Project # 1

Sonya Tahir

March 13, 2016

# Loading Training and Test Data

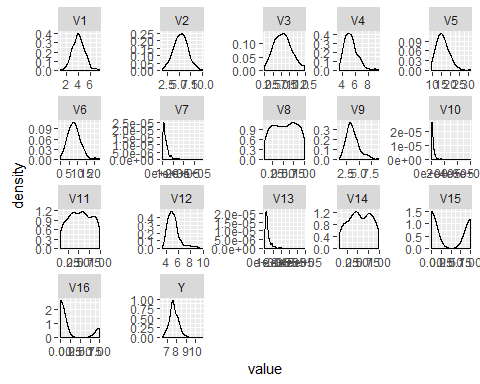
setwd("C:/Sonya/GW/Data Analysis/Assignments/Project01")  
trainingData <- read.csv("X.csv", header = TRUE, sep = " ")  
  
Y <- read.csv("Y.csv", header = TRUE, sep = " ")  
#trainingData <- within(trainingData,rm(Y))  
trainingData$Y <- Y$x  
rm(Y)  
  
testData <- read.csv("Xtest.csv", header = TRUE, sep = " ")

# Looking at the distributions of the variables and their correlations

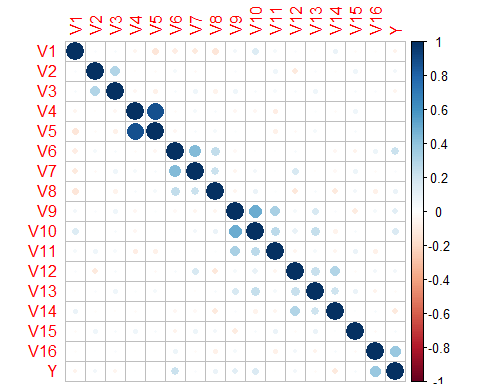
library(ggplot2)  
library(reshape)  
trainingData.melt = melt(trainingData)

## Using as id variables

ggplot(trainingData.melt,aes(x=value))+geom\_density()+facet\_wrap(~variable,scales="free")



library(corrplot)  
corrplot(cor(trainingData))

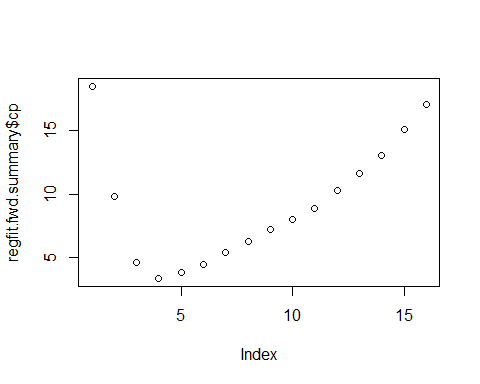


# Forward Stepwise Selection

library(leaps)  
regfit.fwd <- regsubsets(Y~.,trainingData,nvmax=16,method="forward")  
summary(regfit.fwd)

## Subset selection object  
## Call: regsubsets.formula(Y ~ ., trainingData, nvmax = 16, method = "forward")  
## 16 Variables (and intercept)  
## Forced in Forced out  
## V1 FALSE FALSE  
## V2 FALSE FALSE  
## V3 FALSE FALSE  
## V4 FALSE FALSE  
## V5 FALSE FALSE  
## V6 FALSE FALSE  
## V7 FALSE FALSE  
## V8 FALSE FALSE  
## V9 FALSE FALSE  
## V10 FALSE FALSE  
## V11 FALSE FALSE  
## V12 FALSE FALSE  
## V13 FALSE FALSE  
## V14 FALSE FALSE  
## V15 FALSE FALSE  
## V16 FALSE FALSE  
## 1 subsets of each size up to 16  
## Selection Algorithm: forward  
## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16  
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "\*"  
## 2 ( 1 ) " " " " " " " " " " "\*" " " " " " " " " " " " " " " " " " " "\*"  
## 3 ( 1 ) " " " " " " " " " " "\*" " " " " " " "\*" " " " " " " " " " " "\*"  
## 4 ( 1 ) " " " " " " " " " " "\*" " " " " " " "\*" " " " " " " "\*" " " "\*"  
## 5 ( 1 ) " " " " "\*" " " " " "\*" " " " " " " "\*" " " " " " " "\*" " " "\*"  
## 6 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " " " "\*" " " " " " " "\*" " " "\*"  
## 7 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " "\*" " " "\*"  
## 8 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " "\*" "\*" "\*"  
## 9 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 10 ( 1 ) " " "\*" "\*" " " " " "\*" "\*" " " "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 11 ( 1 ) " " "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 12 ( 1 ) "\*" "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 13 ( 1 ) "\*" "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"  
## 14 ( 1 ) "\*" "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"  
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"  
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

