MLR

Brian Dehlinger

December 1, 2019

Original Model Train Test Split, Boxplot

```r
set.seed(10035)
remove_singularities <- function(dataset, gene_name){
  dataset_copy <- dataset
  item <- paste(gene_name, "~.", sep="")
  full_formula <- as.formula(item)
  fit <- lm(full_formula, data=dataset)
  singularities <- attributes(alias(fit)$Complete)$dimnames[[1]]
  for (singularity in singularities){
    dataset_copy[singularity] <- NULL
  }
  return(dataset_copy)
}
read_in_pruned_datasets_for_gene_0.8 <- function(gene_name, path){
  full_path0.8 <- paste(path, gene_name, "_for_r_0.8.txt", sep="")
  Data0.8 <- read.table(full_path0.8, header=TRUE, sep=',')
  Data0.8 <- remove_singularities(Data0.8, gene_name)
  return(Data0.8)
}
#install.packages("DAAG")
#install.packages("caret")
#install.packages("lmtest")
#install.packages("MASS")
#install.packages("car")
#install.packages("reshape")
#install.packages("plotmo")
#install.packages("olsrr")
library(caret)
```

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

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

```r
library(lmtest)
```

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

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

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

```r
library(MASS)
library(car)
```

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

```r
library(reshape)
library(plotmo)
```

```
## Loading required package: Formula
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```r
library(olsrr)
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:MASS':
##
##     cement
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

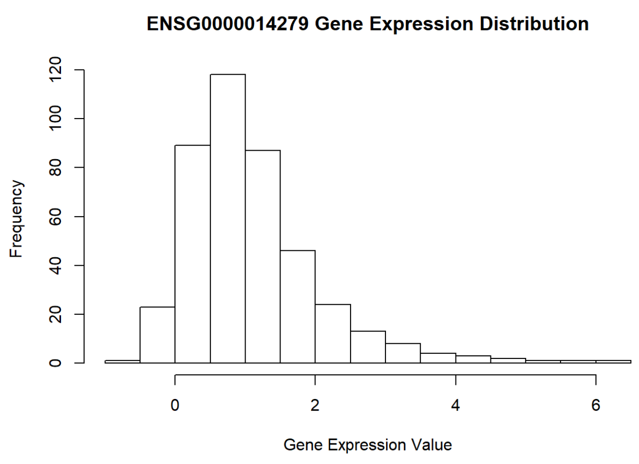
```r
library(DAAG)
```

```
##
## Attaching package: 'DAAG'
```

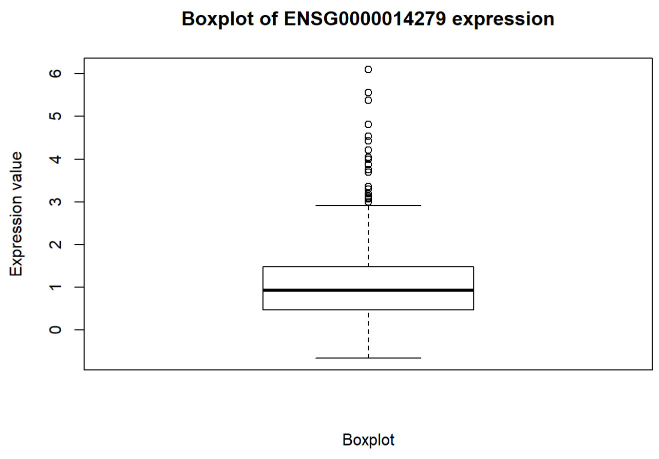
```
## The following object is masked from 'package:car':
##
##     vif
```

```
## The following object is masked from 'package:MASS':
##
##     hills
```

```r
gene_data <- read_in_pruned_datasets_for_gene_0.8("ENSG00000142794", "D:\\Project\\GitStash\\Applied_regression_pr
oject\\")
gene_data <- as.data.frame(gene_data)
hist(gene_data[["ENSG00000142794"]], main="ENSG0000014279 Gene Expression Distribution", ylab="Frequency", xlab="
Gene Expression Value")
```



```
boxplot(gene_data[["ENSG00000142794"]], main="Boxplot of ENSG0000014279 expression", ylab="Expression value", xlab
="Boxplot")
```



```
# We notice that the Expression Data is skewed to the right.
trainIndex <- createDataPartition(gene_data[["ENSG00000142794"]], p=.8, list = FALSE, times=1)

# We do test train split and explicitly create the data partition indexs
# because of the difference seed values produce in different R versions
```

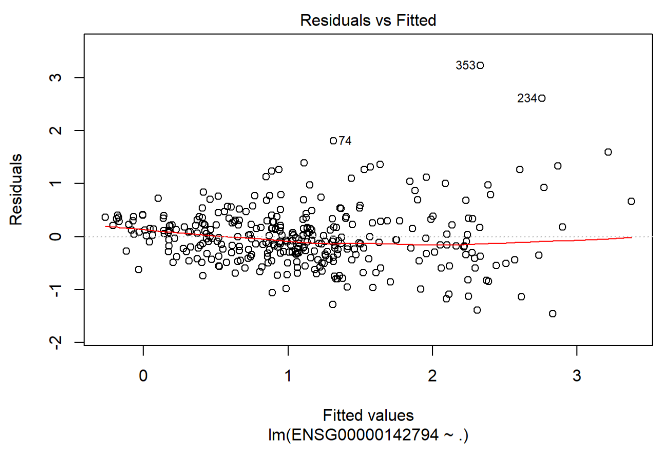
```
gene_data <- gene_data[train_index,]
gene_test <- gene_data[-train_index,]
#We do a train validation split here.
```

First Model Diagnostics

