White Wine Quality Analysis

Abhishek Ramesh

November 28, 2020

Setting up libraries and dataset

```
  #Libraries Imported
  library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
  library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-3
```

```
  library(ISLR)
  library(leaps)
  library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
  library(lmtest)
```

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

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

```
## The following objects are masked from 'package:base':
##
```

```
##     as.Date, as.Date.numeric
```

```r
#Cleaning Dataset
Wine_Quality <- read.csv(
  "/Users/abhishekramesh/Downloads/WineQT.csv",
  header = TRUE, sep = ",")

n <- 1
for (i in colnames(Wine_Quality)){
  #If more than 30% of the data in a column is missing, drop the column
  Wine_Quality[[i]][Wine_Quality[[i]] == ''] <- NA
  na_vals <- sum(is.na(Wine_Quality[[i]]))
  if (na_vals > 0.3*nrow(Wine_Quality)){
    Wine_Quality <- subset(Wine_Quality, select = -c(n))
  }

  n <- n + 1
}

#Drops first column, since it is not a predictor
Wine_Quality <- Wine_Quality[-c(1)]

#Drop rows with NA values
Wine_Quality <- na.omit(Wine_Quality)

#Number of Parameters
numX <- ncol(Wine_Quality)-1
```

Regression Tests

```r
#Setting the dataset and values
#-c(val) should be the response value
x <<- subset(Wine_Quality, select = -c(quality))
x <<- data.matrix(x)
y <<- Wine_Quality$quality

grid.lambda <- 10^seq(10, -2, length=100)

#80% of the sample size for training set
train <<- sample(1:nrow(Wine_Quality), 0.8 * nrow(Wine_Quality))
test <<- setdiff(1:nrow(Wine_Quality), train)

X.train <<- data.matrix(Wine_Quality[train, -(ncol(Wine_Quality))])
Y.train <<- Wine_Quality[train, "quality"]

X.test <<- data.matrix(Wine_Quality[test, -(ncol(Wine_Quality))])
Y.test <<- Wine_Quality[test, "quality"]

trainset <<- Wine_Quality[train, ]
testset <<- Wine_Quality[test, ]
```

Ridge Model

```r
Ridgefunc <- function(){
  ridge.model <- glmnet(x, y, alpha = 0, lambda = grid.lambda, family = "gaussian")
```

```
  set.seed(123)
  ell2.norm <- numeric()
  for (i in 1:length(grid.lambda)){
    ell2.norm[i] <- sqrt(sum(coef(ridge.model)[-1, i]^2))
  }
  plot(x = grid.lambda, y = ell2.norm, xlab = expression(Lambda), ylab = "L2 Norm", xlim = c(10,10000))

  #Min MSPE
  ridge.model.train <- glmnet(X.train, Y.train, alpha=0, lambda = grid.lambda)
  cv.out <- cv.glmnet(X.train, Y.train, alpha=0, family = "gaussian")
  best.lambda <- cv.out$lambda.min
  best.lambda
  plot(cv.out)
  abline(v = log(best.lambda), col="blue")

  #Final Model
  final.model <- glmnet(x, y, alpha = 0, lambda = best.lambda)
  Coef.Ridge <<- coef(final.model)[1:ncol(Wine_Quality), ]

  #Average MSPE
  ridge.pred <- predict(ridge.model.train, s = best.lambda, newx = X.test)
  mspe.ridge <<- mean((ridge.pred - Y.test)^2)
}
```

Lasso Model

```
Lassofunc <- function(){
  lasso.model <- glmnet(x, y, alpha = 1, lambda = grid.lambda, family = "gaussian")

  set.seed(123)
  ell2.norm <- numeric()
  for (i in 1:length(grid.lambda)){
    ell2.norm[i] <- sqrt(sum(coef(lasso.model)[-1, i]^2))
  }
  plot(x = grid.lambda, y = ell2.norm, xlab = expression(Lambda), ylab = "L2 Norm", xlim = c(10,10000))

  #Min MSPE
  lasso.model.train <- glmnet(X.train, Y.train, alpha=1, lambda = grid.lambda)
  cv.out <- cv.glmnet(X.train, Y.train, alpha=1, family = "gaussian")
  best.lambda <- cv.out$lambda.min
  plot(cv.out)
  abline(v = log(best.lambda), col="blue")

  #Final Model
  final.model <- glmnet(x, y, alpha = 1, lambda = best.lambda)
  Coef.lasso <<- coef(final.model)[1:ncol(Wine_Quality), ]

  #Average MSPE
  lasso.pred <- predict(lasso.model.train, s = best.lambda, newx = X.test)
  mspe.lasso <<- mean((lasso.pred - Y.test)^2)
}
```

