ANN_Covid_College_pred

Christopher Vo

3/26/2021

Artificial Neural Networks

```
# Importing the dataset
#dataset = read.csv('Updated_colleges - Sheet1.csv')
dataset = read.csv('/Users/chrisvo/Desktop/Colleges_covid_cases\ -\ Colleges.csv', header=TRUE, sep = "

# including all the necessary independent variables and the dependent variable
dataset = dataset[2:9]

# Encoding the categorical data as factors
dataset$Red.Blue.State = as.numeric(factor(dataset$Red.Blue.State,
                                           levels = c('Red', 'Blue'),
                                           labels = c(1, 2)))

dataset$Type.of.University = as.numeric(factor(dataset$Type.of.University,
                                               levels = c('private', 'public'),
                                               labels = c(1, 2)))

dataset$Area = as.numeric(factor(dataset$Area,
                                 levels = c('urban', 'city', 'suburban', 'rural'),
                                 labels = c(1, 2, 3, 4)))

dataset$US.Region = as.numeric(factor(dataset$US.Region,
                                      levels = c('southwest', 'southeast', 'midwest', 'northeast', 'pacific'
                                      labels = c(1, 2, 3, 4, 5, 6)))
```

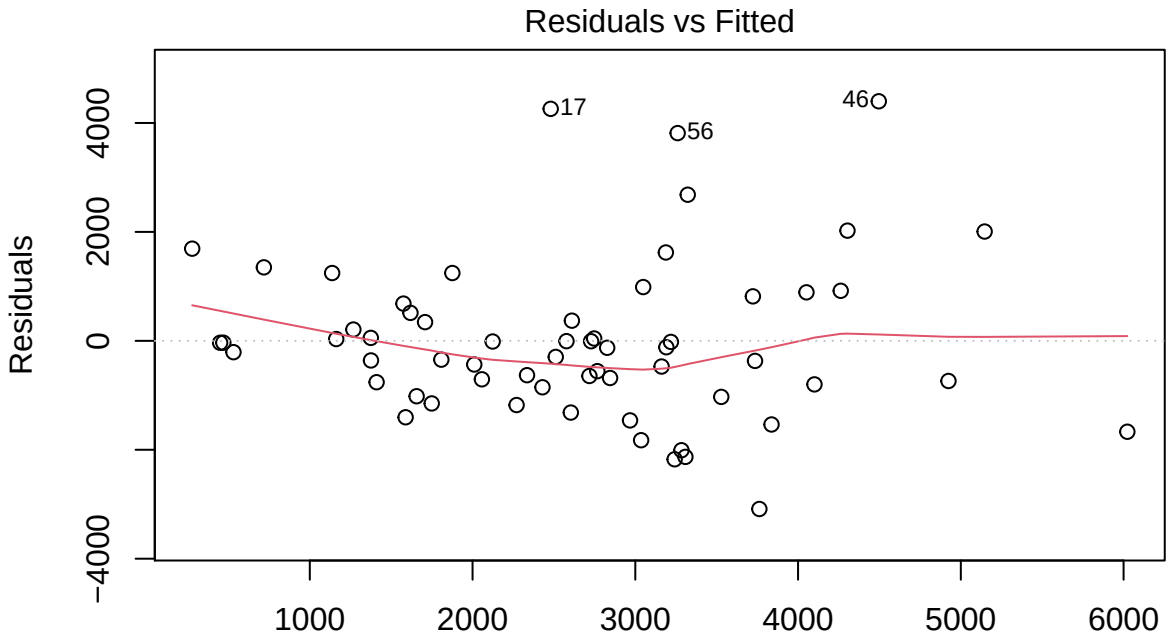
```
# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Cases, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

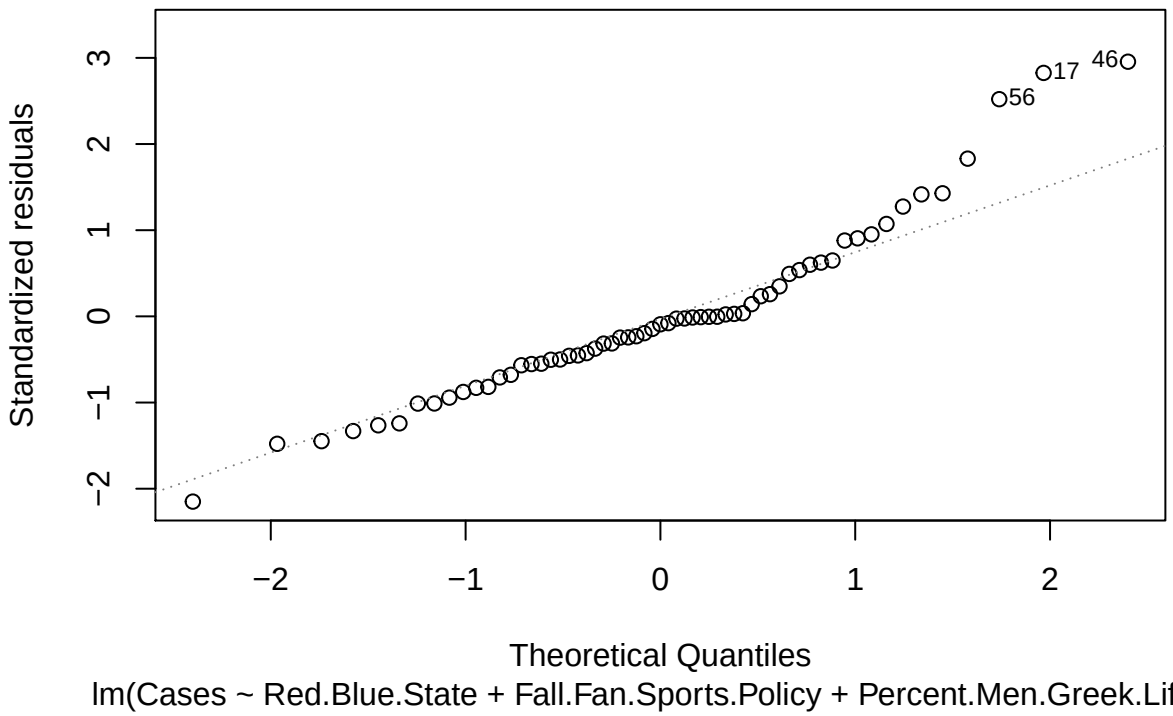
Multiple Linear Regression

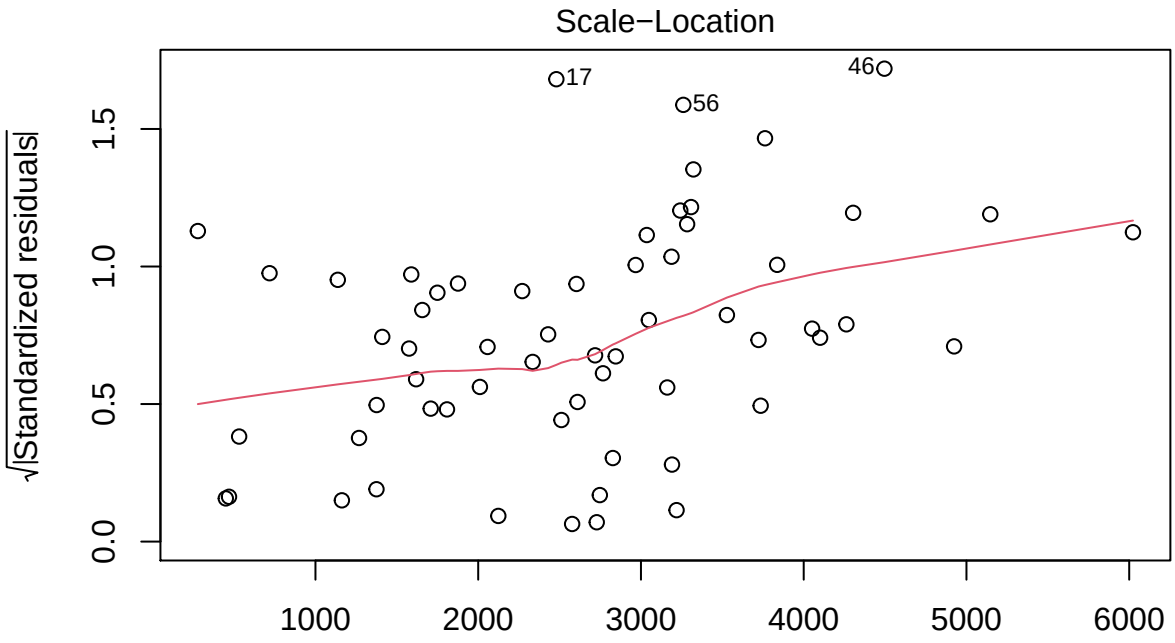
```
full_model <- lm(Cases ~ Red.Blue.State + Fall.Fan.Sports.Policy + Percent.Men.Greek.Life + Undergrad.E

