL class. The predictor variable is a gene G with an expression level 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?

[in] print(c(b0, b1, b1.pval))

[out] -11.1141253 1.9906868 0.1535384

intercept = -11.1141253

slope = 1.9906868

pvalue of slope = 0.1535384 or 0.154 after rounding

pvalue of intercept = 0.153

pvalue is high so it isn’t 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]

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.

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

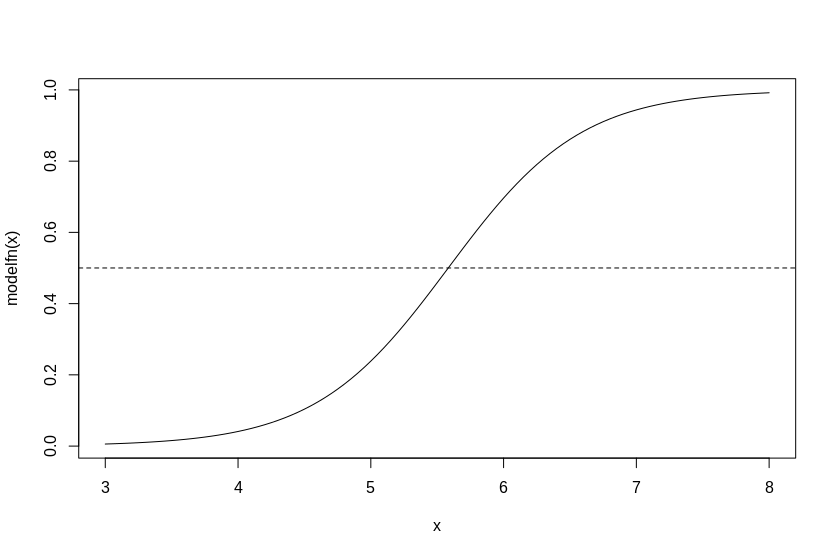
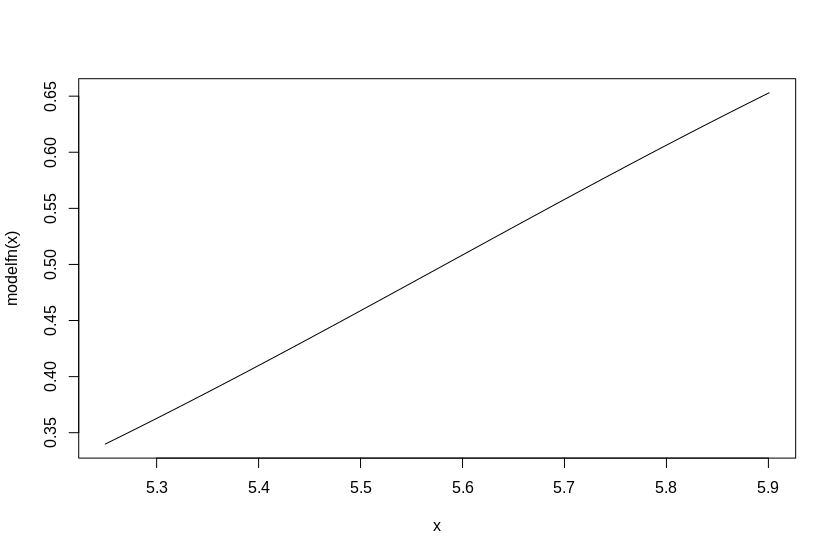
abline(h=.5, lty=2)

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.