OLS Model

```r
OLSfunc <- function(){
  OLS <<- lm(quality ~ . ,Wine_Quality)
  OLS_summary <- summary(OLS)
  OLS_summary

  OLS_residuals <<- OLS$residuals
  OLS_fittedvalues <<- OLS$fitted.values

  Coef.OLS <<- OLS_summary$coefficients[,1]
  Coef.OLS

  #Average MSPE Value
  mspe.OLS <<- mean((Y.test - predict.lm(OLS, testset))^2)
  mspe.OLS

  OLSList <- list(OLS_summary, Coef.OLS, mspe.OLS)
  return(OLSList)
}
```

Outliers/Influential Points

Hypothesis Testing

```r
HypothesisTesting <- function(){
  #Hypothesis testing for the columns
  ConfidenceInterval <- round(confint(OLS, c("fixed.acidity", "volatile.acidity"
                                             , "citric.acid", "residual.sugar", "chlorides"
                                             , "free.sulfur.dioxide", "total.sulfur.dioxide"
                                             , "density", "pH", "sulphates", "alcohol")
                                      , level = 0.95), 5) #95% Confidence Interval

  for (i in 1:numX) {
    if (!is.na(ConfidenceInterval[i, 1]) && !is.na(ConfidenceInterval[i, 2])) {
      if (ConfidenceInterval[i, 1] < 0 && ConfidenceInterval[i, 2] > 0) {
        cat(colnames(Wine_Quality)[i], ": Fail to reject null hypothesis at 95% Confidence Interval\n")
      } else {
        cat(colnames(Wine_Quality)[i], ": Reject null hypothesis at 95% Confidence Interval\n")
      }
    } else {
      cat(colnames(Wine_Quality)[i], ": Not applicable\n")
    }
  }


}
```

Model Selection

```r
ModelSelection <- function(){
  b <- regsubsets(quality ~ ., data=Wine_Quality)
  rs <- summary(b)
  lengthRS <- length(rs)

  AIC<-50*log(rs$rss/50) + (2:(lengthRS+1))*2
  plot(AIC~I(1:lengthRS), ylab="AIC", xlab="Number of Predictors")
```

```
  abline(v = which.min(AIC), col="red")

  plot(1:lengthRS, rs$adjr2, xlab = "Number of Predictors", ylab = "Adjusted R-Squared")
  abline(v = which.max(rs$adjr2), col="red")
  plot(1:lengthRS, rs$cp, xlab = "Number of Predictors", ylab = "Mallow's Cp")
  abline(v = which.min(rs$cp), col="red")
  plot(1:lengthRS, rs$bic, xlab = "Number of Predictors", ylab = "BIC")
  abline(v = which.min(rs$bic), col="red")

  cat("According to AIC, only", which.min(AIC), "predictors should be there\n")
  cat("According to Adjusted R-Squared, only", which.max(rs$adjr2), "predictors should be there\n")
  cat("According to Mallow's Cp, only", which.min(rs$cp), "predictors should be there\n")
  cat("According to BIC, only", which.min(rs$bic), "predictors should be there\n")
}
```

Formal F-tests

```
Ftest <- function(){
  #Since F-test, OLS variables
  OLS1 <- lm(quality ~ volatile.acidity, Wine_Quality)
  OLS2 <- lm(quality ~ volatile.acidity + residual.sugar, Wine_Quality)
  OLS3 <- lm(quality ~ volatile.acidity + residual.sugar + density, Wine_Quality)
  OLS4 <- lm(quality ~ volatile.acidity + residual.sugar + density + alcohol, Wine_Quality)
  OLS5 <- lm(quality ~ volatile.acidity + residual.sugar + density + alcohol + pH, Wine_Quality)


  varTesting <- var.test(OLS1, OLS2)
  if ((varTesting$conf.int[1]<1) && (varTesting$conf.int[2]>1)){
    cat("Fail to reject null hypothesis in F-test for:", names(OLS5$coefficients[2]), "\n")
    cat(names(OLS5$coefficients[2]), "and", names(OLS5$coefficients[3]), "have equal variance\n")
  } else {
    cat("Reject null hypothesis in F-test for:", names(OLS5$coefficients[2]), "\n")
  }

  if (varTesting$p.value < 0.05){
    cat("Reject null hypothesis and choose the first model in F-test\n")
  } else{
    cat("Fail to reject null hypothesis and choose the second model in F-test\n")
  }
}
```

Diagnostic Tests Linearity

```
Linearity <- function(){
  plot(x = OLS_residuals, y = OLS_fittedvalues, xlab = "Residuals", ylab = "Fitted Values") +
    abline(h = mean(OLS$fitted.values), col="Red")

  corResidualFitted <<- cor(OLS$fitted.values, OLS$residuals)

  if (corResidualFitted<0.15 && corResidualFitted>-0.15){
    print("Linearity Assumption is met")
    cat("Correlation between Residual and Fitted values: ", corResidualFitted, "\n")
  } else{
    print("Linear Assumption is not met")
    cat("Correlation between Residual and Fitted values: ", corResidualFitted, "\n")
```

```
  }
}
```