regfit.fwd.summary <- summary(regfit.fwd)  
  
plot(regfit.fwd.summary$cp)



regfit.fwd.summary$cp

## [1] 18.434575 9.803577 4.667887 3.391909 3.883427 4.502929 5.401741  
## [8] 6.299140 7.231571 8.000520 8.853650 10.243317 11.608166 13.057483  
## [15] 15.033046 17.000000

coef(regfit.fwd,4)

## (Intercept) V6 V10 V14 V16   
## 7.504421e+00 2.452135e-02 1.457402e-06 -2.017012e-01 4.610936e-01

# Linear Regression Model

library(boot)  
lm1 <- glm(Y~., data = trainingData)  
lm1.cv <- cv.glm(trainingData,lm1)  
lm1.mse <- lm1.cv$delta[1]  
lm1.mse

## [1] 0.241857

lm2 <- glm(Y~V6+V10+V14+V16, data = trainingData)  
lm2.cv <- cv.glm(trainingData,lm2)  
lm2.mse <- lm2.cv$delta[1]  
lm2.mse

## [1] 0.2241865

lm3 <- glm(Y~V6+V10+V14+V16+V1, data = trainingData)  
lm3.cv <- cv.glm(trainingData,lm3)  
lm3.mse <- lm3.cv$delta[1]  
lm3.mse

## [1] 0.224655

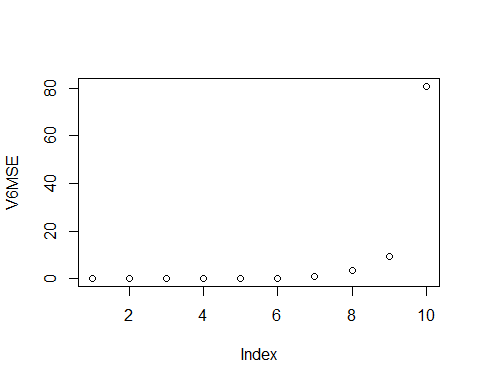
lm4 <- glm(Y~V6+V10+V14+V16+V2,data=trainingData)  
lm4.cv <- cv.glm(trainingData,lm4)  
lm4.mse <- lm4.cv$delta[1]  
lm4.mse

## [1] 0.2255125

I tried adding each variable one by one to the model with the four selected variables (V6, V10, V14 and V16) but none of them improved the mse.

# Higher Degree Polynomials

V6MSE <- rep(0,10)  
for(i in 1:10){  
 templm <- glm(Y~V10+V14+V16+poly(V6,i),data=trainingData)  
 tempCV <- cv.glm(trainingData,templm)  
 V6MSE[i] <- tempCV$delta[1]  
}  
plot(V6MSE)



which.min(V6MSE)