summary(full_model)
```

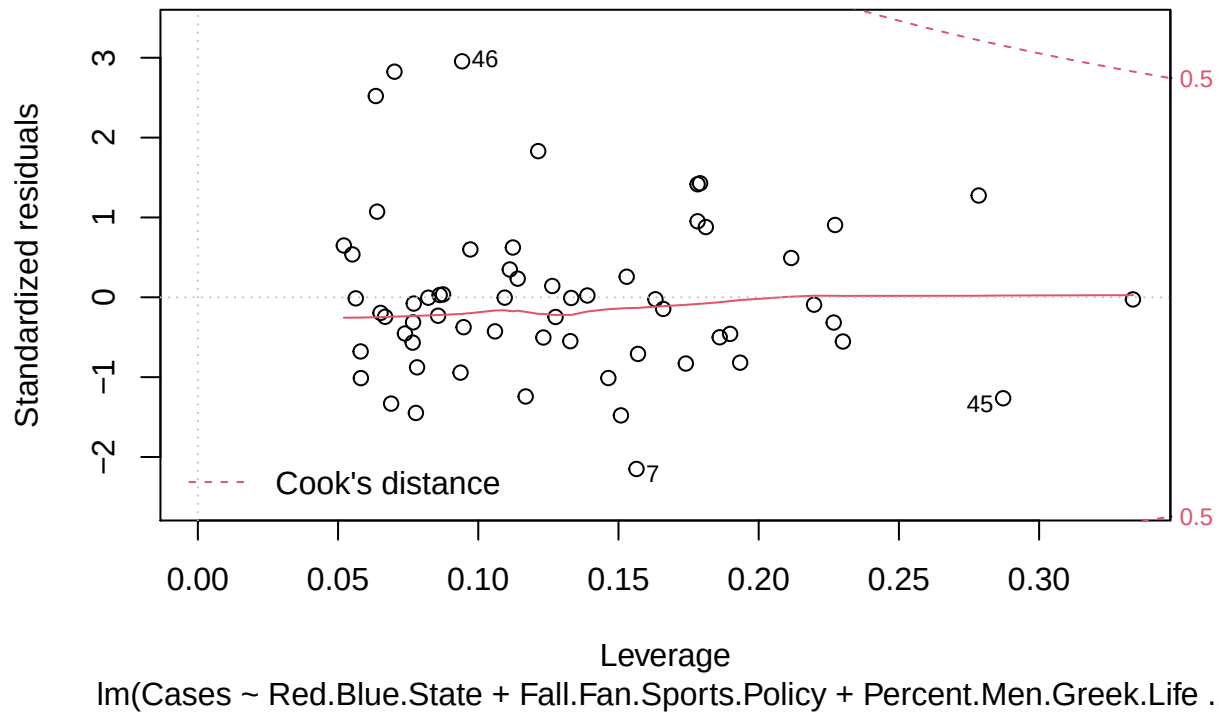
```
##
## Call:
## lm(formula = Cases ~ Red.Blue.State + Fall.Fan.Sports.Policy +
##     Percent.Men.Greek.Life + Undergrad.Enrollment + Type.of.University +
##     Area + US.Region, data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3087.2  -799.7  -127.4   683.6  4398.1
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1360.2989  1708.3478  -0.796   0.4294
## Red.Blue.State          -561.9576   506.1668  -1.110   0.2719
## Fall.Fan.Sports.Policy    62.8678   509.4920   0.123   0.9023
## Percent.Men.Greek.Life  6929.8450  3411.1204   2.032   0.0472 *
## Undergrad.Enrollment       0.1280     0.0284   4.506 3.68e-05 ***
## Type.of.University       205.3924   772.8109   0.266   0.7914
## Area                     108.5559   272.1858   0.399   0.6916
## US.Region                -45.0828   168.3558  -0.268   0.7899
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1564 on 53 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.4055, Adjusted R-squared:  0.327
## F-statistic: 5.164 on 7 and 53 DF,  p-value: 0.0001581
```

```r
plot.new()

plot(full_model)
```

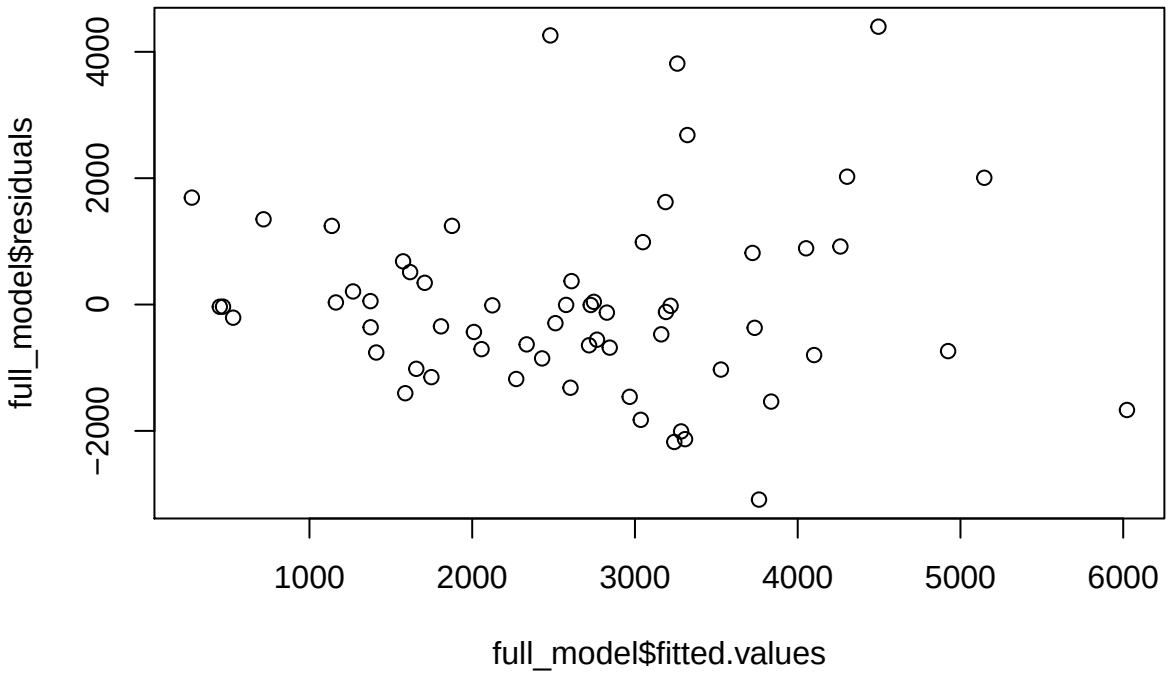

Residuals vs Fitted
Residuals
Fitted values
lm(Cases ~ Red.Blue.State + Fall.Fan.Sports.Policy + Percent.Men.Greek.Life ...
17
56
46
4000
2000
0
−4000
1000
2000
3000
4000
5000
6000


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Cases ~ Red.Blue.State + Fall.Fan.Sports.Policy + Percent.Men.Greek.Life ...
17
56
46
3
2
1
0
−1
−2
−2
−1
0
1
2

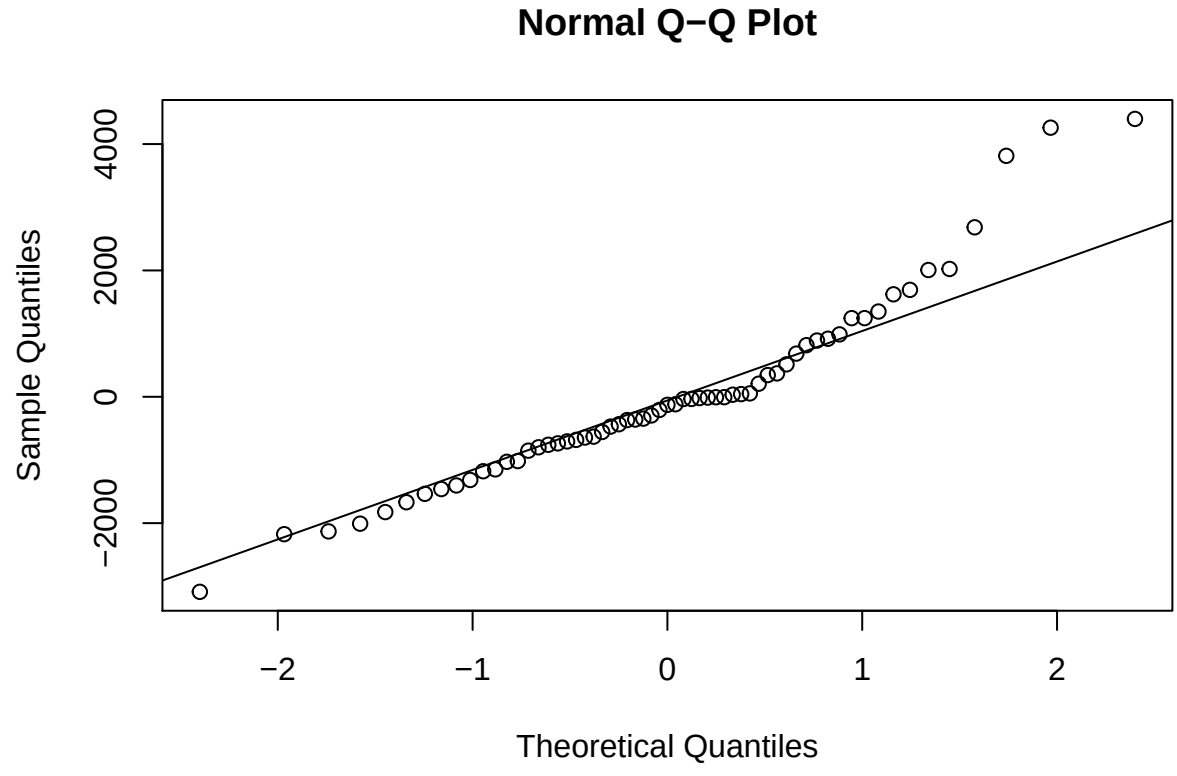





```
# LINE Assumptions
# residuals vs. fits plot
plot(x = full_model$fitted.values, y = full_model$residuals)
```




```
# QQ Plot
qqnorm(full_model$residuals)
qqline(full_model$residuals)
```




```
# Shapiro-Wilks Test
shapiro.test(full_model$residuals) # normality condition is not met
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  full_model$residuals
## W = 0.93471, p-value = 0.002877
```