Homoscedasticity/Heteroscedasticity

```
Heteroscedasticity <- function(){
  bptestval <<- bptest(OLS)$p.value

  if (bptestval < 0.05){
    cat("Since p.value is small, Heteroscedasticity is proven for the dataset\n")
    cat("Breusch Pagan Test p.value = ", bptestval, "\n")
  } else {
    cat("Since p.value is not small enough, Homoscedasticity is proven for the dataset\n")
    cat("Breusch Pagan Test p.value = ", bptestval, "\n")
  }
}
```

Normality

```
Normality <- function(){
  set.seed(123)
  par(mfrow = c(1,2))
  qqnorm(OLS_residuals)
  hist(OLS_residuals, n=40)

  #Kolmogorov-Smirnov Test
  OLSresidual_mean <- mean(OLS_residuals)
  OLSresidual_sd <- sd(OLS_residuals)

  normal.sample <- rnorm(5000, mean = OLSresidual_mean, sd = OLSresidual_sd)
  pVal_kstest <<- ks.test(OLS_residuals, normal.sample)[2]

  if (pVal_kstest < 0.05){
    print("We reject the null hypothesis since p-value is not large enough.")
    print("Residuals are not normally distributed")
    paste("Pvalue:", pVal_kstest)
  } else{
    print("We fail to reject the null hypothesis since p-value is large.")
    paste("Pvalue:", pVal_kstest)
    print("Residuals are normally distributed")
  }
}
```

Transformations

```
Transformations <- function(){
  if (bptestval < 0.05){
    cat("Run variance stabalization, due to Heteroscedasticity in the dataset\n")
  }

  if (pVal_kstest < 0.05){
    cat("Apply Box-Cox transformation, due to the data not having normality\n")
  }

  if ((corResidualFitted > 0.15) | (corResidualFitted < (-0.15))){
   cat("Transform relationship between x and y with a quadratic relation,
```

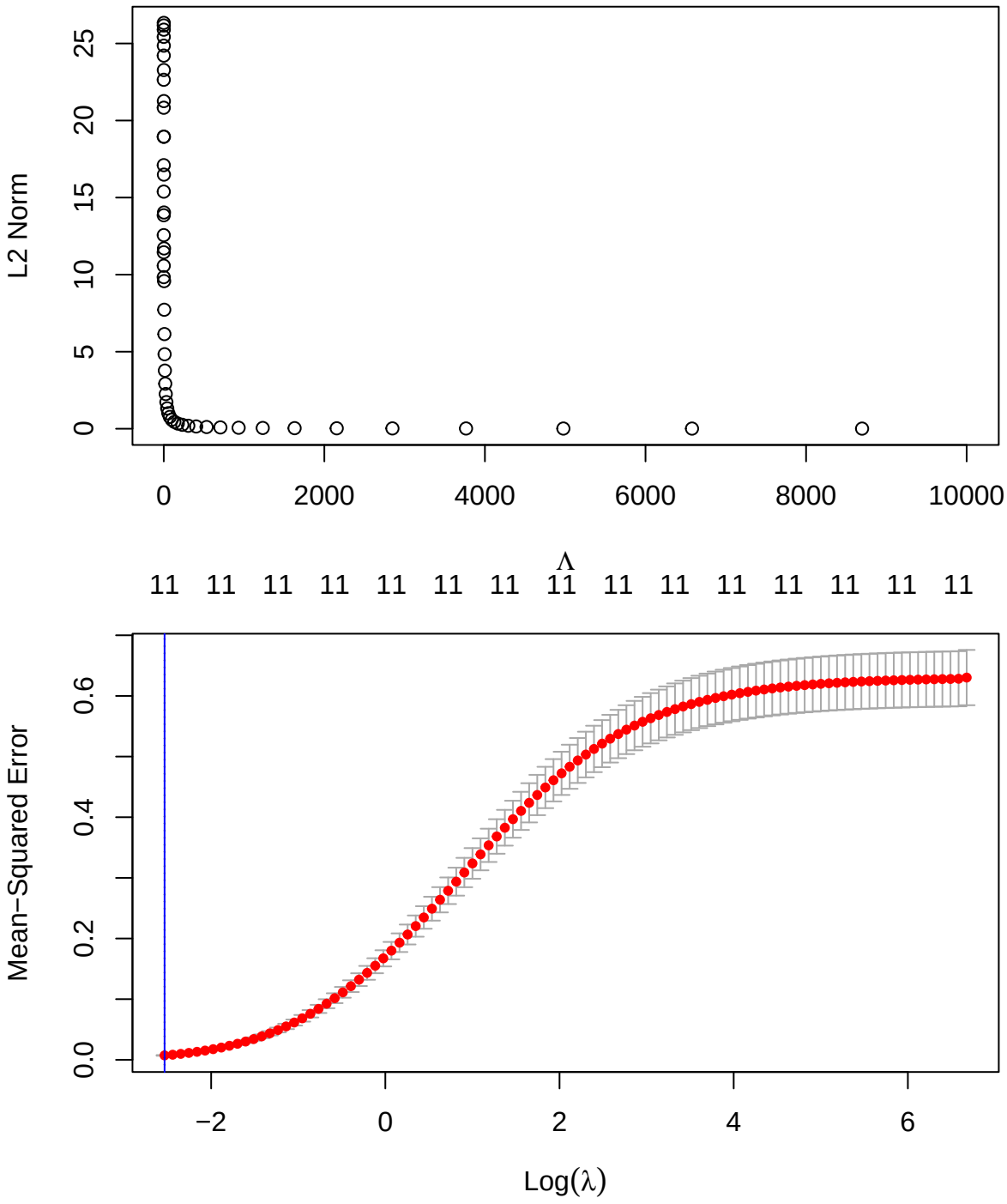
```
      since linearity assumption was not met\n")
  }

  #Look at the QQplot between theoretical quantiles and sample quantiles
  #since the tails are drifting to the side, apply log or squareroot transformation
}
```