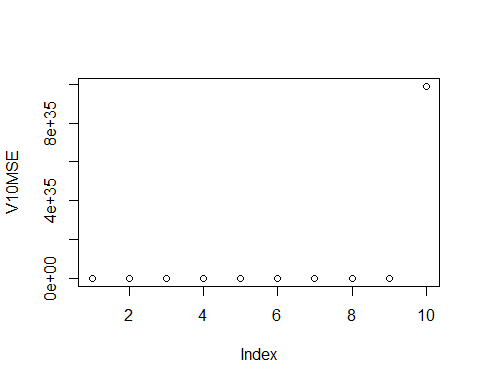
## [1] 2

V6MSE

## [1] 0.2241865 0.2236377 0.2263477 0.2411428 0.3202095 0.2836982  
## [7] 1.1713913 3.3291337 9.3839681 80.7724824

V6 = 2

V10MSE <- rep(0,10)  
for(i in 1:10){  
 templm <- glm(Y~V6+V14+V16+poly(V10,i),data=trainingData)  
 tempCV <- cv.glm(trainingData,templm)  
 V10MSE[i] <- tempCV$delta[1]  
}  
plot(V10MSE)



which.min(V10MSE)

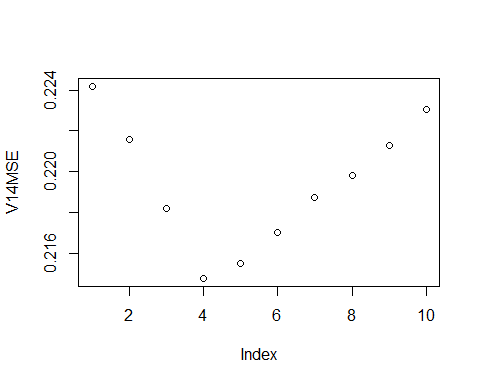
## [1] 2

V10MSE

## [1] 2.241865e-01 2.164234e-01 3.734898e-01 5.134418e+00 2.357648e+01  
## [6] 2.592440e+04 1.991549e+07 4.434826e+08 4.354071e+09 9.888519e+35

v10 = 2

V14MSE <- rep(0,10)  
for(i in 1:10){  
 templm <- glm(Y~V6+V10+V16+poly(V14,i),data=trainingData)  
 tempCV <- cv.glm(trainingData,templm)  
 V14MSE[i] <- tempCV$delta[1]  
}  
plot(V14MSE)



which.min(V14MSE)

## [1] 4

V14MSE

## [1] 0.2241865 0.2215748 0.2181805 0.2147552 0.2154761 0.2169900 0.2187195  
## [8] 0.2198148 0.2212589 0.2230338

v14 = 4

We cannot make polynomials of higher degree for V16 because degree must be less than the number of unique points and V16 only has two unique points, 0 and 1.

lm5 <- glm(Y~poly(V6,2)+poly(V10,2)+poly(V14,4)+V16,  
 data=trainingData)  
lm5.cv <- cv.glm(trainingData,lm5)  
lm5.mse <- lm5.cv$delta[1]  
lm5.mse

## [1] 0.2101609

mse is 0.2101 for lm5

# Splines

library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-9. For overview type 'help("mgcv-package")'.

model.s1 <- gam(Y~V6+V10+V14+V16, data = trainingData)  
summary(model.s1)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V6 + V10 + V14 + V16  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.504e+00 9.590e-02 78.249 < 2e-16 \*\*\*  
## V6 2.452e-02 7.806e-03 3.141 0.00189 \*\*   
## V10 1.457e-06 5.691e-07 2.561 0.01105 \*   
## V14 -2.017e-01 1.111e-01 -1.816 0.07060 .   
## V16 4.611e-01 7.286e-02 6.328 1.17e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
##   
## R-sq.(adj) = 0.193 Deviance explained = 20.6%  
## GCV = 0.21838 Scale est. = 0.21401 n = 250

library(gamclass)

## Warning: package 'gamclass' was built under R version 3.2.4

CVgam(formula(model.s1),data = trainingData)

## GAMscale CV-mse-GAM   
## 0.2140 0.2233

MSE 0.2233

model.s2 <- gam(Y~V6+s(V10)+V14+V16, data = trainingData)  
summary(model.s2)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V6 + s(V10) + V14 + V16  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.559184 0.092843 81.419 < 2e-16 \*\*\*  
## V6 0.024156 0.007737 3.122 0.00201 \*\*   
## V14 -0.213210 0.110230 -1.934 0.05424 .   
## V16 0.461426 0.072224 6.389 8.39e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(V10) 1.935 2.36 4.628 0.0079 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.208 Deviance explained = 22.4%  
## GCV = 0.21517 Scale est. = 0.21006 n = 250