```
library(car)
```

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

```
# VIF of all predictors relatively low
# No dependencies detected
vif(full_model)
```

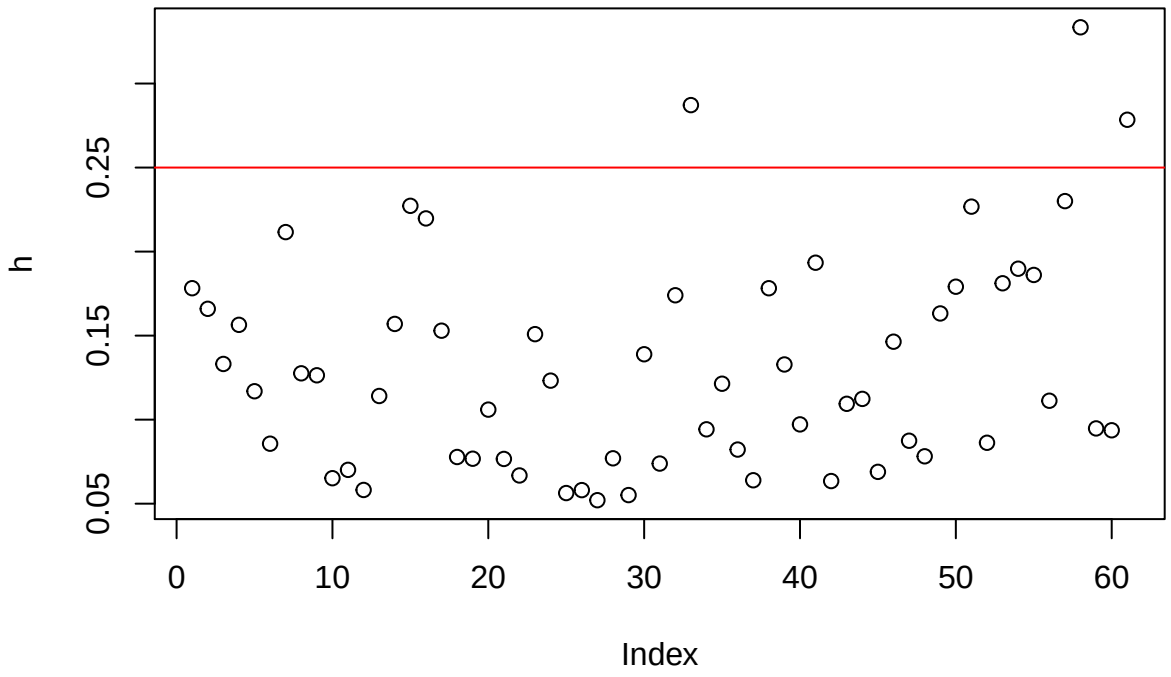
```
##         Red.Blue.State Fall.Fan.Sports.Policy Percent.Men.Greek.Life
##               1.501566               1.618839               2.159123
##   Undergrad.Enrollment     Type.of.University                   Area
##               2.213200               2.202698               1.331409
##              US.Region
##               1.579059
```

```
# checking for outliers in model
# find studentized residuals
# residual error estimate
e_hat = full_model$residuals

# sigma2 estimate
sigma2_hat = summary(full_model)$sigma^2

# construct the Hat matrix using design matrix ---> leverage
# design matrix
# the first column is 1 - estimation of intercept
X = model.matrix(full_model)

# hat matrix
H = X %*% solve(t(X) %*% X) %*% t(X)

# diagonal element
# get diagonal elements of hat matrix
h = diag(H)
h
```

```
##          2          3          6          7          9         10         12
## 0.17820607 0.16594805 0.13313450 0.15641624 0.11690177 0.08567914 0.21160427
##         13         14         15         17         18         19         22
## 0.12756476 0.12638376 0.06515261 0.07011819 0.05814720 0.11406461 0.15695342
##         23         25         26         27         28         29         30
## 0.22723373 0.21974889 0.15292971 0.07776433 0.07674807 0.10595986 0.07661292
##         33         35         36         37         38         39         40
## 0.06678023 0.15087583 0.12321146 0.05631397 0.05804721 0.05206185 0.07698815
##         41         42         43         44         45         46         47
## 0.05510795 0.13891153 0.07385522 0.17400965 0.28716994 0.09423206 0.12136844
##         48         49         51         52         54         55         56
## 0.08218697 0.06392150 0.17820172 0.13282575 0.09717874 0.19338389 0.06345597
##         57         58         61         62         63         64         66
## 0.10942930 0.11234154 0.06891804 0.14638600 0.08743839 0.07817424 0.16318560
##         69         70         71         73         74         75         76
## 0.17909620 0.22675069 0.08623095 0.18114802 0.18981221 0.18609601 0.11124441
```

```
##         77         78         79         80         81
## 0.23003337 0.33346684 0.09476800 0.09367024 0.27844979
```

```
# Large leverage points
p = length(coefficients(full_model))
n = 64

# Check for large leverage points
thresh1 = 2*p / n
thresh2 = 3*p / n

# scatterplot
# plot the leverage
plot(h)

# plot the first threshold
abline(h = thresh1, col = 'red')

# plot the second threshold
abline(h = thresh2, col = 'blue') # none above here
```




```
# use which() function to check
# these are not necessarily outliers

which(h >= thresh1)
```

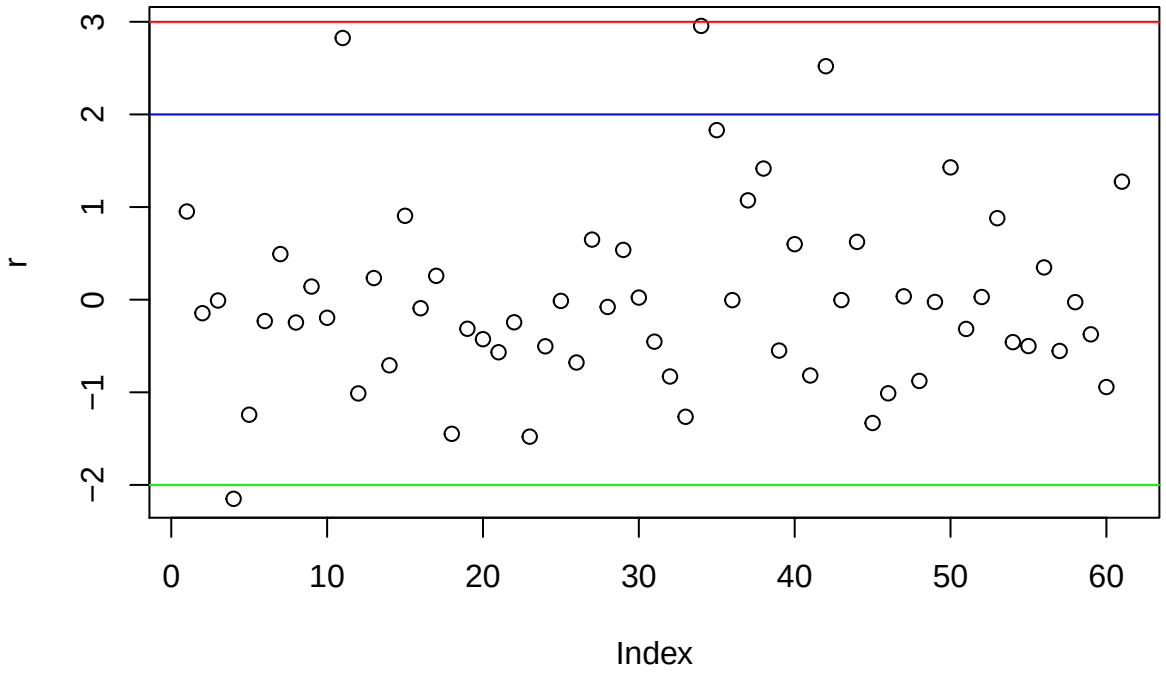
```
## 45 78 81
## 33 58 61
```

```
which(h >= thresh2)
```

```
## named integer(0)
```

```
# Calculate studentized residuals
r = e_hat / sqrt(sigma2_hat * (1-h) )
```

```r
# scatter plot involving the studentized deleted residuals.
plot(r)

# create cuttoff points of 2 and 3
abline(a = 2, b = 0, col = 'blue')
abline(a = 3, b = 0, col = 'red')
abline(a = -2, b = 0, col = 'green')
```




```r
# which studentized residual is greater than 2 and 3
which(r >= 2)
```

```
## 17 46 56
## 11 34 42
```

```r
which(r >= 3)
```

```
## named integer(0)
```

```r
which( r <= -2)
```

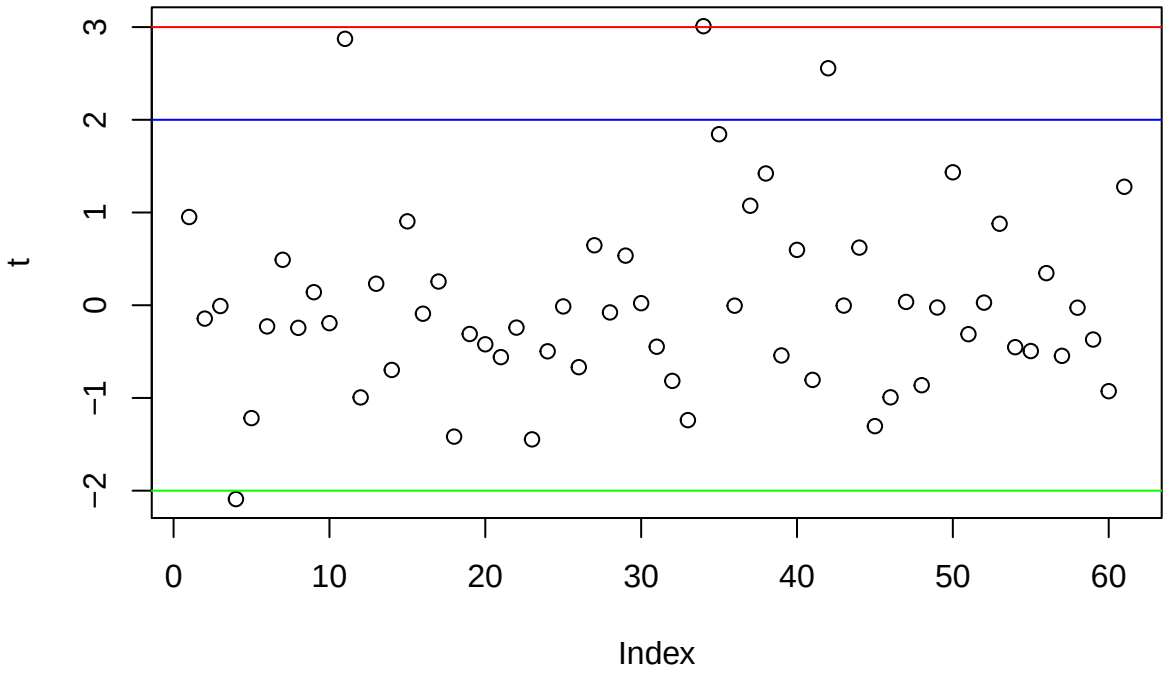
```
## 7
## 4
```