```
model <- lm(ENSG00000142794~., data=gene_data)
summary(model)
```

```
##
## Call:
## lm(formula = ENSG00000142794 ~ ., data = gene_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4584 -0.3311 -0.0545  0.2781  3.2295
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.0773438  0.8375846   1.286   0.1996
## rs12734589  -0.2426399  0.3650244  -0.665   0.5069
## rs2004380    0.1102502  1.0093512   0.109   0.9131
## rs10737456  -0.1014757  0.2853902  -0.356   0.7225
## rs10799692   0.0890538  0.5131390   0.174   0.8624
## rs3820296   -0.2600382  0.3517204  -0.739   0.4604
## rs10916988   0.1944314  0.1744793   1.114   0.2663
## rs1976403    0.5034704  0.4112980   1.224   0.2222
## rs1780324    0.4284627  0.2894724   1.480   0.1402
## rs4654753   -0.4658774  0.4593801  -1.014   0.3116
## rs79760554   0.0278784  0.1362679   0.205   0.8381
## rs12729540  -0.0386049  0.2499062  -0.154   0.8774
## rs12043777  -0.1137017  0.1445647  -0.787   0.4324
## rs10799702   0.1519789  0.2212533   0.687   0.4928
## rs4061838   -0.1683068  0.1846747  -0.911   0.3630
## rs7545635    0.0693701  0.2444405   0.284   0.7768
## rs4654946   -0.0593146  0.5275433  -0.112   0.9106
## rs2047653   -0.0215676  0.4805840  -0.045   0.9642
## rs4079441    0.0160076  0.5254259   0.030   0.9757
## rs12034222   0.1939562  0.1212573   1.600   0.1111
## rs3820292   -0.0718418  0.1268148  -0.567   0.5716
## rs75417796   0.1798876  0.3004644   0.599   0.5500
## rs7541779    0.0238033  0.2965717   0.080   0.9361
## rs75417790   0.0475355  0.2394576   0.199   0.8428
## rs10799691  -0.0251948  0.2950647  -0.085   0.9320
## rs1971328    0.2014015  0.2281684   0.883   0.3783
## rs12744514   0.6974904  0.5034685   1.385   0.1673
## rs41310392  -0.4924510  0.3108441  -1.584   0.1145
## rs75056920   0.0976996  0.1316812   0.742   0.4589
## rs1697421   -0.2674773  0.2314293  -1.156   0.2490
## rs146767219  0.0388611  0.1074221   0.362   0.7179
## rs12132412   0.0488435  0.1550308   0.315   0.7530
## rs112985962 -0.1747530  0.6483509  -0.270   0.7878
## rs147900768  0.1793372  0.2227159   0.805   0.4215
## rs1814737   -0.6478764  0.7304950  -0.887   0.3760
## rs56286426  -0.0108865  0.1338131  -0.081   0.9352
## rs17420195   1.2040546  0.9058701   1.329   0.1851
## rs113548640  0.1619209  0.3515875   0.461   0.6456
## rs146564096 -0.1459169  0.1311852  -1.112   0.2672
## rs10916989   0.0113086  0.2306912   0.049   0.9609
## rs150401191  0.0091174  0.6004537   0.015   0.9879
## rs3010180   -0.1346392  0.1276371  -1.055   0.2926
## rs78885464   0.1262377  0.1141166   1.106   0.2698
## rs6694671    0.0356021  0.3917338   0.091   0.9277
## rs144522615 -0.0696422  0.1296253  -0.537   0.5916
## rs61775934  -0.1628478  0.4170355  -0.390   0.6965
## rs3010181    0.1408603  0.1079256   1.305   0.1931
## rs1780323   -0.0641018  0.1257974  -0.510   0.6108
## rs61778366  -0.3494258  0.2960935  -1.180   0.2392
## rs79659018   0.0159360  0.1274369   0.125   0.9006
## rs10737458  -0.3630167  0.4858445  -0.747   0.4557
## rs11584744   0.5002325  0.3351833   1.492   0.1369
## rs201590042 -0.0149337  0.1038923  -0.144   0.8858
## rs1566524   -0.1873926  0.4552565  -0.412   0.6810
## rs61775950   0.0831508  0.0978564   0.850   0.3964
## rs144443608  0.0877285  0.1013082   0.866   0.3874
## rs10916986   0.3573069  0.2995042   1.193   0.2341
## rs904927     0.1363907  0.3373079   0.404   0.6863
## rs190849739  0.0054703  0.1267022   0.043   0.9656
## rs143561157  0.1553284  0.1161436   1.337   0.1824
## rs12740648  -0.0110182  0.1509227  -0.073   0.9419
## rs12118362   0.0390582  0.1495256   0.261   0.7942
## rs1809914   -0.4849652  0.2207155  -2.197   0.0290 *
## rs41310412   0.0082098  0.2822820   0.029   0.9768
## rs6423191    0.0007244  0.1983363   0.004   0.9971
## rs140609058 -0.2737547  0.3502827  -0.782   0.4353
## rs7555005   -0.1160461  0.2698248  -0.430   0.6675
## rs71512991  -0.0922331  0.1062848  -0.868   0.3864
## rs139043162 -0.0988816  0.3860193  -0.256   0.7981
## rs2800935   -0.0890408  0.1292364  -0.689   0.4915
## rs113324018  0.1567611  0.2392068   0.655   0.5129
## rs904928     0.0440956  0.1553133   0.284   0.7767
## rs7533048    0.1191272  0.2959662   0.403   0.6877
## rs12239666  -0.1800842  0.2227685  -0.808   0.4197
## rs972662     0.1707553  0.2390314   0.714   0.4757
## rs9726624   -0.0631679  0.1772911  -0.356   0.7219
## rs2800774    0.0266631  0.0939158   0.284   0.7767
## rs56268398   0.0605925  0.1983985   0.305   0.7603
## rs12082914   0.1894957  0.3400845   0.557   0.5779
## rs78566499   0.0775510  0.2753531   0.282   0.7785
## rs1809915    0.2982296  0.1990040   1.499   0.1353
## rs7547671    0.2785216  0.8860444   0.314   0.7535
## rs146549873 -0.0763266  0.0938884  -0.813   0.4171
## rs113934925 -0.1689566  0.9265713  -0.182   0.8555
## rs200384063 -0.0353196  0.2725797  -0.130   0.8970
## rs4654930    0.3933332  0.3124555   1.259   0.2093
## rs3855556    0.0959987  0.1545367   0.621   0.5351
## rs35836191  -0.6655432  0.5295844  -1.257   0.2101
## rs60803995   0.1947940  0.1536928   1.267   0.2063
## rs12037596  -0.1170011  0.1806859  -0.648   0.5179
## rs2651402    0.4172604  0.3110401   1.342   0.1811
## rs41290414  -0.3163552  0.4893582  -0.646   0.5186
## rs72476502   0.0790231  0.2649570   0.298   0.7658
## rs58090121  -0.0165627  0.1430192  -0.116   0.9079
## rs904937    -0.1079859  0.1691946  -0.638   0.5239
## rs41265985  -0.3857862  0.2618278  -1.473   0.1420
## rs77025042  -0.0157810  0.1191521  -0.132   0.8947
## rs7417849   -0.1141053  0.1663477  -0.686   0.4934
## rs113558389 -0.1506948  0.2873036  -0.525   0.6004
## rs904929     0.1129374  0.1973569   0.572   0.5677
## rs10799704  -0.0001877  0.0840813  -0.002   0.9982
## rs148061397  0.0406450  0.2051900   0.198   0.8432
## rs115933091  0.1055186  0.3592554   0.294   0.7692
## rs143553186  0.5298658  0.2947059   1.798   0.0735 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6711 on 233 degrees of freedom
## Multiple R-squared:  0.6148, Adjusted R-squared:  0.4446
## F-statistic: 3.611 on 103 and 233 DF,  p-value: 2.972e-16
```

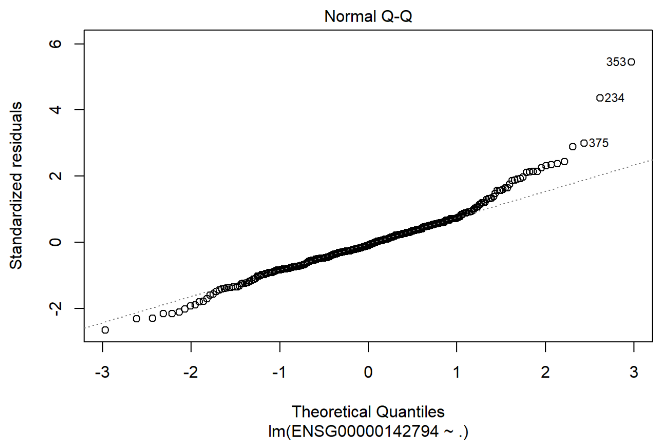
```
plot(model, which=1)
```