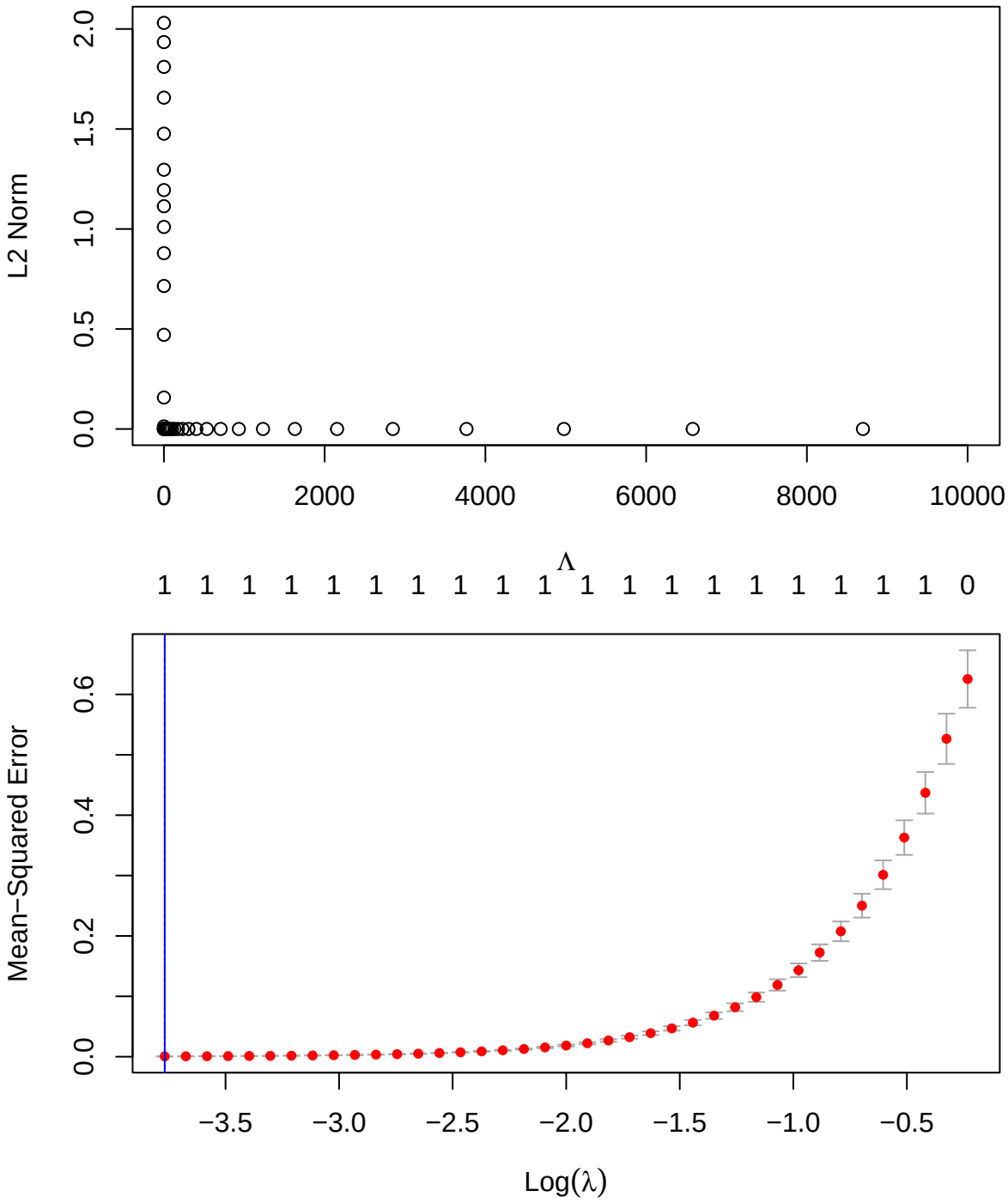
Functions

```
RegressionFunction <- function(){
  Ridgefunc()
  Lassofunc()
  OLSfunc()
  mspeVals <- c(mspe.OLS, mspe.lasso, mspe.ridge)
  mspeNames <- c("OLS", "Lasso", "Ridge")
  cat(mspeNames[which.min(mspeVals)], "is the best regression with a MSPE of",
      mspeVals[which.min(mspeVals)], "\n")
}

Diagnostic <- function(){
  Linearity()
  Heteroscedasticity()
  Normality()
}

Outliers <- function(){
  HypothesisTesting()
}
```