```r
which( r <= -3)
```

```
## named integer(0)
```

```r
# Studentized deleted residuals
# r is from the calculation of the studentized residuals
t = r * ((n-p-1) / (n-p-r)) ^(1/2)
plot(t)
abline( h = 2, col = 'blue')
abline( h = 3, col = 'red')

abline( h = -2, col = 'green')
```

```r
abline( h = -3, col = 'orange')
```




```r
# common cut-off points |t_i| >= 2 or 3
which(t >= 2)
```

```
## 17 46 56
## 11 34 42
```

```r
which(t >= 3)
```

```
## 46
## 34
```

```r
which(t <= -2)
```

```
## 7
## 4
```

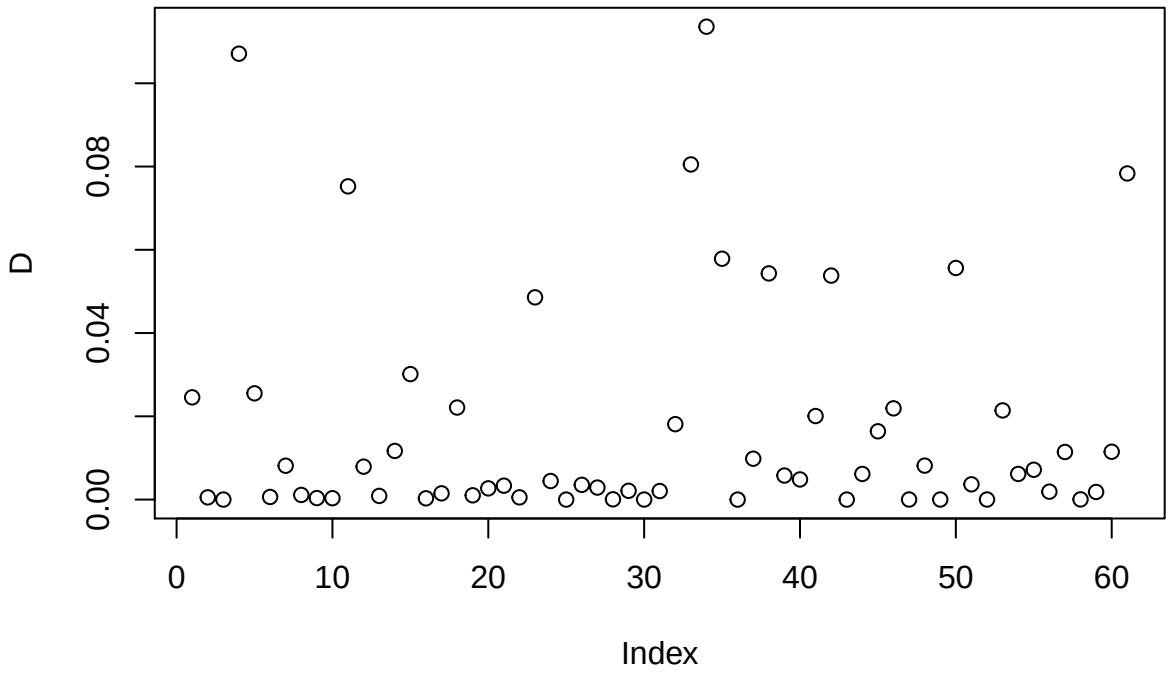
```r
which(t <= -3)
```

```
## named integer(0)
```

```r
order(abs(t))
```

```
##  [1] 43 36  3 25 30 49 58 52 47 28 16  9  2 10  6 13 22  8 17 19 51 56 59 20 31
## [26] 54  7 55 24 29 39 57 21 40 44 27 26 14 41 32 48 53 15 60  1 46 12 37  5 33
## [51] 61 45 18 38 50 23 35  4 42 11 34
```

```r
# Cook's Distance
# r is studentized residual
# cook's distance for each point
D = (1 / p) * r^2 * (h / (1-h) )
plot(D)
```




```
which(D >= 1)
```

```
## named integer(0)
```

```
which(D >= 0.5)
```

```
## named integer(0)
```

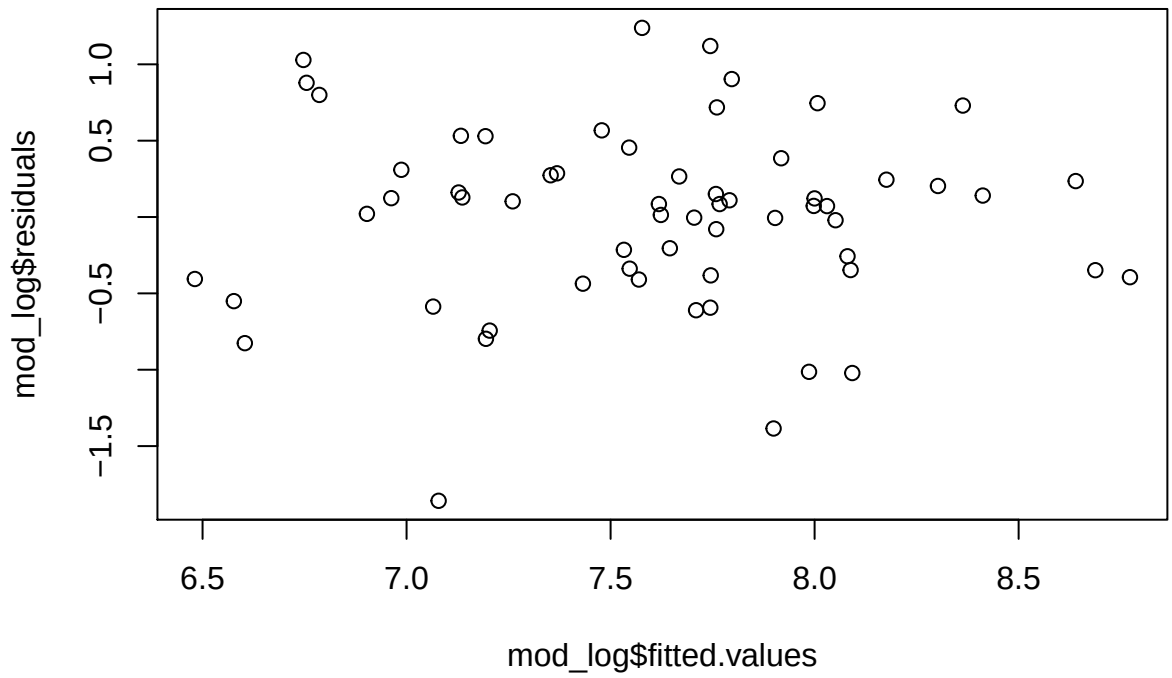
Let's do a log transformation on the model

```
mod_log <- lm(log(Cases) ~ Red.Blue.State + Fall.Fan.Sports.Policy + Percent.Men.Greek.Life + Undergrad

summary(mod_log)
```

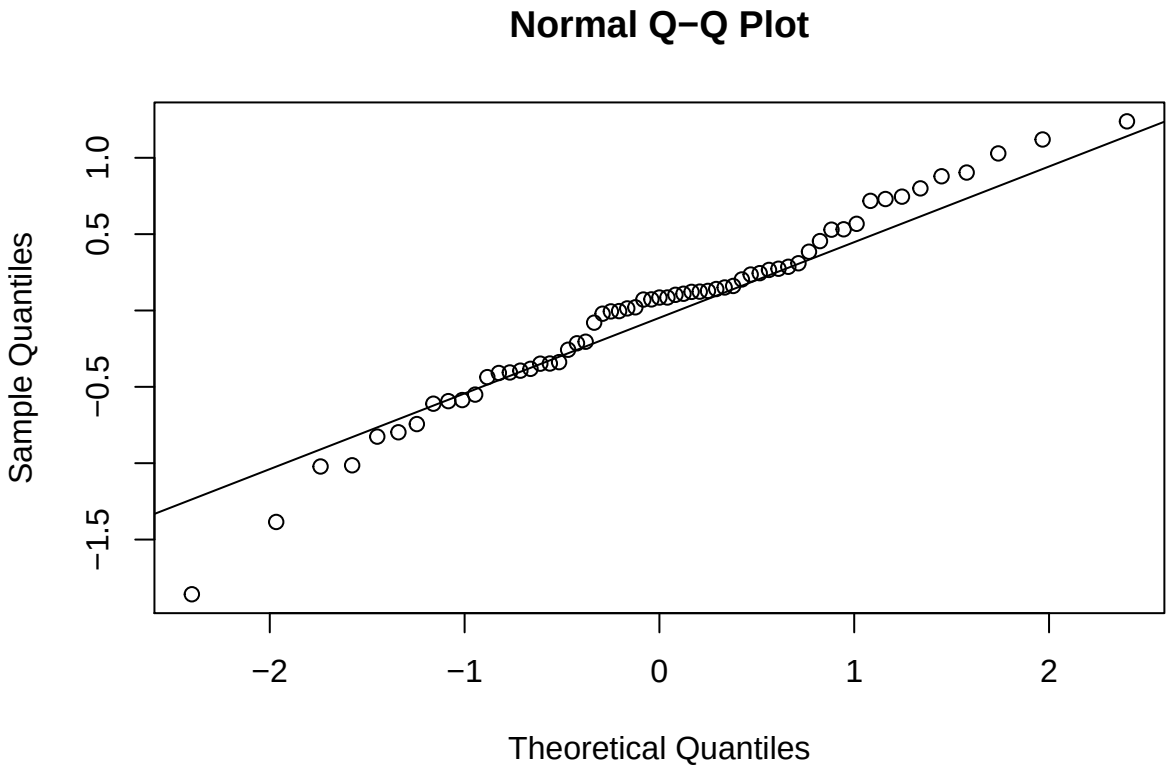
```
##
## Call:
## lm(formula = log(Cases) ~ Red.Blue.State + Fall.Fan.Sports.Policy +
##     Percent.Men.Greek.Life + Undergrad.Enrollment + Type.of.University +
##     Area + US.Region, data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85817 -0.38210  0.08481  0.28618  1.23860
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.455e+00  6.973e-01   7.823 2.15e-10 ***
## Red.Blue.State         -4.116e-01  2.066e-01  -1.992  0.05148 .
## Fall.Fan.Sports.Policy  1.476e-02  2.080e-01   0.071  0.94370
## Percent.Men.Greek.Life  4.114e+00  1.392e+00   2.955  0.00466 **
## Undergrad.Enrollment    4.897e-05  1.159e-05   4.224 9.47e-05 ***
## Type.of.University      3.948e-01  3.154e-01   1.252  0.21618
```