```
bptest(model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model
## BP = 96.199, df = 103, p-value = 0.6693
```

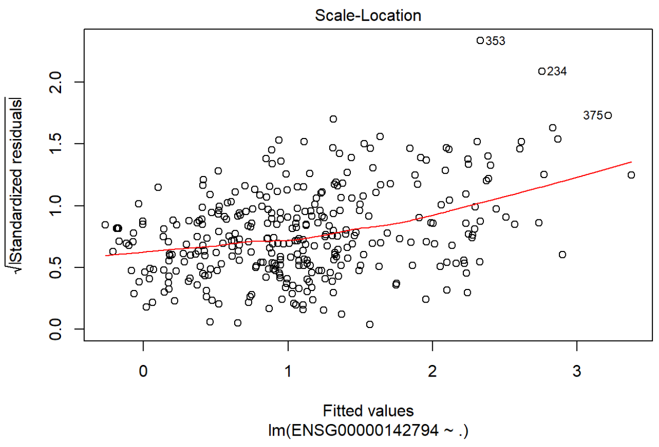
```
# We note from the Residuals Vs Fitted Plot that there
#might be a slight concern with nonconstant variance but for the Breusch-Pagan test
# we fail to reject with an alpha of 0.05.

plot(model, which=2)
```



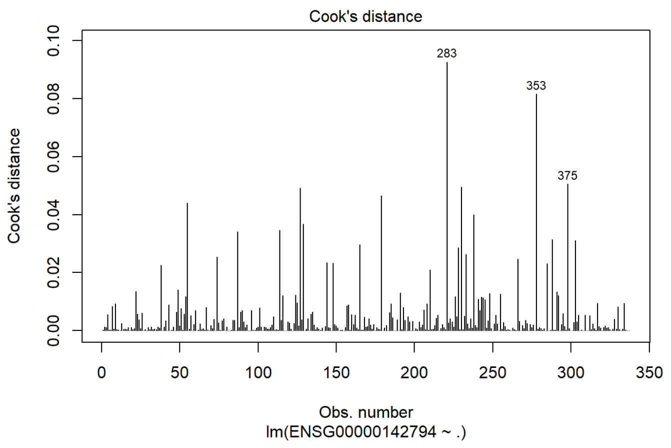
```
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.94027, p-value = 2.08e-10
```

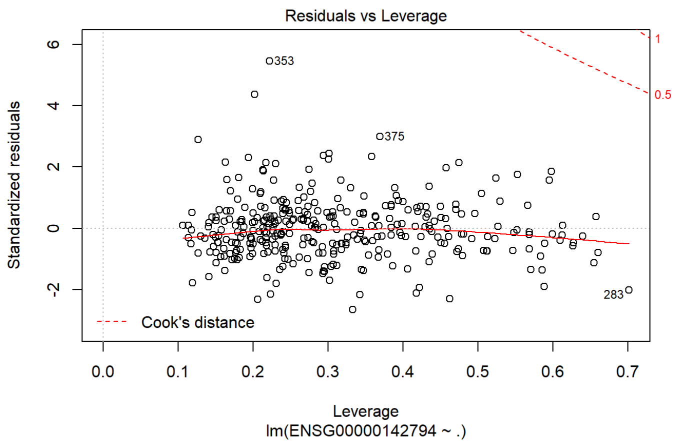
```
# However, we notice an issue with normality from a QQPlot that
# suggests the data is skewed and we reject the shapiro-wilks test with an alpha of 0.05

plot(model, which=3)
```



```
#The Scale-Location plot has a slope most likely because there isn't enough data for the fitted values.

plot(model, which=4)
```



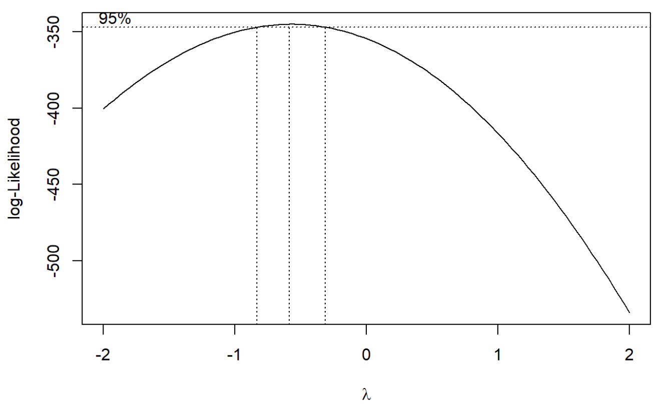
```
plot(model, which=5)
```



```
# We don't notice an issues with overly influential points in the residuals vs Leverage Plot.
# However, we do note some points are considered to have high leverage later on. But these points
# are actually important to the variation we want to capture in our data.

summary(model)$adj.r.squared
```

```
## [1] 0.444552
```

```
# We have an adjusted R-squared of 0.44452.

test <- model <- lm(ENSG00000142794+2~., data=gene_data)
# We do a Boxcox on the model and note that we should do a transformation on Y(1/sqrt(Y+2))
boxcox(test)
```



VIFs of Original Model

```
# VIF Calculation and setting N and P
n <- nrow(gene_data)
p <- length(model$coefficients)

# There is an indication of severe multicolinearity.
car::vif(model)
```

```
##   rs12734589    rs2004380   rs10737456   rs10799692    rs3820296   rs10916988
##    53.938076   396.667047    29.989972    98.776324    41.200683    11.627147
##    rs1976403    rs1780324    rs4654753   rs79760554   rs12729540   rs12043777
##    59.105285    32.812037    70.967745     5.029291    22.203345     8.439732
##   rs10799702    rs4061838    rs7545635    rs4654946    rs2047653    rs4079441
##    18.192176    12.458436    21.141962    85.906720    73.463714    89.795592
##   rs12034222    rs3820292   rs75417796    rs7541779   rs75417790   rs10799691
##     6.065616     6.054684     1.756140    27.444465     1.846154    27.469175
##    rs1971328   rs12744514   rs41310392   rs75056920    rs1697421  rs146767219
##    15.900064   103.196154    40.269403     7.224826    23.143299     3.799889
##   rs12132412  rs112985962  rs147900768    rs1814737   rs56286426   rs17420195
##     8.346162   113.962670     8.909498   131.440962     7.271876   199.760324
##  rs113548640  rs146564096   rs10916989  rs150401191    rs3010180   rs78885464
##    30.700811     5.847086    19.261110    94.030738     8.797080     4.748219
##    rs6694671  rs144522615   rs61775934    rs3010181    rs1780323   rs61778366
##    46.954267     8.112206    53.300421     6.207780     4.287989    20.657987
##   rs79659018   rs10737458   rs11584744  rs201590042    rs1566524   rs61775950
##     8.109858    65.126021    31.466721     3.403908    77.820080     3.513458
##  rs144443608   rs10916986     rs904927  rs190849739  rs143561157   rs12740648
##     2.019597    33.746042    28.487079     9.052471     6.058666     4.889190
##   rs12118362    rs1809914   rs41310412    rs6423191  rs140609058    rs7555005
##     5.767630    14.932888     7.027638    17.147129    17.061526    27.146524
##   rs71512991  rs139043162    rs2800935  rs113324018     rs904928    rs7533048
##     2.913536    16.995030     8.307933    11.325202     4.666285    17.181528
##   rs12239666     rs972662    rs9726624    rs2800774   rs56268398   rs12082914
##    14.776399    13.066115     2.152436     2.108870     7.648545    30.252107
##   rs78566499    rs1809915    rs7547671  rs146549873  rs113934925  rs200384063
##     9.431760    13.113966   240.217181     3.809010   144.059295     4.097976
##    rs4654930    rs3855556   rs35836191   rs60803995   rs12037596    rs2651402
##    19.221395     8.404156    35.165796     8.655057    12.487565     1.881941
##   rs41290414   rs72476502   rs58090121     rs904937   rs41265985   rs77025042
##    37.250580     9.801619     3.754337     9.915960    12.999266     1.703049
##    rs7417849  rs113558389     rs904929   rs10799704  rs148061397  rs115933091
##     8.494297     8.513513    12.096988     1.916642     1.841093     1.689051
##  rs143553186
##     1.506332
```