Overall Function

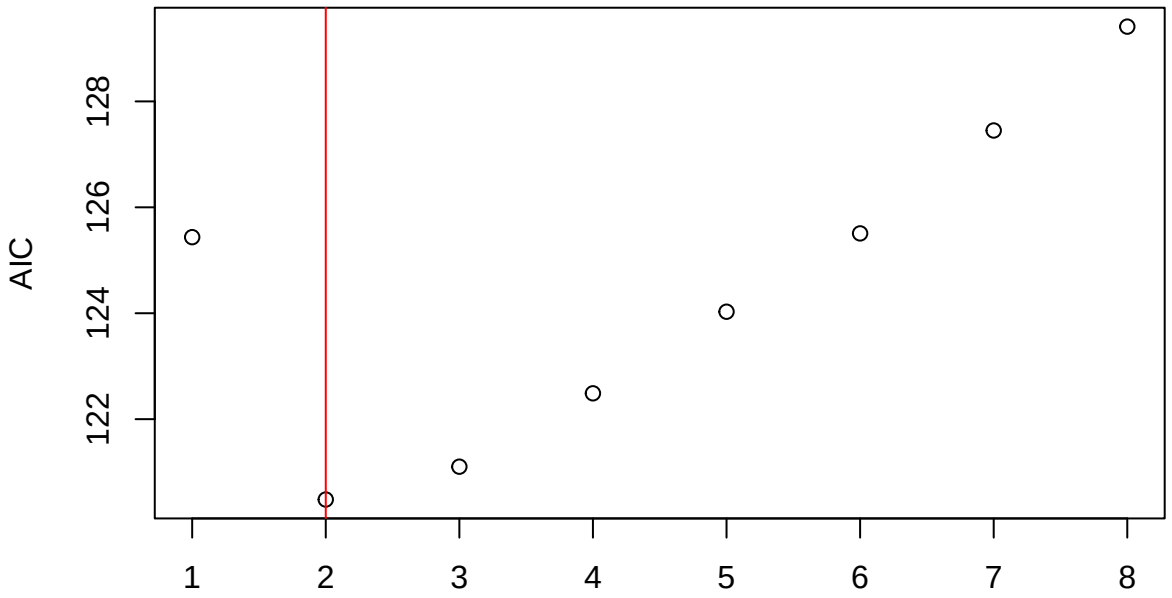
```
RegressionSoftware <- function(){
  RegressionFunction()
  Outliers()
  ModelSelection()
  Ftest()
  Diagnostic()
  Transformations()
}

RegressionSoftware()
```