```
## Area                    5.925e-03  1.111e-01   0.053  0.95767
## US.Region               3.228e-02  6.872e-02   0.470  0.64050
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6382 on 53 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.4354, Adjusted R-squared:  0.3608
## F-statistic: 5.838 on 7 and 53 DF,  p-value: 4.742e-05
```

```r
plot.new()
# residuals vs. fits plot
plot(x = mod_log$fitted.values, y = mod_log$residuals)
```




```r
# QQ Plot
qqnorm(mod_log$residuals)
qqline(mod_log$residuals)
```




```
# Shapiro-Wilks Test
shapiro.test(mod_log$residuals) # normality condition is met
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mod_log$residuals
## W = 0.97772, p-value = 0.3292
```

```
# Rechecking collinearity issues
# There are no collinearity issues
# All predictors have VIF's less than 2.5
library(car)
vif(mod_log)
```

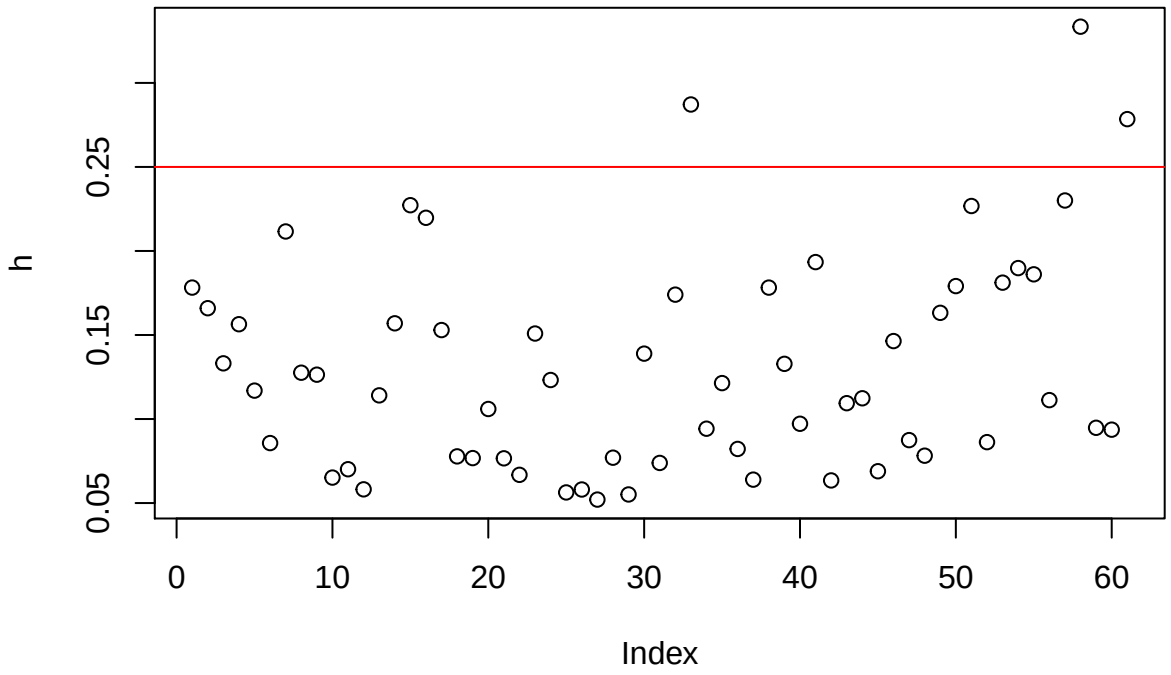
```
##         Red.Blue.State Fall.Fan.Sports.Policy Percent.Men.Greek.Life
##               1.501566               1.618839               2.159123
##   Undergrad.Enrollment     Type.of.University                   Area
##               2.213200               2.202698               1.331409
##              US.Region
##               1.579059
```

```
# checking for outliers in model without population
# find studentized residuals
# residual error estimate
e_hat = mod_log$residuals

# sigma2 estimate
sigma2_hat = summary(mod_log)$sigma^2

# construct the Hat matrix using design matrix ---> leverage
```

```r
# design matrix
# the first column is 1 - estimation of intercept
X = model.matrix(mod_log)

# hat matrix
H = X %*% solve(t(X) %*% X) %*% t(X)

# diagonal element
# get diagonal elements of hat matrix
h = diag(H)
h
```

```
##          2          3          6          7          9         10         12
## 0.17820607 0.16594805 0.13313450 0.15641624 0.11690177 0.08567914 0.21160427
##         13         14         15         17         18         19         22
## 0.12756476 0.12638376 0.06515261 0.07011819 0.05814720 0.11406461 0.15695342
##         23         25         26         27         28         29         30
## 0.22723373 0.21974889 0.15292971 0.07776433 0.07674807 0.10595986 0.07661292
##         33         35         36         37         38         39         40
## 0.06678023 0.15087583 0.12321146 0.05631397 0.05804721 0.05206185 0.07698815
##         41         42         43         44         45         46         47
## 0.05510795 0.13891153 0.07385522 0.17400965 0.28716994 0.09423206 0.12136844
##         48         49         51         52         54         55         56
## 0.08218697 0.06392150 0.17820172 0.13282575 0.09717874 0.19338389 0.06345597
##         57         58         61         62         63         64         66
## 0.10942930 0.11234154 0.06891804 0.14638600 0.08743839 0.07817424 0.16318560
##         69         70         71         73         74         75         76
## 0.17909620 0.22675069 0.08623095 0.18114802 0.18981221 0.18609601 0.11124441
##         77         78         79         80         81
## 0.23003337 0.33346684 0.09476800 0.09367024 0.27844979
```

```r
# Large leverage points
p = length(coefficients(mod_log))
n = 64

# Check for large leverage points
thresh1 = 2*p / n
thresh2 = 3*p / n

# scatterplot
# plot the leverage
plot(h)

# plot the first threshold
abline(h = thresh1, col = 'red')

# plot the second threshold
abline(h = thresh2, col = 'blue') # none above here
```



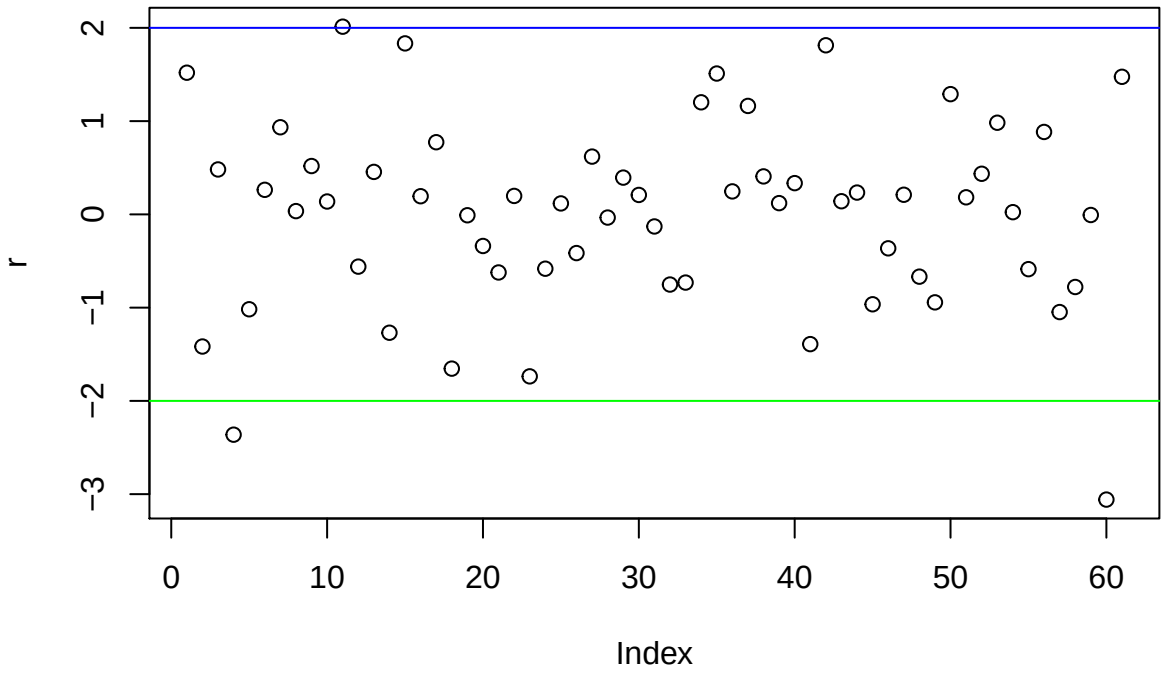

```
# use which() function to check
# these are not necessarily outliers

which(h >= thresh1)
```

```
## 45 78 81
## 33 58 61
```

```
which(h >= thresh2)
```

```
## named integer(0)
```

```
# Calculate studentized residuals
r = e_hat / sqrt(sigma2_hat * (1-h) )

# scatter plot involving the studentized deleted residuals.
plot(r)

# create cuttoff points of 2 and 3
abline(a = 2, b = 0, col = 'blue')
abline(a = 3, b = 0, col = 'red')
abline(a = -2, b = 0, col = 'green')
```