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,

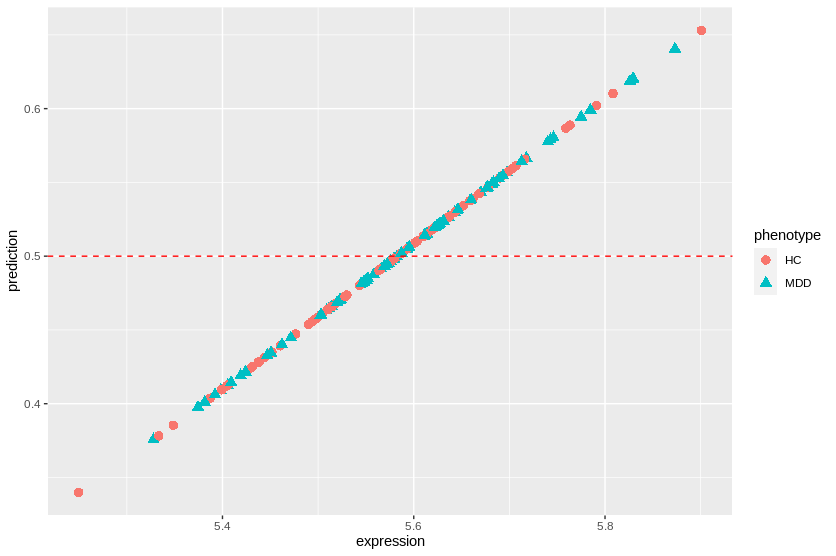
color=phenotype, # shape and color

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?

the rule is if the probability of belonging to MDD >=.5 then it is classified as MDD. The probability is calculated using a sigmoidal function with an input of the linear regression model that has the coefficients listed above and gene.expr as the input to the linear model.

The plot above shows the probability of the given input belonging to MDD. So by simply looking at the plot we see the predictions are almost random. It very bad

[in] print(sum(predicted.class==1))

[out] 76

76 predicted as MDD

[in] print(sum(correct.class==TRUE))

[out] 85

85 correctly classified, 72 incorrectly classified.

**4.** Add code to calculate the classification accuracy (the number of correctly classified samples divided by the number of samples). Does this gene lead to a good classifier? (As a reference, a particularly bad classifier would be flipping a coin, which would have 50% accuracy on average).

[in] print(sum(correct.class==TRUE)/length(correct.class))

[out] 0.5414013

pardon my french but the model is heckin’ bad. It barely is better than random. This literally could happen from a random classifier.

# 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.class <- predicted.class == pheno.factor.relevel

**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)

[out] "AAK1" "1.99068675569479" "0.15353836175625"

which is the gene name AAK1, the slope, and the p value of the slope from the linear model we previously made

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,]

[out]

gene b1 p.val

1 MDGA1 3.27222557173125 4.917554e-05

2 ZDHHC20 3.44752422025746 1.664413e-04

3 NPFF -4.93598819959303 2.931886e-04

4 IRF2BPL 4.02807033481585 3.994416e-04

5 ARFGAP1 4.50243248968132 4.083146e-04

6 FAM138A -3.90725563811714 4.308357e-04

7 UBD 2.02594802661092 4.757671e-04

8 BCL2L12 3.55042230110902 6.054376e-04

9 POGZ -3.36092614792742 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:

(Ggene.row <- which(rownamesxS.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**

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)

**# Add code for accuracy**

predicted.class <- as.integer(predicted.probs >=.5) # predicted class

pheno.factor.relevel # true class

# vector of True (correctly predicted) and False (wrongly predicted)

correct.class <- predicted.class == pheno.factor.relevel

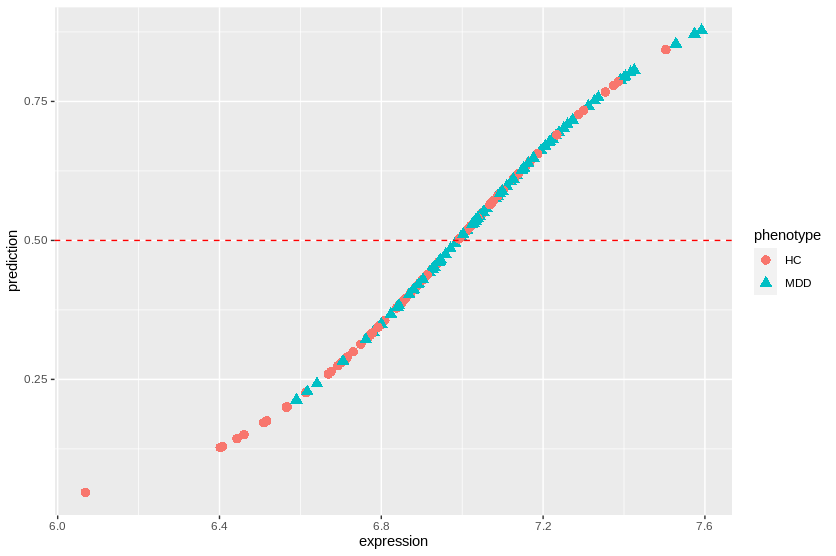
print(sum(predicted.class==1))