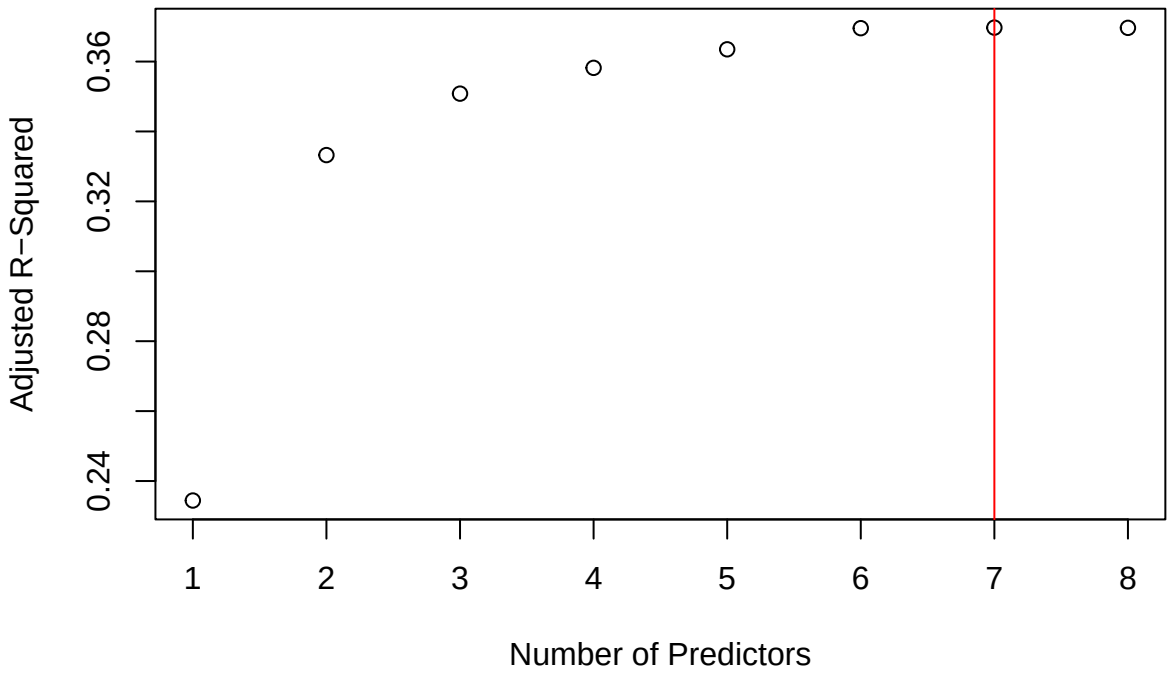

L2 Norm
0
5
10
15
20
25
0
2000
4000
6000
8000
10000
Λ
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
Mean−Squared Error
0.0
0.2
0.4
0.6
−2
0
2
4
6
Log(λ)