```
# which studentized residual is greater than 2 and 3
which(r >= 2)
```

```
## 17
## 11
```

```
which(r >= 3)
```

```
## named integer(0)
```

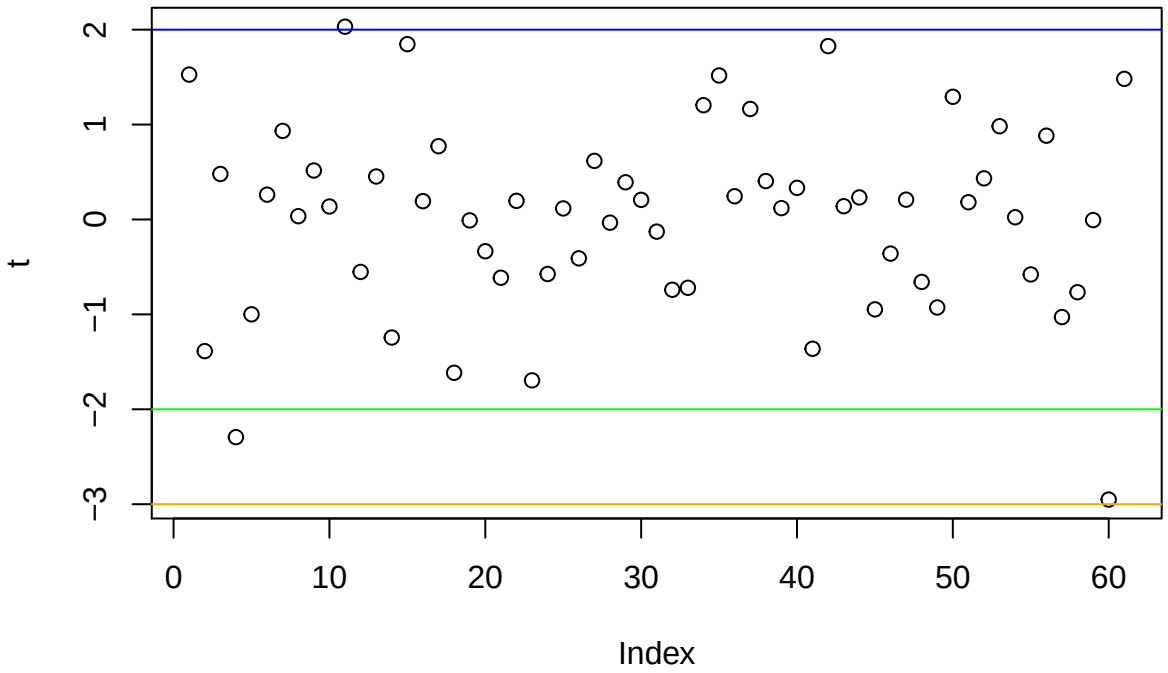
```
which( r <= -2)
```

```
##  7 80
##  4 60
```

```
which( r <= -3)
```

```
## 80
## 60
```

```
# Studentized deleted residuals
# r is from the calculation of the studentized residuals
t = r * ((n-p-1) / (n-p-r)) ^(1/2)
plot(t)
abline( h = 2, col = 'blue')
abline( h = 3, col = 'red')

abline( h = -2, col = 'green')
abline( h = -3, col = 'orange')
```




```
# common cut-off points |t_i| >= 2 or 3
which(t >= 2)
```

```
## 17
## 11
```

```
which(t >= 3)
```

```
## named integer(0)
```

```
which(t <= -2)
```

```
##  7 80
##  4 60
```

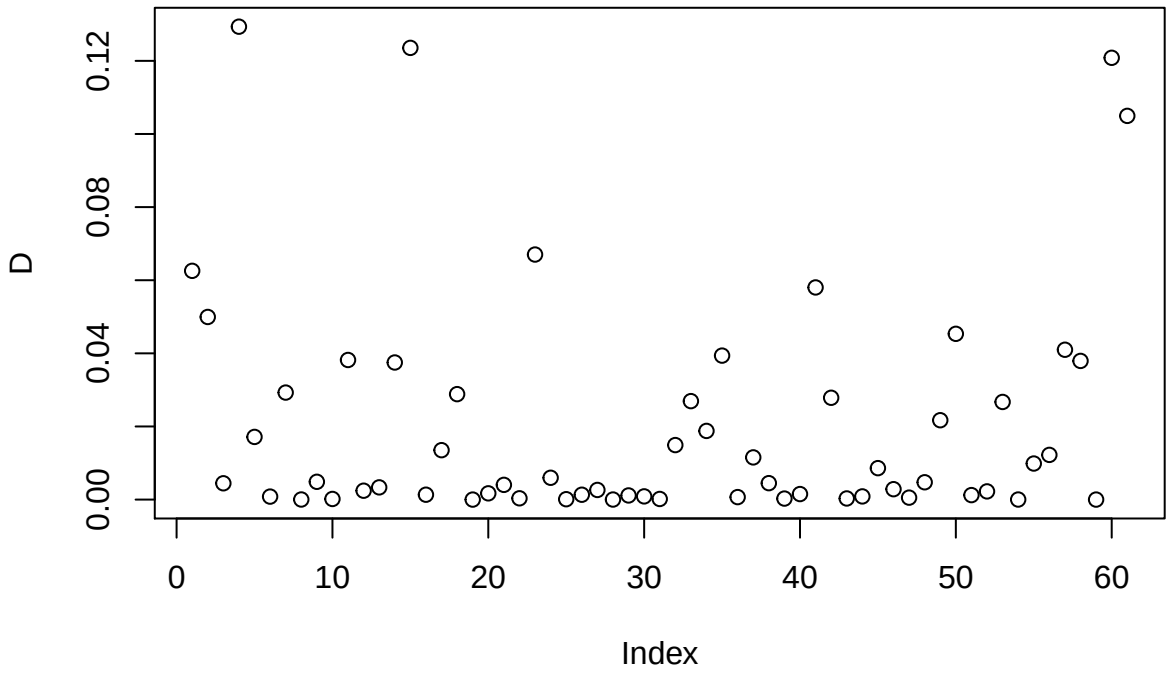
```
which(t <= -3)
```

```
## named integer(0)
```

```
order(abs(t))
```

```
##  [1] 59 19 54 28  8 25 39 31 10 43 51 16 22 30 47 44 36  6 40 20 46 29 38 26 52
## [26] 13  3  9 12 24 55 21 27 48 33 32 58 17 56 49  7 45 53  5 57 37 34 14 50 41
## [51]  2 61 35  1 18 23 42 15 11  4 60
```

```
# Cook's Distance
# r is studentized residual
# cook's distance for each point
D = (1 / p) * r^2 * (h / (1-h) )
plot(D)
```




```
which(D >= 1)
```

```
## named integer(0)
```

```
which(D >= 0.5)
```

```
## named integer(0)
```

```
# There are no clear outliers so we will go with this model and proceed to model selection
```

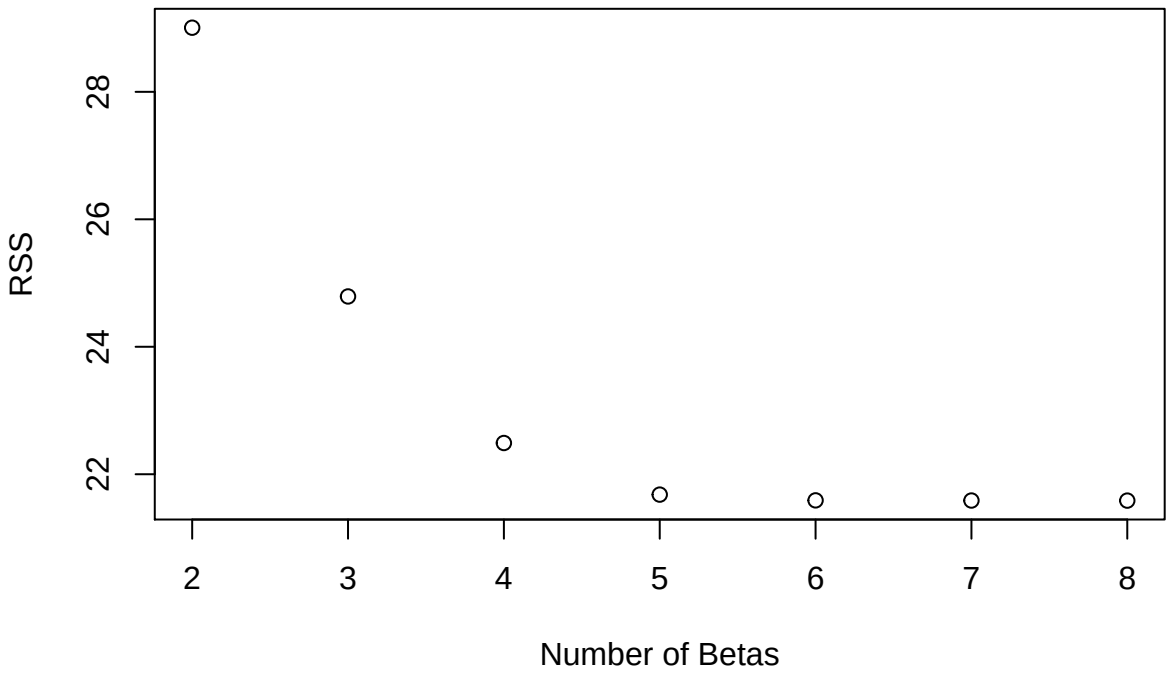
Finding the chosen predictors when n predictors are chosen