```
test <- lm(ENSG00000142794+2~., data=gene_data)
```

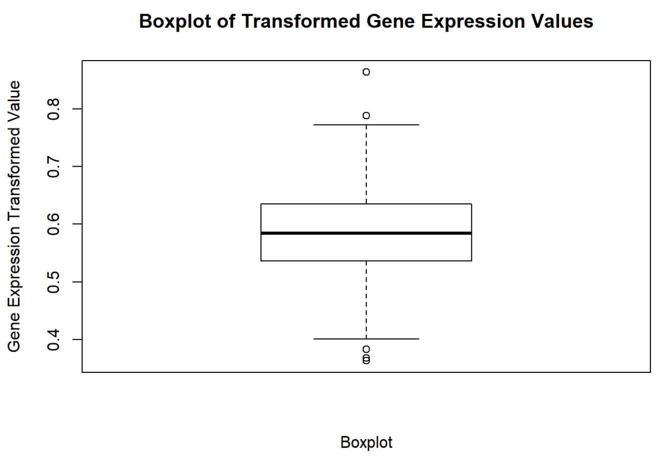
Doing the Added Variable Plots

```
# The added variable plots suggest a lot of the predictors do not
# add new information when the other predictors are included in the model.
pdf("AddedVariablesBeforeSelection.pdf")
avPlots(model, ask=FALSE)
dev.off()
```

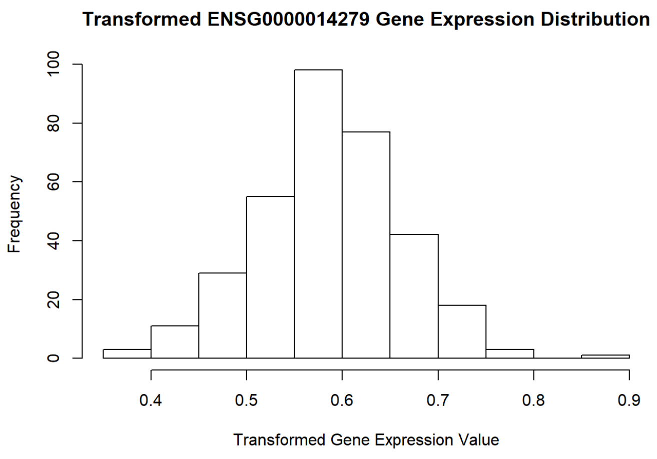
```
## png
##   2
```

Transforming the Data

```
# We do the transform on the train and test datasets. # We also set n.
# Additionally, we also plot the transformed values and note they appear to be normally distributed.
transformed_gene_data <- gene_data
transformed_gene_test <- gene_test
n <- nrow(transformed_gene_data)
transformed_gene_data[["ENSG00000142794"]] <- 1/(sqrt(transformed_gene_data[["ENSG00000142794"]]+2))
transformed_gene_test[["ENSG00000142794"]] <- 1/(sqrt(transformed_gene_test[["ENSG00000142794"]]+2))
boxplot(transformed_gene_data["ENSG00000142794"], main="Boxplot of Transformed Gene Expression Values", ylab="Gene
Expression Transformed Value", xlab="Boxplot")
```



```
hist(transformed_gene_data[["ENSG00000142794"]], main="Transformed ENSG0000014279 Gene Expression Distribution", y
lab="Frequency", xlab="Transformed Gene Expression Value")
```



# Model Selection Process

```
# We do variable selection and we do a strict variable selection with a p-value of 0.01. Step-Forward BIC, and Ste
p-Backward BIC.
# We then put the selected variables for each model together and do a best subset exhaustive search over just thes
e variables.

naieve_model <- lm(ENSG00000142794~., data=transformed_gene_data)
naieve_model_empty <- lm(ENSG00000142794~1, data=transformed_gene_data)


selected_p <- ols_step_backward_p(naieve_model, prem=0.01)
selected_p_model <- selected_p$model
p <- length(selected_p_model$coefficients)
RSS_selected <- c(crossprod(selected_p_model$residuals))
MSE <- RSS_selected / length(selected_p_model$residuals)
selected_p_stats <- rbind(sqrt(MSE), BIC(selected_p_model), summary(selected_p_model)$adj.r.squared, DAAG::press(s
elected_p_model))




selected<- stepAIC(naieve_model, k=log(n))
p <- length(selected$coefficients)
RSS_selected <- c(crossprod(selected$residuals))
MSE <- RSS_selected / length(selected$residuals)
selected_stats <- rbind(sqrt(MSE), BIC(selected), summary(selected)$adj.r.squared, DAAG::press(selected))

forward_bic <- stepAIC(naieve_model_empty, scope = list(upper=naieve_model, lower=naieve_model_empty), direction="
forward", k=log(n))
p <- length(forward_bic$coefficients)
RSS_selected <- c(crossprod(forward_bic$residuals))
MSE <- RSS_selected / length(forward_bic$residuals)
forward_bic_stats <- rbind(sqrt(MSE), BIC(forward_bic), summary(forward_bic)$adj.r.squared, DAAG::press(forward_bi
c))
```

VIF Stats of Selected Models

```
car::vif(selected)
```

```
##  rs12734589  rs10799692 rs113548640   rs6694671   rs1566524  rs35836191 
##    5.036327    4.997166    2.697843    3.118706    3.069425    1.694852 
##  rs60803995 
##    2.351035
```

```
car::vif(selected_p_model)
```

```
##  rs12734589 rs113548640   rs6694671   rs1566524  rs35836191  rs60803995 
##    3.283511    2.146153    2.669397    3.068997    1.694835    2.347506
```

```
car::vif(forward_bic)
```

```
##  rs12734589   rs1976403   rs4654753   rs6694671 rs113548640  rs10916989 
##    3.192661    2.848736    4.706859    3.299466    5.526336    3.981726
```

```
# We now do not see any issues with multicolinearity which suggests
# we do not need to use ridge regression on these reduced variable models.
```

Selected Model Choosing

```
search_full_model <- lm(ENSG00000142794~rs12734589 + rs1976403 + rs4654753 + rs6694671 + rs113548640 + rs10916989
+ rs1566524, rs35836191 + rs60803995 + rs10799692, data=transformed_gene_data)

all_possible <- ols_step_best_subset(search_full_model)
all_possible
```

```
##                                 Best Subsets Regression
## ------------------------------------------------------------------------------------------
## Model Index    Predictors
## ------------------------------------------------------------------------------------------
##      1         rs12734589
##      2         rs6694671 rs113548640
##      3         rs12734589 rs6694671 rs113548640
##      4         rs12734589 rs1976403 rs6694671 rs113548640
##      5         rs12734589 rs4654753 rs6694671 rs113548640 rs10916989
##      6         rs12734589 rs1976403 rs4654753 rs6694671 rs113548640 rs10916989
##      7         rs12734589 rs1976403 rs4654753 rs6694671 rs113548640 rs10916989 rs1566524
## ------------------------------------------------------------------------------------------
##
##                                                     Subsets Regression Summary