print(sum(correct.class==TRUE))

print(sum(correct.class==TRUE)/length(correct.class))

correct: 98

accuracy: 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

glmnet.model <- cv.glmnet(t(GxS.covfilter), pheno.factor.relevel, alpha=.1,

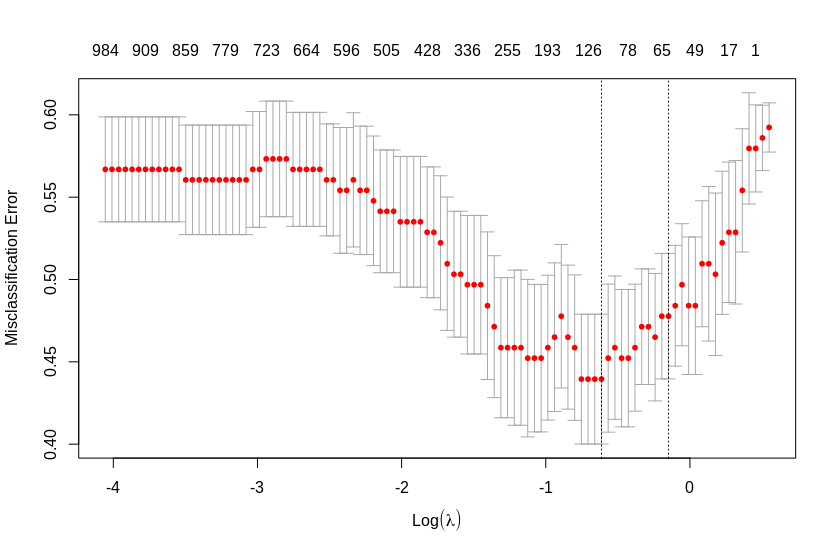
family="binomial", type.measure="class")

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.

About 126 is the best lambda to minimize classification error.



# get the penalized regression coefficients

glmnet.coeffs <- predict(glmnet.model,type="coefficients")

# 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

# 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

Non-zero coef:

[1] "(Intercept)" "ABCE1" "AHCTF1P1" "ARAP1" "ARFGAP1" "ATXN7L2" "BCL2L12" "C16orf74" "C1orf35"

[10] "CBL" "CD83" "CDC123" "CDC14B" "CSPG4" "DNM1P35" "DPEP3" "DYNLRB1" "EARS2"

[19] "ELP3" "EXOSC3" "FAM129B" "FAM138A" "FAM193A" "FKBP5" "FOXRED2" "GSC" "HMGN2P46"

[28] "ICOS" "IFI16" "IRF2BPL" "KANTR" "KLHL22" "LOC375196" "LRRC37BP1" "MARK4" "MDGA1"

[37] "MEN1" "METAP1D" "NDUFB1" "NPFF" "OSBPL3" "PANX1" "POGZ" "PPP2R5C" "RASA1"

[46] "RCSD1" "SRSF5" "SS18L2" "STX16-NPEPL1" "SUMO3" "SUPT7L" "TOMM34" "TSR3" "TUSC3"

[55] "UBD" "UBR2" "UPK1A" "VDAC3" "ZBTB11-AS1" "ZCWPW1" "ZDHHC20" "ZFP36L2" "ZMYM6NB"

accuracy: 0.8535032

E. 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.

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.method="multisurf",

nbd.metric = "manhattan",

msurf.sd.frac=.5, k=0,

neighbor.sampling="none",

dopar.nn = F, dopar.reg = T,

padj.method="bonferroni", verbose=F)

colnames(npdr.MDD.results)

library(dplyr)

# attributes with FDR-adjusted p-value<.05

top.p05.npdr <- npdr.MDD.results %>% filter(pval.adj<.05) %>% pull(att)