```
library(leaps)
# selecting according to RSS
sub_fit <- regsubsets(log(Cases) ~ Red.Blue.State + Fall.Fan.Sports.Policy + Percent.Men.Greek.Life + U

sum_fit <- summary(sub_fit)
sum_fit$which
```

```
##   (Intercept) Red.Blue.State Fall.Fan.Sports.Policy Percent.Men.Greek.Life
## 1        TRUE          FALSE                  FALSE                  FALSE
## 2        TRUE          FALSE                  FALSE                   TRUE
## 3        TRUE           TRUE                  FALSE                   TRUE
## 4        TRUE           TRUE                  FALSE                   TRUE
## 5        TRUE           TRUE                  FALSE                   TRUE
## 6        TRUE           TRUE                   TRUE                   TRUE
## 7        TRUE           TRUE                   TRUE                   TRUE
##   Undergrad.Enrollment Type.of.University  Area US.Region
## 1                 TRUE              FALSE FALSE     FALSE
## 2                 TRUE              FALSE FALSE     FALSE
## 3                 TRUE              FALSE FALSE     FALSE
## 4                 TRUE               TRUE FALSE     FALSE
```

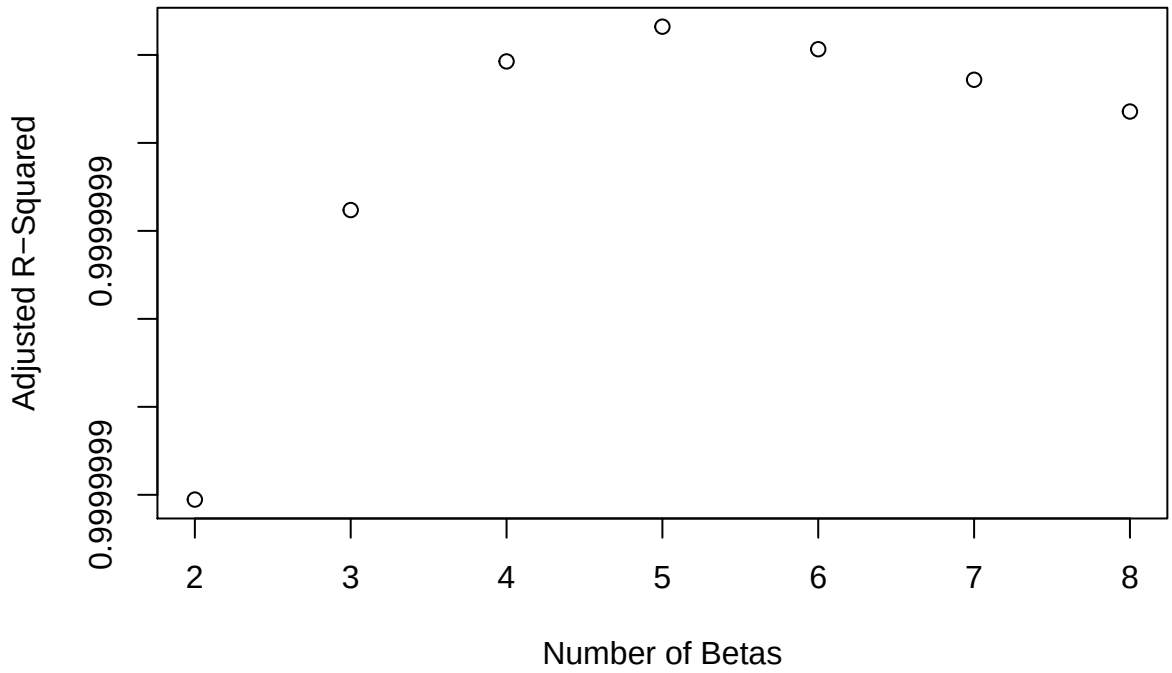
```
## 5                TRUE              TRUE FALSE      TRUE
## 6                TRUE              TRUE FALSE      TRUE
## 7                TRUE              TRUE  TRUE      TRUE
```

```r
# Residual Sum of Squares
RSS = sum_fit$rss
p_all= 2:8
plot(x = p_all, y = RSS,
    xlab="Number of Betas",
    ylab="RSS")
```



Adjusted R-squared

```r
#TSS = sum( (sqrt (df$Weight) - mean(sqrt(df$Weight)) )^2 )
TSS = sum( (training_set$Cases - mean(training_set$Cases) )^2 )
RSS_all = sum_fit$rss
p_full = 8
# vector of all the number of betas
p_all= 2:8
n = 64
R_adj = 1 - (RSS_all/(n-p_all)) / (TSS/(n-1))
plot(x = p_all, y = R_adj,
    xlab="Number of Betas",
    ylab="Adjusted R-Squared")
```



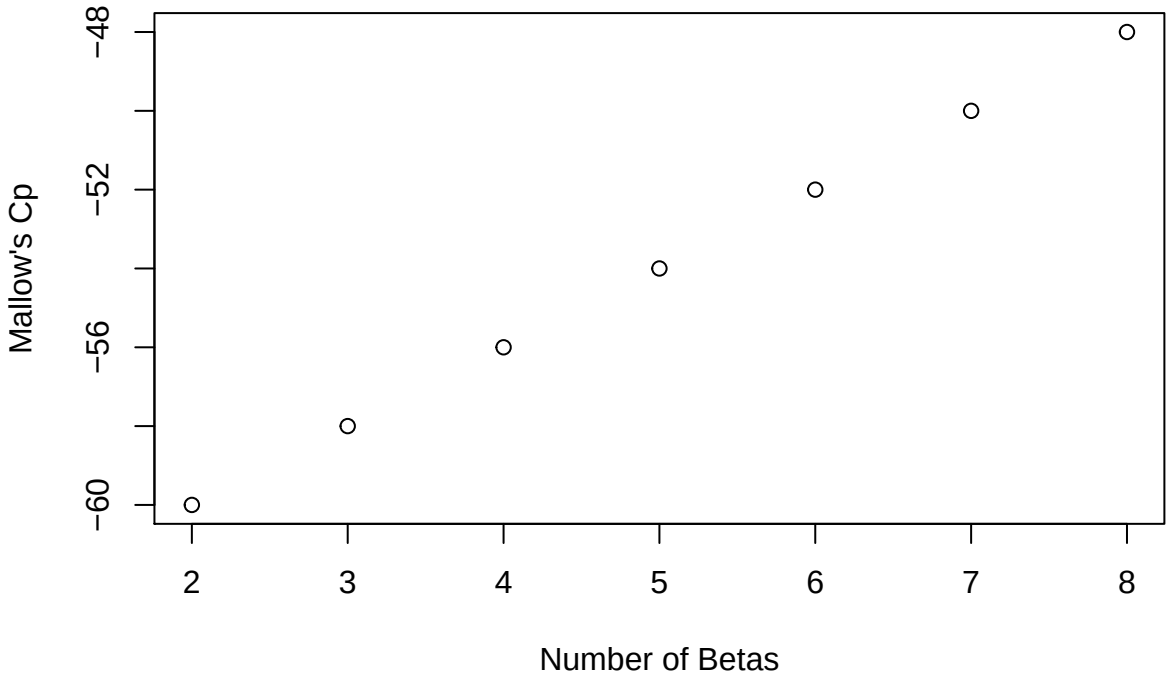

```
# Use function:
sum_fit$adjr2
```

```
## [1] 0.2284492 0.3292389 0.3807814 0.3924178 0.3839475 0.3726086 0.3608053
```

```
# the RSS are not as high as the previous example - about 62%.
# Model selected: 5 betas/4 predictors
```

Mallow's Cp

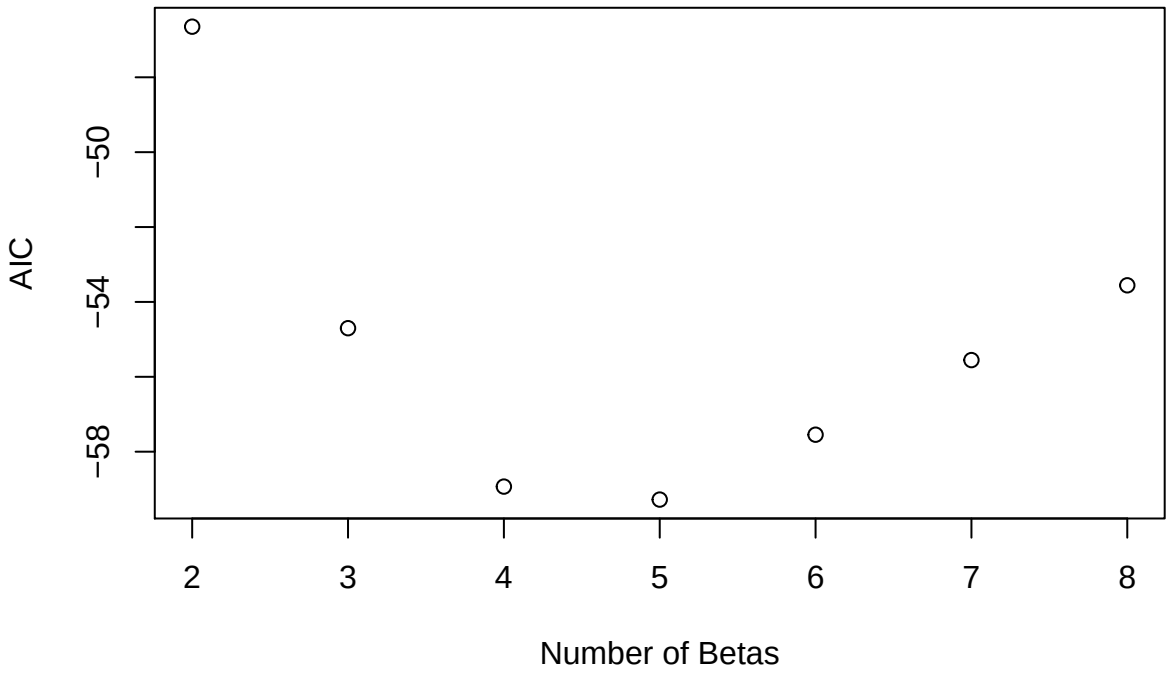
```
sigma2_hat_Full = summary(full_model)$sigma^2
Cp = (RSS_all/sigma2_hat_Full) + 2*p_all - n
# Use function:

plot(x = p_all, y = Cp,
     xlab="Number of Betas",
     ylab="Mallow's Cp")
# intercept 0, slope 1
abline(0,1) # !!!
```




```
# select the smallest cp on the line
# MODEL SELECTED: Will disregard - Need a larger training set for this method to be valid
```

the sub model chosen by AIC