## ----------------------------------------------------------------------------------------------------------------
-----------------------
##                          Adj.        Pred

## Model    R-Square    R-Square    R-Square       C(p)          AIC        SBIC       SBC         MSEP      FPE
HSP       APC
## ----------------------------------------------------------------------------------------------------------------
-----------------------
##   1        0.4428      0.4412      0.4356     12886.6397     -972.0540      NA      -960.5938    0.0033    0.00
33    0.0000    0.5638
##   2        0.5173      0.5144      0.5077     11122.3098    -1018.3882      NA     -1003.1079    0.0028    0.00
28    0.0000    0.4914
##   3        0.5479      0.5438      0.5357     10397.7557    -1038.4744      NA     -1019.3740    0.0027    0.00
27    0.0000    0.4630
##   4        0.5550      0.5496      0.5398     10231.6838    -1041.7965      NA     -1018.8760    0.0026    0.00
26    0.0000    0.4584
##   5        0.5639      0.5573      0.5477     10021.9716    -1046.6223      NA     -1019.8817    0.0026    0.00
26    0.0000    0.4519
##   6        0.5751      0.5674       0.556      9758.9045    -1053.3680      NA     -1022.8074    0.0026    0.00
26    0.0000    0.4429
##   7        0.5798      0.5709      0.5588      9649.2532    -1055.1209      NA     -1020.7402    0.0025    0.00
25    0.0000    0.4406
## ----------------------------------------------------------------------------------------------------------------
-----------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

```
final_stats <- cbind(selected_p_stats,selected_stats,forward_bic_stats)
row.names(final_stats) <- c("RMSE", "BIC", "ADJ-R-Squared", "PRESS")
colnames(final_stats) <- c("BackwardSelected_By_P_0.01", "Backward_BIC", "ForwardBIC")
knitr::kable(final_stats)
```

	BackwardSelected_By_P_0.01	Backward_BIC	ForwardBIC
RMSE	0.0498458	0.0493727	0.0495121
BIC	-1018.2806629	-1018.8876841	-1022.8073740
ADJ-R-Squared	0.5615122	0.5684880	0.5673627
PRESS	0.8735692	0.8599983	0.8632144

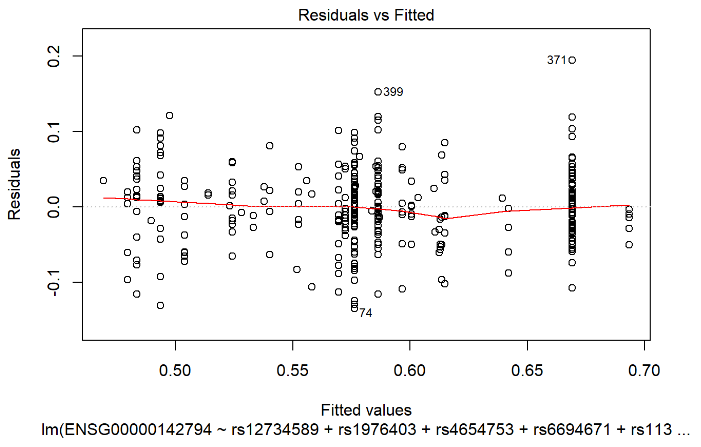
```
# We note that the optimal model is the one selected by forward_bic
# if we consider all the variables all these models give us and their best possible subsets.
# We try to minimize SBC(BIC) and keep a reasonable Adj-R-Squared(>0.55) and MSE.
# We note that MSE is near minmal at 4 predictors. But we have a lower SBC with 6 predictors and a slightly higher
R-squared.
model <- forward_bic
```

Diagnostics of Chosen Model

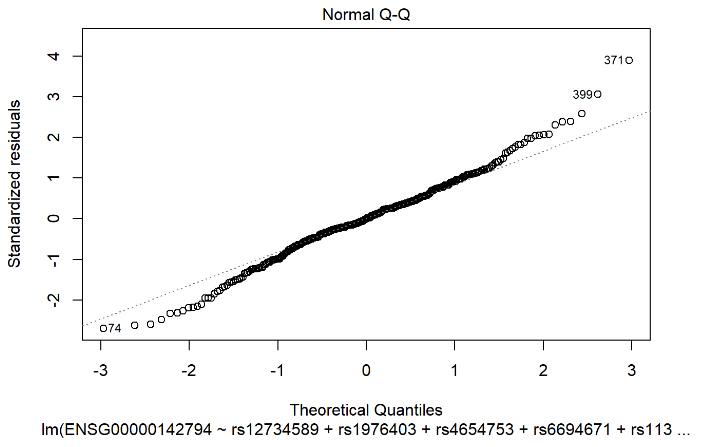
```
# We redo the model diagnostics from the beginning on the selected model.
# We will then compare this to one of the other models in our testing.
summary(model)
```

```
##
## Call:
## lm(formula = ENSG00000142794 ~ rs12734589 + rs1976403 + rs4654753 +
##     rs6694671 + rs113548640 + rs10916989, data = transformed_gene_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.134598 -0.026933  0.000069  0.028130  0.194233
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.524253   0.010447  50.183  < 2e-16 ***
## rs12734589   0.027072   0.006622   4.088 5.46e-05 ***
## rs1976403   -0.019831   0.006733  -2.946 0.003453 **
## rs4654753    0.033980   0.008821   3.852 0.000141 ***
## rs6694671    0.035719   0.007743   4.613 5.68e-06 ***
## rs113548640 -0.036296   0.011122  -3.263 0.001216 **
## rs10916989  -0.024367   0.007821  -3.116 0.001995 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05003 on 330 degrees of freedom
## Multiple R-squared:  0.5751, Adjusted R-squared:  0.5674
## F-statistic: 74.44 on 6 and 330 DF,  p-value: < 2.2e-16
```

```
# All variables appear significant




# We note that the residual vs fitted plot appears to be random.
plot(model, which=1)
```



```
# We notice a small issue with normality still but this can probably be ignored.
# There is no issue with non constant error variance which is more important.
plot(model, which=2)
```



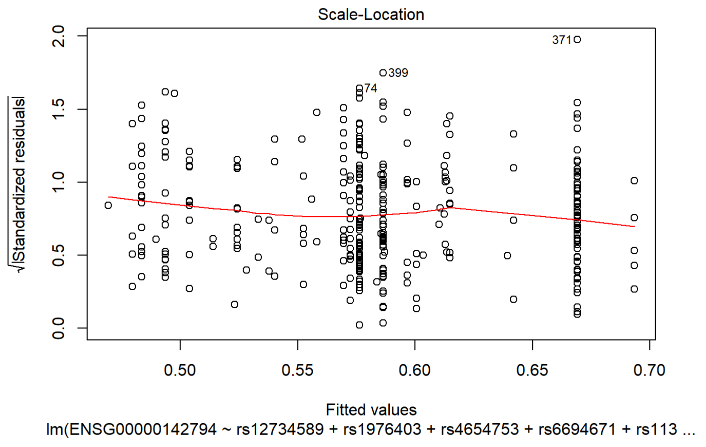
```
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.98986, p-value = 0.01963
```