"PRPF6" "ZDHHC20" "NPFF" "CREG1" "UBD" "WDYHV1" "GPBAR1" "TAB3" "ATXN7L2" "BANF2" "SS18L2"

[12] "ZNF622" "TRAM1" "NFATC3" "DLEC1" "RBBP5" "ORMDL2" "RAB25" "PRDM5" "CLK2P" "C6orf118" "PANX1"

[23] "KLHL22" "EXOC2" "HNRNPUL2" "HNRNPDL" "CBL" "BCL2L12" "LTB4R2" "ARID2" "ANGPT1" "HMGN3-AS1" "N4BP2L2"

[34] "DNM1P35" "TSPAN15" "UBA6-AS1" "STIM2" "SRFBP1" "SLC35B4" "GOLGA6A" "SRSF5" "DYNC2H1" "ARFGAP1" "CIAPIN1"

[45] "LOC148413" "VPS33A" "BRD3" "EXOSC2" "LUC7L2" "CLIP1-AS1" "STK38L" "IMMP1L" "ZCWPW1" "GNL3" "H3F3C"

[56] "VMAC" "KPNA2" "SRSF12" "MITF" "MDGA1" "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) %>%

drop\_na(att) %>% filter(att!="") %>% pull(att)

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

[in] top.1.npdr <- npdr.MDD.results %>% filter(pval.adj<1) %>% pull(att)

top.npdr <- top.1.npdr %>% dplyr::slice(1:200) %>%

drop\_na(att) %>% filter(att!="") %>% pull(att)

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

[out]

**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**

**EXD1**

**RAET1E**

**STK4-AS1**

**WDR78**

**EIF1AD**

**DPEP3**

**POLR3E**

**POGZ**

**ABHD4**

**MRFAP1L1**

**SUPT7L**

**NUP88**

**SNX29P1**

**NHLRC4**

**TRIM28**

**BRS3**

**ERCC3**

**SUMO4**

**EPCAM**

**PCYT1A**

**SSH2**

**ARRDC2**

**RCSD1**

**DNAJB7**

**TBC1D23**

**POLD4**

**HERC5**

**ARHGEF19**

**PBX3**

**ZRANB2-AS1**

**AMD1**

**FOXN3-AS1**

**COX7B2**

**APH1B**

**TSR3**

**PMPCB**

**RGL3**

**RPL12**

**TAF6**

**TECR**

**WAC-AS1**

**FAM201A**

**SZT2**

**EARS2**

**DDX3Y**

**KCNA4**

**MEX3D**

**CREBL2**

**ARL5B**

**NSDHL**

**TTC1**

**MAN1B1**

**C1orf53**

**YTHDF1**

**CSNK2B**

**PSMG3**

**MED13L**

**ST3GAL3**

**NKX1-1**

**MSL3P1**

**USP15**

**CYB561D1**

**SEC14L1P1PRPF6**

**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**

**EXD1**

**RAET1E**

**STK4-AS1**

**WDR78**

**EIF1AD**

**DPEP3**

**POLR3E**

**POGZ**

**ABHD4**

**MRFAP1L1**

**SUPT7L**

**NUP88**

**SNX29P1**

**NHLRC4**

**TRIM28**

**BRS3**

**ERCC3**

**SUMO4**

**EPCAM**

**PCYT1A**

**SSH2**

**ARRDC2**

**RCSD1**

**DNAJB7**

**TBC1D23**

**POLD4**

**HERC5**

**ARHGEF19**

**PBX3**

**ZRANB2-AS1**

**AMD1**

**FOXN3-AS1**

**COX7B2**

**APH1B**

**TSR3**

**PMPCB**

**RGL3**

**RPL12**

**TAF6**

**TECR**

**WAC-AS1**

**FAM201A**

**SZT2**

**EARS2**

**DDX3Y**

**KCNA4**

**MEX3D**

**CREBL2**

**ARL5B**

**NSDHL**

**TTC1**

**MAN1B1**

**C1orf53**

**YTHDF1**

**CSNK2B**

**PSMG3**

**MED13L**

**ST3GAL3**

**NKX1-1**

**MSL3P1**

**USP15**

**CYB561D1**

**SEC14L1P1**