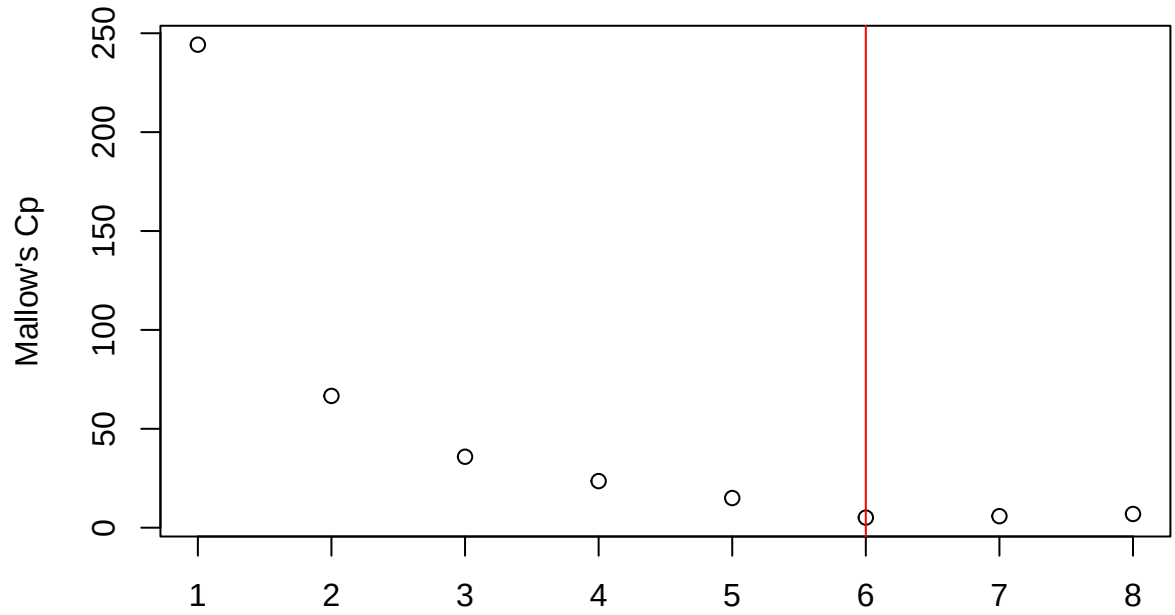


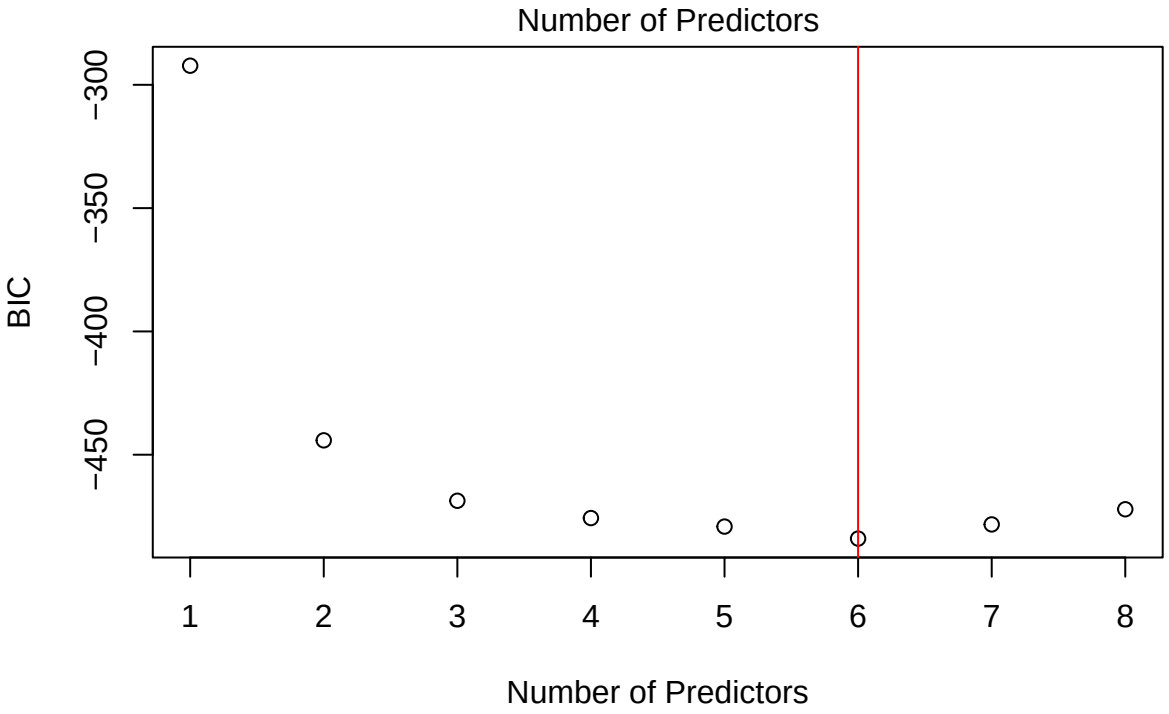

```
## Lasso is the best regression with a MSPE of 0.0006216088
## volatile.acidity : Not applicable
## citric.acid : Reject null hypothesis at 95% Confidence Interval
## residual.sugar : Fail to reject null hypothesis at 95% Confidence Interval
## chlorides : Fail to reject null hypothesis at 95% Confidence Interval
## free.sulfur.dioxide : Reject null hypothesis at 95% Confidence Interval
## total.sulfur.dioxide : Fail to reject null hypothesis at 95% Confidence Interval
## density : Reject null hypothesis at 95% Confidence Interval
## pH : Fail to reject null hypothesis at 95% Confidence Interval
## sulphates : Reject null hypothesis at 95% Confidence Interval
```

```
## alcohol : Reject null hypothesis at 95% Confidence Interval
## quality : Reject null hypothesis at 95% Confidence Interval
```