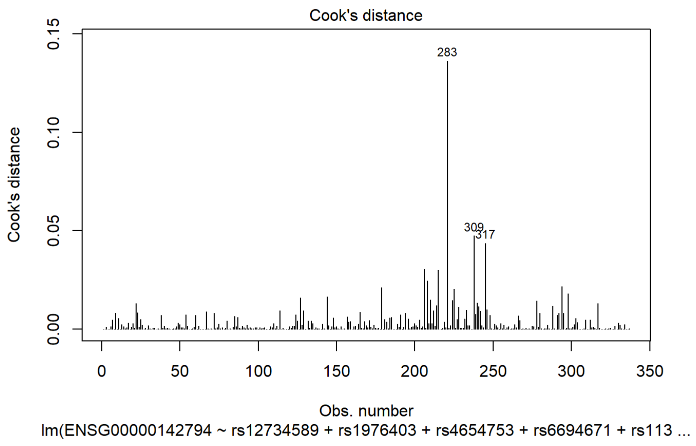
```
bptest(model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model
## BP = 2.7847, df = 6, p-value = 0.8353
```

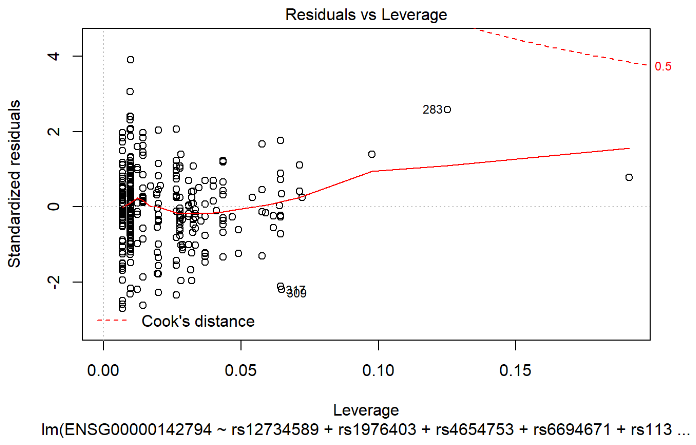
```
# We notice potential outliers in the Cook's Distance Plot and Residuals vs Leverage Plot
# so we will do further analysis on these points.
# Particulary points 283, 309, and 317 have a high cook's distance and are candidate outliers.
# The Scale Location Plot doesn't appear to have a slope which is good.
plot(model, which=3)
```



```
plot(model, which=4)
```



```
plot(model, which=5)
```



```
# The added variable plots look much better.

pdf("AddedVariablesAfterSelection.pdf")
avPlots(model, ask=FALSE)
dev.off()
```

```
## png
##   2
```

```
# All variables appear significant.
# MSE looks very good.
knitr::kable(anova(model))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
rs12734589	1	0.8610066	0.8610066	343.928080	0.0000000
rs1976403	1	0.1251881	0.1251881	50.006228	0.0000000
rs4654753	1	0.0464546	0.0464546	18.556256	0.0000218
rs6694671	1	0.0403177	0.0403177	16.104843	0.0000742
rs113548640	1	0.0208500	0.0208500	8.328506	0.0041598
rs10916989	1	0.0243041	0.0243041	9.708235	0.0019954
Residuals	330	0.8261383	0.0025034	NA	NA
# Sensitivity	Analys	is			

```
# We do outlier analysis here

cd <- cooks.distance(model)
transformed_gene_dropped <- model$model[-c(which(cd > 4 / (n-p))),]
cooks_distance_model <- lm(ENSG00000142794~., data=transformed_gene_dropped)


# We note that we would like to retain these points regardless as they are interesting and represent variation
# in the population that can be informative in clinical situations.
# We opt to not use Robust Regression since the effect on the final model is not serious enough.



leverage_threshold <- (3*(p))/n
leverage_values <- hatvalues(model)
transformed_gene_dropped_leverage <- model$model[-c(which(leverage_values > leverage_threshold)),]
leverage_model <- lm(ENSG00000142794~., data=transformed_gene_dropped_leverage)

betas <- as.data.frame(dfbetas(model))

# We print the largest DFBeta for each parameter and the index of that DFBeta.
to_remove <- c()
for(column in betas){
  outliers <- which(abs(column) > 0.11)
  print(max(abs(column)))
  print(which(abs(column) == max(abs(column))))
  to_remove <- union(to_remove, outliers)
}
```

```
## [1] 0.4216956
## [1] 238
## [1] 0.5156542
## [1] 245
## [1] 0.8014485
## [1] 221
## [1] 0.3773799
## [1] 221
## [1] 0.7861822
## [1] 221
## [1] 0.8713133
## [1] 221
## [1] 0.3917799
## [1] 238
```

```
# One of 238, 245, and 221 give the max DFBetas the coefficients. These are the greatest outliers for DFBetas.
transformed_gene_dropped_betas <- model$model[-c(to_remove),]
new_model_betas <- lm(ENSG00000142794~., data=transformed_gene_dropped_betas)
coefficients <- rbind(model$coefficients,new_model_betas$coefficients,cooks_distance_model$coefficients, leverage_
model$coefficients)
row.names(coefficients) <- c("Normal Model", "DroppedHighDFBetas", "DroppedHighCooks", "DroppedHighLeverage")
```

Sensitiy Analysis Continued

```
# We make a summary table that shows what happens when we drop high DFBetas, highCooks, and High Leverage Points f
rom the model.
model_data <- c()
add_to_coefficients_data <- function(model, model_data){
  current_model <- model
  RSS_selected <- c(crossprod(current_model$residuals))
  MSE <- RSS_selected / length(current_model$residuals)
  current_model_stats <- cbind(sqrt(MSE), BIC(current_model), summary(current_model)$adj.r.squared, DAAG::press(cu
rrent_model), shapiro.test(model$residuals)$p.value, bptest(model)$p.value)
  colnames(current_model_stats) <- c("RMSE", "BIC", "ADJ-R-Squared", "PRESS", "Shapiro_Wilks_Test_P_value", "Breu
sch-Pagan_Test_P_value")
  model_data <- rbind(model_data, current_model_stats)
  shapiro.test(model$residuals)
  return(model_data)
}
model_data <- add_to_coefficients_data(model, model_data)
model_data <- add_to_coefficients_data(cooks_distance_model, model_data)
model_data <- add_to_coefficients_data(new_model_betas, model_data)
model_data <- add_to_coefficients_data(leverage_model, model_data)

# Here is the summary table
summary_of_excluding_points <- cbind(coefficients, model_data)
knitr::kable(summary_of_excluding_points)
```

	(Intercept)	rs12734589	rs1976403	rs4654753	rs6694671	rs113548640	rs10916989	RMSE	BIC	ADJ-R-Squared	PRESS	Shapiro_Wilks_Te
Normal Model	0.5242534	0.0270721	- 0.0198311	0.0339801	0.0357185	-0.0362962	-0.0243674	0.0495121	- 1022.8074	0.5673627	0.8632144	
DroppedHighDFBetas	0.5242154	0.0268792	- 0.0159879	0.0372453	0.0372679	-0.0397645	-0.0286545	0.0441744	- 1038.9073	0.6071241	0.6451452	
DroppedHighCooks	0.5288562	0.0291684	- 0.0170019	0.0298446	0.0345575	-0.0387295	-0.0245814	0.0434263	-947.5295	0.6313659	0.5618019	
DroppedHighLeverage	0.5282415	0.0369003	- 0.0150414	0.0312738	0.0309058	-0.0392810	-0.0287681	0.0488392	-981.2104	0.5852552	0.7975638	

```
# We only would consider dropping HighDF Betas since dropping the other elements hurts BIC and might
# even cause the non constant error vairance assumption not to hold. We choose not to drop any points.
#The coefficients are not changed greatly enough to warrant us to exclude so many
# points that contain valuable variation from the overall population.