```
# n * log(sigma2_hat) + 2*p
AIC_all <- n * log(RSS_all/n) + 2*p_all
plot(x = p_all, y = AIC_all,
     xlab="Number of Betas",
     ylab="AIC")
```

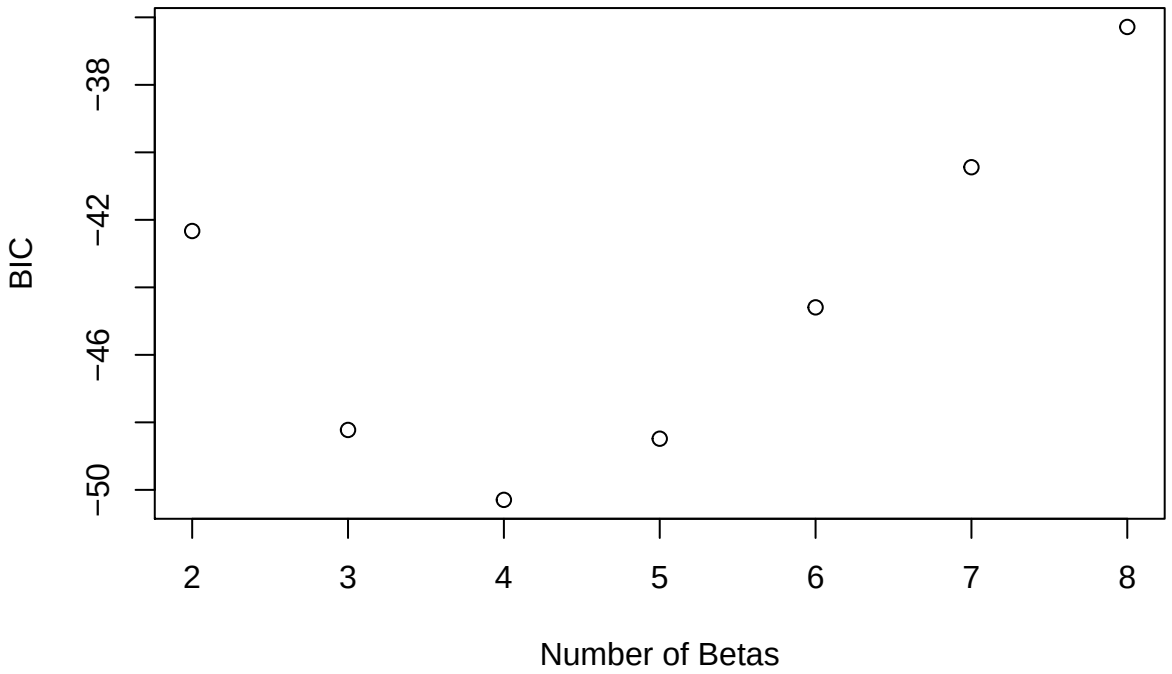



```
# MODEL SELECTED: 5 betas - 4 predictors - has smallest AIC
mod_AIC <- lm(log(Cases) ~ Red.Blue.State + Percent.Men.Greek.Life + Undergrad.Enrollment + Type.of.Uni

summary(mod_AIC)
```

```
##
## Call:
## lm(formula = log(Cases) ~ Red.Blue.State + Percent.Men.Greek.Life +
##     Undergrad.Enrollment + Type.of.University, data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.84633 -0.36293  0.07315  0.30047  1.22472
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              5.552e+00  6.462e-01   8.593 8.26e-12 ***
## Red.Blue.State          -3.759e-01  1.674e-01  -2.245  0.02873 *
## Percent.Men.Greek.Life   3.946e+00  1.161e+00   3.399  0.00125 **
## Undergrad.Enrollment     4.728e-05  1.057e-05   4.474 3.80e-05 ***
## Type.of.University       4.142e-01  2.864e-01   1.446  0.15368
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6222 on 56 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.4329, Adjusted R-squared:  0.3924
## F-statistic: 10.69 on 4 and 56 DF,  p-value: 1.659e-06
```

the sub model chosen by BIC

```
BIC_all <- n * log(RSS_all/n) + log(n) * p_all
plot(x = p_all, y = BIC_all,
     xlab="Number of Betas",
     ylab="BIC")
```




```
#Model Selected: 4 betas - 3 predictors - Has smallest BIC
mod_BIC <- lm(log(Cases) ~ Red.Blue.State + Percent.Men.Greek.Life + Undergrad.Enrollment, data = train

summary(mod_BIC)
```

```
##
## Call:
## lm(formula = log(Cases) ~ Red.Blue.State + Percent.Men.Greek.Life +
##     Undergrad.Enrollment, data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8542 -0.3613  0.1136  0.3145  1.3126
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             6.246e+00  4.374e-01  14.280  < 2e-16 ***
## Red.Blue.State         -4.051e-01  1.678e-01  -2.414  0.01901 *
## Percent.Men.Greek.Life  3.445e+00  1.118e+00   3.080  0.00318 **
## Undergrad.Enrollment    5.503e-05  9.193e-06   5.986 1.51e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6281 on 57 degrees of freedom
##   (3 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.4117, Adjusted R-squared:  0.3808
## F-statistic:  13.3 on 3 and 57 DF,  p-value: 1.086e-06
```

Support Vector Regression - Log transformed

```
# Splitting the dataset into the Training set and Test set
library(caTools)
library(e1071)
supportVec = svm(formula = log(Cases) ~ Red.Blue.State + Fall.Fan.Sports.Policy + Percent.Men.Greek.Life

supportVec
```

```
##
## Call:
## svm(formula = log(Cases) ~ Red.Blue.State + Fall.Fan.Sports.Policy +
##     Percent.Men.Greek.Life + Undergrad.Enrollment + Type.of.University +
##     Area + US.Region, data = training_set, type = "eps-regression")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1428571
##     epsilon:  0.1
##
##
## Number of Support Vectors:  51
```

```
summary(supportVec)
```

```
##
## Call:
## svm(formula = log(Cases) ~ Red.Blue.State + Fall.Fan.Sports.Policy +
##     Percent.Men.Greek.Life + Undergrad.Enrollment + Type.of.University +
##     Area + US.Region, data = training_set, type = "eps-regression")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1428571
##     epsilon:  0.1
##
##
## Number of Support Vectors:  51
```

```
y_pred3 = predict(supportVec, newdata = test_set)
y_pred3
```

```
##        4        5        8       11       16       20       21       24
## 8.203088 7.753174 7.182472 7.191245 6.937480 7.856334 7.574213 7.140196
##       31       32       34       50       53       59       65       67
```

```
## 7.626538 8.048044 7.763510 8.383537 7.852178 8.036168 7.968251 7.597196
##       68
## 8.468959
```

Random Forest Regression - Log transformed

```
# Splitting the dataset into the Training set and Test set

library(caTools)

# Fitting the Random Forest Regression to the dataset

library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
#third arguement will be number of trees
#randFor = randomForest(formula = log(Cases) ~ Red.Blue.State + Fall.Fan.Sports.Policy + Percent.Men.Gr

#randFor
#summary(randFor)

#y_pred4 = predict(randFor, newdata = test_set)
#y_pred4
```

For each model, calculate the "mean prediction error" in the testing dateset.

MSE for Full Model

```
k = 7
y_pred_full = predict(mod_log, newdata = test_set)
test = test_set$Cases
numerator = sum((test - y_pred_full)^2)
MSE_FULL = numerator/(n - k - 1)
MSE_FULL
```

```
## [1] 14381949
```

MSE for AIC and Adjusted R-Squared

```
k = 4
y_pred_AIC = predict(mod_AIC, newdata = test_set)
test = test_set$Cases
numerator = sum((test - y_pred_AIC)^2)
MSE_AIC = numerator/(n - k - 1)
MSE_AIC
```

```
## [1] 10786432
```

MSE for BIC

```
k = 3
y_pred_BIC = predict(mod_BIC, newdata = test_set)
test = test_set$Cases
numerator = sum((test - y_pred_BIC)^2)
MSE_BIC = numerator/(n - k - 1)
MSE_BIC
```

```
## [1] 9956345
```

MSE for Support Vector Regression

```
k = 7
y_pred_supportVec = predict(supportVec, newdata = test_set)
test = test_set$Cases
numerator = sum((test - y_pred_supportVec)^2)
MSE_SUPPORTVEC = numerator/(n - k - 1)
MSE_SUPPORTVEC
```

```
## [1] 14380474
```

MSE for Random Regression

```
#k = 7
#y_pred_randfor = predict(randFor, newdata = test_set)
#test = test_set$Cases
#numerator = sum((test - y_pred_randfor)^2)
#MSE_RANDFOR = numerator/(n - k - 1)
#MSE_RANDFOR
```

```
# Fitting ANN to the Training set
# h2o package - open source software platform that can train deep learning model in seconds - like conn
#install.packages('h2o')
library(h2o)
```

```
## Warning: package 'h2o' was built under R version 4.0.2
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
```

```
## For more information visit https://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```
#h2o instance, we want to connnect to a server
#nthreads allows us to build the model, we need a lot of cores , -1 takes all the available cores your
h2o.init(nthreads = -1)
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         17 days 15 hours
##     H2O cluster timezone:       America/New_York
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    5 months and 26 days !!!
##     H2O cluster name:           H2O_started_from_R_chrisvo_huk890
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   2.00 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.0 (2020-04-24)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (5 months and 26 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
classifier = h2o.deeplearning(y = 'Cases',
                              #just a dataframe, but need to make it an h2o dataframe
                              training_frame = as.h2o(training_set),
                              activation = 'Rectifier',
                              # number of inputs reflects how many layers,
                              # deep learning so let's use at least 2 layers
                              # specify as a vector
                              # 4 nodes in each of the 2 hidden layers
                              hidden = c(4,4),
                              #epochs - number of times the dataset should be iterated
                              epochs = 100,
```

```
                                # -2 allows for some parameter tuning
                                train_samples_per_iteration = -2)
```

```
## Warning in use.package("data.table"): data.table cannot be used without R
## package bit64 version 0.9.7 or higher. Please upgrade to take advangage of
## data.table speedups.
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
# Predicting the Test set results
ANN_pred = h2o.predict(classifier, newdata = as.h2o(test_set[-9]))
```

```
## Warning in use.package("data.table"): data.table cannot be used without R
## package bit64 version 0.9.7 or higher. Please upgrade to take advangage of
## data.table speedups.
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
#y_pred3 = predict(classifier, newdata = test_set)
ANN_pred
```

```
##     predict
## 1 6282.7421
## 2 2537.2475
## 3  912.1603
## 4 1217.5998
## 5  334.4092
## 6 3207.9947
##
## [17 rows x 1 column]
```

```
n = 17
k = 7
test = test_set$Cases
numerator = sum((test - ANN_pred)^2)
MSE_ANN = numerator/(n - k - 1)
MSE_ANN
```

```
## [1] 6669660
```

```
#h2o.shutdown()
```