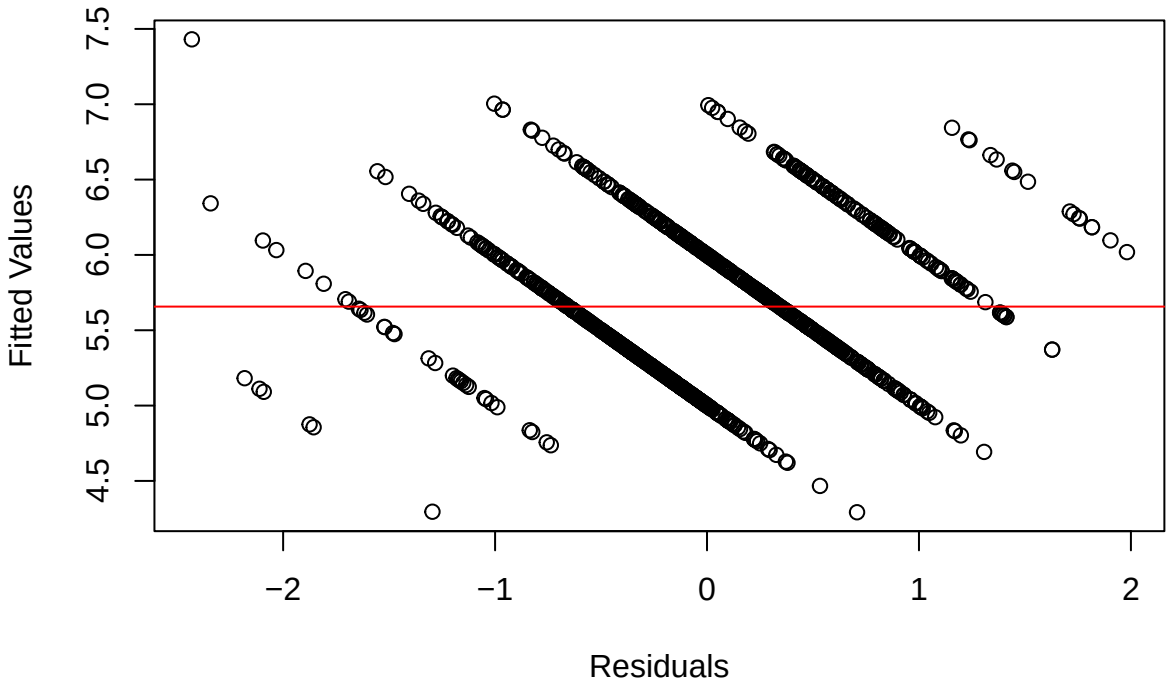




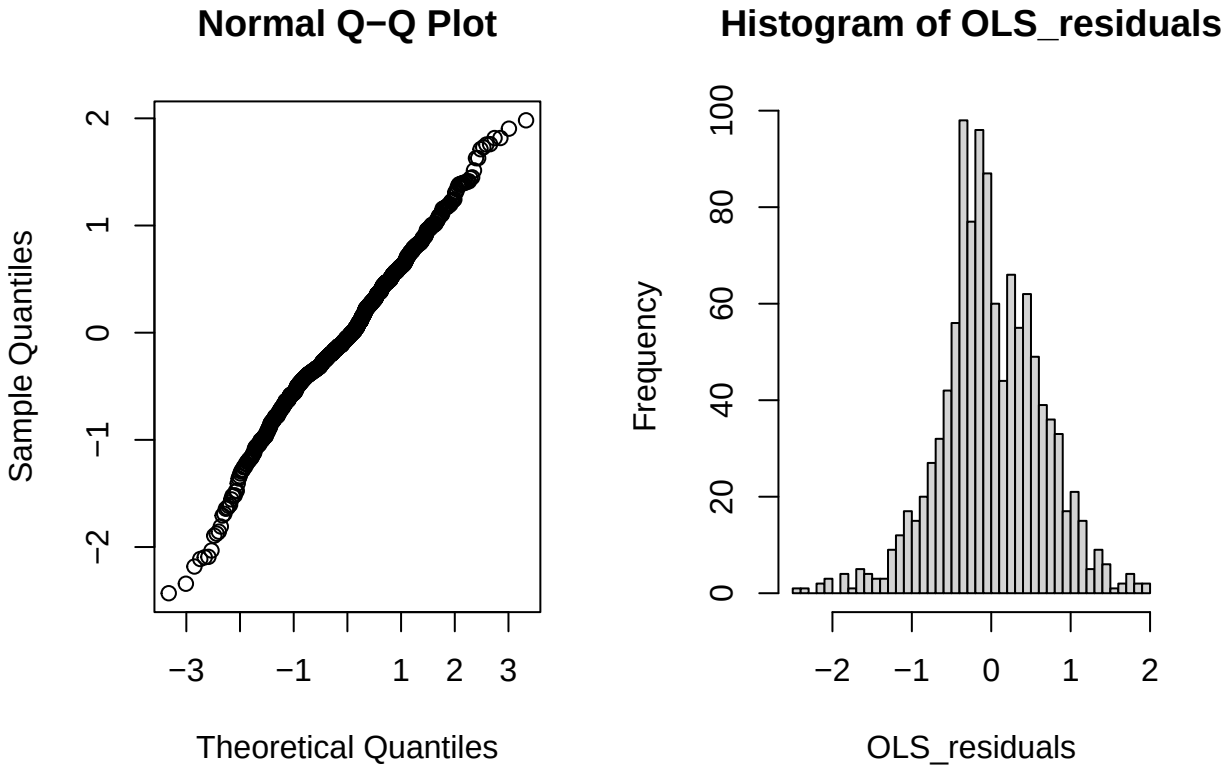





```
## According to AIC, only 2 predictors should be there
## According to Adjusted R-Squared, only 7 predictors should be there
## According to Mallow's Cp, only 6 predictors should be there
## According to BIC, only 6 predictors should be there
## Fail to reject null hypothesis in F-test for: volatile.acidity
## volatile.acidity and residual.sugar have equal variance
## Fail to reject null hypothesis and choose the second model in F-test
```




```
## [1] "Linearity Assumption is met"
## Correlation between Residual and Fitted values:  -1.020753e-17
## Since p.value is small, Heteroscedasticity is proven for the dataset
## Breusch Pagan Test p.value =  3.115096e-08
```




```
## [1] "We fail to reject the null hypothesis since p-value is large."
## [1] "Residuals are normally distributed"
## Run variance stabalization, due to Heteroscedasticity in the dataset
```

Summary of the data regression:

The best regression for this dataset was OLS with a MSPE of 0.5683808

All predictors are useful except: citric.acid, chlorides, total.sulfur.dioxide (thus giving 8 useful predictors)

AIC says to use 2 predictors, while Adjusted Rsquared, Cp, and BIC says to use the 8 predictors

Transformations to use on this dataset: Variance Stabalization and Box-Cox Transformation