# Dropping the High DFBetas points might be something we would consider if
# we had more data to represent the true diversity of the population but dropping
# these points might exclude points that include important variation.
```

Training the Final Model and Diagnostics of Final Model

```r
# We compare the validation model and original model and get MSPR and compare it to MSE.

predictions <- predict(model, transformed_gene_test)
validation_model_one <- lm(formula(model), data=transformed_gene_test)

# Comparison of Model Stats
model_stats <- cbind(sigma(model)^2, summary(model)$adj.r.squared, DAAG::press(model))
validation_model_stats <- cbind(sigma(validation_model_one)^2, summary(validation_model_one)$adj.r.squared, DAAG::
press(validation_model_one))
comparison_of_stats <- rbind(model_stats, validation_model_stats)
colnames(comparison_of_stats) <- c("MSE", "Adj-R-Squared","PRESS")
rownames(comparison_of_stats) <- c("Original Model", "Validation Model")
comparison_of_stats
```

```
##                          MSE Adj-R-Squared     PRESS
## Original Model   0.002503449     0.5673627 0.8632144
## Validation Model 0.002299010     0.5480023 0.1912980
```

```r
knitr::kable(comparison_of_stats)
```

	MSE	Adj-R-Squared	PRESS
Original Model	0.0025034	0.5673627	0.8632144
Validation Model	0.0022990	0.5480023	0.1912980

```r
# MSE vs MSPR
mspr_vs_mse <- cbind(sigma(model)^2, mean((transformed_gene_test[["ENSG00000142794"]]-predictions)^2))
colnames(mspr_vs_mse) <- c("MSE", "MSPR")
knitr::kable(mspr_vs_mse)
```

MSE	MSPR
0.0025034	0.0021703

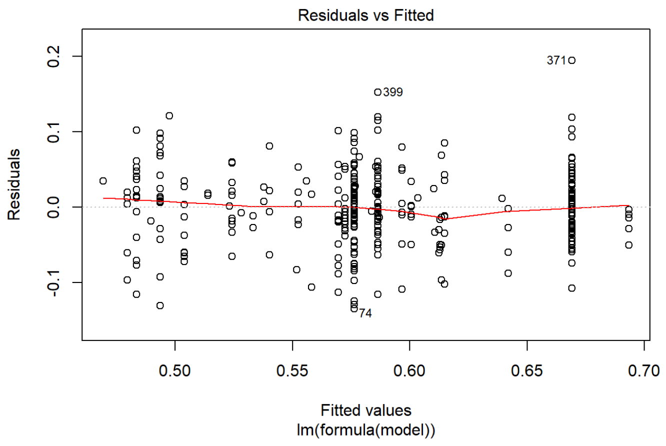
```r
# Fitting the Final Multiple Linear Regression Model :)
final_gene_data <- read_in_pruned_datasets_for_gene_0.8("ENSG00000142794", "D:\\Project\\GitStash\\Applied_regress
ion_project\\")
transformed_final_gene_data <- final_gene_data
transformed_final_gene_data[["ENSG00000142794"]] <- 1/(sqrt(transformed_final_gene_data[["ENSG00000142794"]]+2))
final_model <- lm(formula(model), data=transformed_gene_data)
summary(final_model)
```

```
##
## Call:
## lm(formula = formula(model), data = transformed_gene_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.134598 -0.026933  0.000069  0.028130  0.194233
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.524253   0.010447  50.183  < 2e-16 ***
## rs12734589   0.027072   0.006622   4.088 5.46e-05 ***
## rs1976403   -0.019831   0.006733  -2.946 0.003453 **
## rs4654753    0.033980   0.008821   3.852 0.000141 ***
## rs6694671    0.035719   0.007743   4.613 5.68e-06 ***
## rs113548640 -0.036296   0.011122  -3.263 0.001216 **
## rs10916989  -0.024367   0.007821  -3.116 0.001995 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05003 on 330 degrees of freedom
## Multiple R-squared:  0.5751, Adjusted R-squared:  0.5674
## F-statistic: 74.44 on 6 and 330 DF,  p-value: < 2.2e-16
```

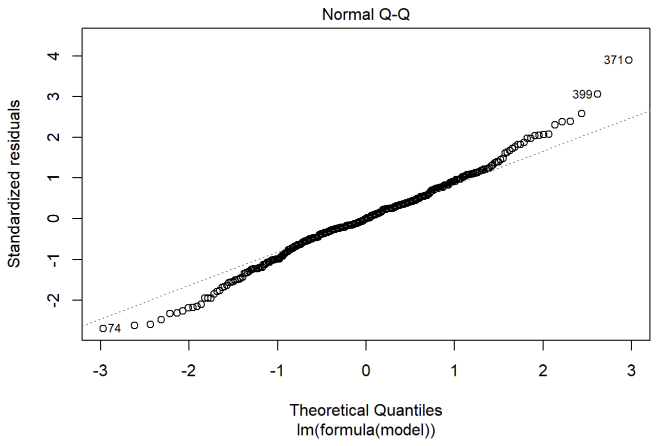
```r
knitr::kable(anova(final_model))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
rs12734589	1	0.8610066	0.8610066	343.928080	0.0000000
rs1976403	1	0.1251881	0.1251881	50.006228	0.0000000
rs4654753	1	0.0464546	0.0464546	18.556256	0.0000218
rs6694671	1	0.0403177	0.0403177	16.104843	0.0000742
rs113548640	1	0.0208500	0.0208500	8.328506	0.0041598
rs10916989	1	0.0243041	0.0243041	9.708235	0.0019954
Residuals	330	0.8261383	0.0025034	NA	NA

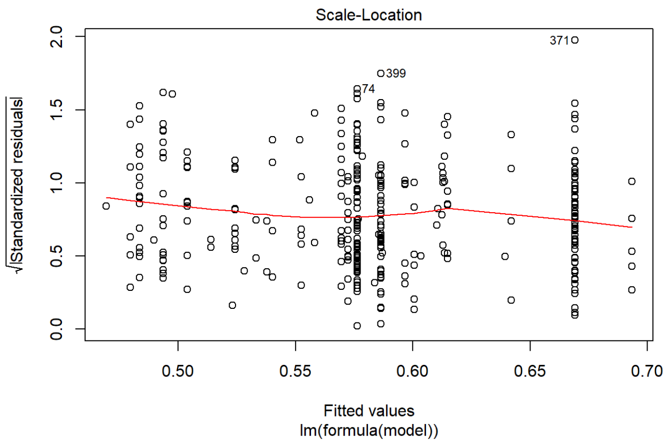
```r
plot(final_model, which=1)
```



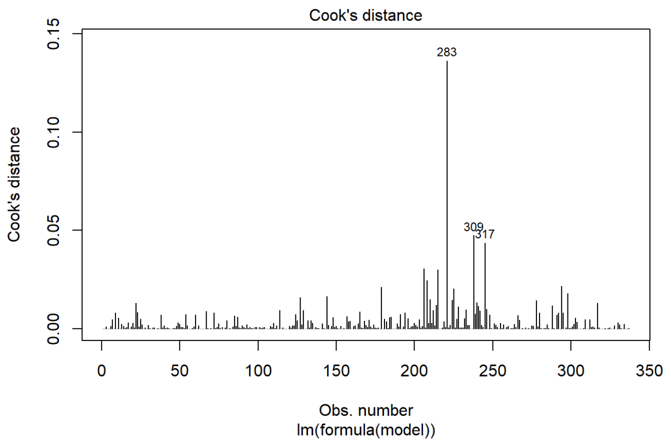
```
plot(final_model, which=2)
```



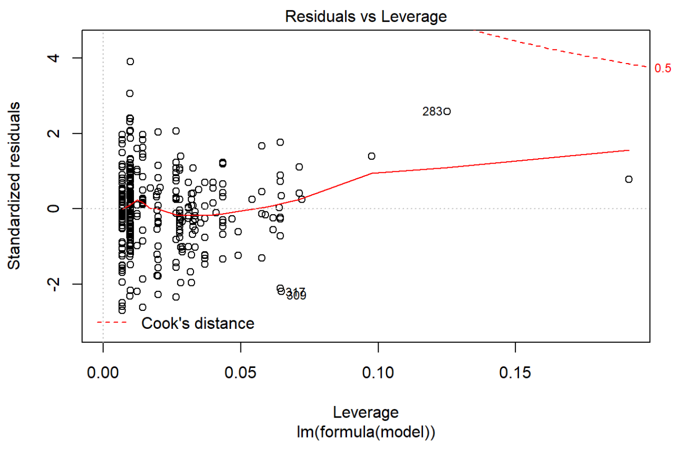
```
plot(final_model, which=3)
```



```
plot(final_model, which=4)
```



```r
plot(final_model, which=5)
```



```r
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.98986, p-value = 0.01963
```

```r
bptest(final_model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  final_model
## BP = 2.7847, df = 6, p-value = 0.8353
```

```r
# We note the same diagnostic findings as we did from the selected model. There is a slight issue with normality a
nd some outliers.
```

Outlier Analysis in Final Model

```r
n <- nrow(transformed_final_gene_data)
p <- length(final_model$coefficients)
model <- final_model
# We do outlier analysis here

cd <- cooks.distance(model)
transformed_gene_dropped <- model$model[-c(which(cd > 4 / (n-p))),]
cooks_distance_model <- lm(ENSG00000142794~., data=transformed_gene_dropped)



leverage_threshold <- (3*(p))/n
leverage_values <- hatvalues(model)
transformed_gene_dropped_leverage <- model$model[-c(which(leverage_values > leverage_threshold)),]
leverage_model <- lm(ENSG00000142794~., data=transformed_gene_dropped_leverage)

betas <- as.data.frame(dfbetas(model))

# We print the largest DFBeta for each parameter and the index of that DFBeta.
to_remove <- c()
for(column in betas){
  outliers <- which(abs(column) > 2/sqrt(n))
  print(max(abs(column)))
  print(which(abs(column) == max(abs(column))))
  to_remove <- union(to_remove, outliers)
}
```

```
## [1] 0.4216956
## [1] 238
## [1] 0.5156542
## [1] 245
## [1] 0.8014485
## [1] 221
## [1] 0.3773799
## [1] 221
## [1] 0.7861822
## [1] 221
## [1] 0.8713133
## [1] 221
## [1] 0.3917799
## [1] 238
```

```
#238, 221, and 245 are possible massive outliers
transformed_gene_dropped_betas <- model$model[-c(to_remove),]
new_model_betas <- lm(ENSG00000142794~., data=transformed_gene_dropped_betas)
coefficients <- rbind(model$coefficients,new_model_betas$coefficients, cooks_distance_model$coefficients, leverage
_model$coefficients)
row.names(coefficients) <- c("Normal Model", "DroppedHighDFBetas", "DroppedHighCooks", "DroppedHighLeverage")
```

Outlier Analysis Continued of Final Model

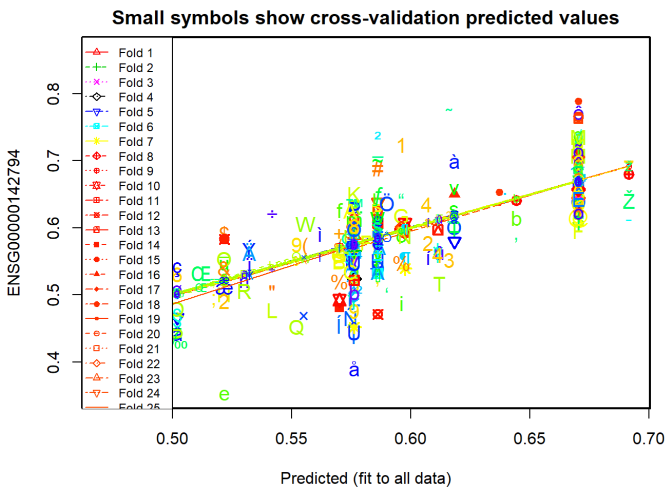
```
# We make a summary table that shows what happens when we drop high DFBetas, highCooks, and High Leverage Points f
rom the model.
model_data <- c()
add_to_coefficients_data <- function(model, model_data){
  current_model <- model
  RSS_selected <- c(crossprod(current_model$residuals))
  MSE <- RSS_selected / length(current_model$residuals)
  current_model_stats <- cbind(sqrt(MSE), BIC(current_model), summary(current_model)$adj.r.squared, DAAG::press(cu
rrent_model), shapiro.test(model$residuals)$p.value, bptest(model)$p.value)
  colnames(current_model_stats) <- c("RMSE", "BIC", "ADJ-R-Squared", "PRESS", "Shapiro_Wilks_Test_P_value", "Breu
sch-Pagan_Test_P_value")
  model_data <- rbind(model_data, current_model_stats)
  shapiro.test(model$residuals)
  return(model_data)
}
model_data <- add_to_coefficients_data(model, model_data)
model_data <- add_to_coefficients_data(cooks_distance_model, model_data)
model_data <- add_to_coefficients_data(new_model_betas, model_data)
model_data <- add_to_coefficients_data(leverage_model, model_data)

# Here is the summary table
summary_of_excluding_points <- cbind(coefficients, model_data)
knitr::kable(summary_of_excluding_points)
```

	(Intercept)	rs12734589	rs1976403	rs4654753	rs6694671	rs113548640	rs10916989	RMSE	BIC	ADJ-R-Squared	PRESS	Shapiro_Wilks_Te
Normal Model	0.5242534	0.0270721	-0.0198311	0.0339801	0.0357185	-0.0362962	-0.0243674	0.0495121	-1022.8074	0.5673627	0.8632144	
DroppedHighDFBetas	0.5244619	0.0294151	-0.0164700	0.0376958	0.0333010	-0.0355336	-0.0282190	0.0434386	-1029.1651	0.6155703	0.6111564	
DroppedHighCooks	0.5242282	0.0298767	-0.0116887	0.0281422	0.0377016	-0.0447463	-0.0243940	0.0425174	-921.8093	0.6437677	0.5179535	
DroppedHighLeverage	0.5282157	0.0376963	-0.0170775	0.0327219	0.0280078	-0.0343316	-0.0281172	0.0490726	-952.8234	0.5846780	0.7864094	

```
#We note that dropping the high DFBetas again results in a satisfied normality assumption but we choose to retain
these points.
```

Leave One Out Cross Validation MSE of Final Model

```
p <- length(model$coefficients)
n <- nrow(model$model)
cv.lm(model, data=transformed_final_gene_data, m=n, printit = FALSE)
```



```
leave_one_out_mse <- press(model)/(n-p)
# Leave one out cross-validated MSE is 0.00262. This is excellent
```