CVgam(formula(model.s2),data = trainingData)

## GAMscale CV-mse-GAM   
## 0.2101 0.2186

Using spline for V6 alone did not reduce MSE. Using it on V10 did. So we will keep it. I also tried spline on V14 and V16 with no improvement in mse. Next, I tried using different values for k in spline function for each variable. The only combination that improved the mse was for V10 with k=3.

model.s3 <- gam(Y~V6+V7+s(V10,k=3)+V14+V16, data = trainingData)  
summary(model.s3)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V6 + V7 + s(V10, k = 3) + V14 + V16  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.558e+00 9.273e-02 81.502 < 2e-16 \*\*\*  
## V6 2.845e-02 8.588e-03 3.313 0.00106 \*\*   
## V7 -9.057e-07 7.767e-07 -1.166 0.24473   
## V14 -2.175e-01 1.100e-01 -1.976 0.04924 \*   
## V16 4.560e-01 7.219e-02 6.317 1.26e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(V10) 1.842 1.975 6.873 0.00308 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.211 Deviance explained = 22.9%  
## GCV = 0.21522 Scale est. = 0.20933 n = 250

CVgam(formula(model.s3),data = trainingData)

## GAMscale CV-mse-GAM   
## 0.2093 0.2173

0.2173

# Trying again with subset selection

#library(leaps)  
regfit.full <- regsubsets(Y~.,trainingData, nvmax = 16)  
summary(regfit.full)

## Subset selection object  
## Call: regsubsets.formula(Y ~ ., trainingData, nvmax = 16)  
## 16 Variables (and intercept)  
## Forced in Forced out  
## V1 FALSE FALSE  
## V2 FALSE FALSE  
## V3 FALSE FALSE  
## V4 FALSE FALSE  
## V5 FALSE FALSE  
## V6 FALSE FALSE  
## V7 FALSE FALSE  
## V8 FALSE FALSE  
## V9 FALSE FALSE  
## V10 FALSE FALSE  
## V11 FALSE FALSE  
## V12 FALSE FALSE  
## V13 FALSE FALSE  
## V14 FALSE FALSE  
## V15 FALSE FALSE  
## V16 FALSE FALSE  
## 1 subsets of each size up to 16  
## Selection Algorithm: exhaustive  
## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16  
## 1 ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "\*"  
## 2 ( 1 ) " " " " " " " " " " "\*" " " " " " " " " " " " " " " " " " " "\*"  
## 3 ( 1 ) " " " " " " " " " " "\*" " " " " " " "\*" " " " " " " " " " " "\*"  
## 4 ( 1 ) " " " " " " " " " " "\*" " " " " " " "\*" " " " " " " "\*" " " "\*"  
## 5 ( 1 ) " " " " "\*" " " " " "\*" " " " " " " "\*" " " " " " " "\*" " " "\*"  
## 6 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " " " "\*" " " " " " " "\*" " " "\*"  
## 7 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " "\*" " " "\*"  
## 8 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " "\*" "\*" "\*"  
## 9 ( 1 ) " " "\*" "\*" " " " " "\*" " " " " "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 10 ( 1 ) " " "\*" "\*" " " " " "\*" "\*" " " "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 11 ( 1 ) " " "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 12 ( 1 ) "\*" "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*" "\*"  
## 13 ( 1 ) "\*" "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"  
## 14 ( 1 ) "\*" "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"  
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"  
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

Order in which variables are selected is : V16, V6, V10, V14, V3, V2, V9, V15, V12, V7, V8, V1, V13, V11, V4, V5. Lets add these variables to the model in the same order. I checked mse by adding each variable one by one and then trying their polynomial version and with spline. Following is the best model that I got with this process.

model.all <- gam(Y~V16+V6+s(V10, k=2)+poly(V14,4)+poly(V3,4)+V7, data = trainingData)

## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible

summary(model.all)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V16 + V6 + s(V10, k = 2) + poly(V14, 4) + poly(V3, 4) + V7  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.444e+00 6.733e-02 110.560 < 2e-16 \*\*\*  
## V16 4.565e-01 7.146e-02 6.389 8.82e-10 \*\*\*  
## V6 2.806e-02 8.485e-03 3.307 0.00109 \*\*   
## poly(V14, 4)1 -8.158e-01 4.519e-01 -1.805 0.07232 .   
## poly(V14, 4)2 9.693e-01 4.517e-01 2.146 0.03292 \*   
## poly(V14, 4)3 -9.716e-01 4.553e-01 -2.134 0.03389 \*   
## poly(V14, 4)4 9.160e-01 4.520e-01 2.026 0.04385 \*   
## poly(V3, 4)1 -6.632e-01 4.506e-01 -1.472 0.14236   
## poly(V3, 4)2 -4.812e-01 4.493e-01 -1.071 0.28530   
## poly(V3, 4)3 -6.942e-01 4.563e-01 -1.521 0.12947   
## poly(V3, 4)4 4.294e-01 4.503e-01 0.954 0.34129   
## V7 -9.749e-07 7.630e-07 -1.278 0.20262   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(V10) 1.79 1.956 6.588 0.005 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.249 Deviance explained = 28.8%  
## GCV = 0.21077 Scale est. = 0.19914 n = 250

CVgam(formula(model.all),data = trainingData)

## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible

## GAMscale CV-mse-GAM   
## 0.1991 0.2122

0.2122

# Summary

Best model with lm using polynomials of higher degree was lm5 with mse 0.2101.

lm5 <- glm(Y~poly(V6,2)+poly(V10,2)+poly(V14,4)+V16,  
 data=trainingData)  
lm5.cv <- cv.glm(trainingData,lm5)  
lm5.mse <- lm5.cv$delta[1]  
lm5.mse

## [1] 0.2101609

summary(lm5)

##   
## Call:  
## glm(formula = Y ~ poly(V6, 2) + poly(V10, 2) + poly(V14, 4) +   
## V16, data = trainingData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.32204 -0.25067 -0.03745 0.23436 2.84125   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.63551 0.03193 239.129 < 2e-16 \*\*\*  
## poly(V6, 2)1 1.32883 0.45629 2.912 0.00393 \*\*   
## poly(V6, 2)2 0.46281 0.45721 1.012 0.31244   
## poly(V10, 2)1 1.14752 0.45177 2.540 0.01172 \*   
## poly(V10, 2)2 -0.90004 0.45621 -1.973 0.04966 \*   
## poly(V14, 4)1 -0.91252 0.45205 -2.019 0.04464 \*   
## poly(V14, 4)2 0.93584 0.45234 2.069 0.03963 \*   
## poly(V14, 4)3 -1.00405 0.45656 -2.199 0.02882 \*   
## poly(V14, 4)4 0.81974 0.45310 1.809 0.07167 .   
## V16 0.45267 0.07132 6.347 1.08e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.2019767)  
##   
## Null deviance: 66.028 on 249 degrees of freedom  
## Residual deviance: 48.474 on 240 degrees of freedom  
## AIC: 321.36  
##   
## Number of Fisher Scoring iterations: 2

Best model with splines was model.s3 with mse 0.2172

model.s3 <- gam(Y~V6+s(V10,k=3)+V14+V16, data = trainingData)  
summary(model.s3)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V6 + s(V10, k = 3) + V14 + V16  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.561008 0.092767 81.505 < 2e-16 \*\*\*  
## V6 0.024075 0.007729 3.115 0.00206 \*\*   
## V14 -0.215232 0.110107 -1.955 0.05175 .   
## V16 0.461038 0.072118 6.393 8.21e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(V10) 1.835 1.973 6.569 0.00411 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.209 Deviance explained = 22.5%  
## GCV = 0.21469 Scale est. = 0.20967 n = 250

CVgam(formula(model.s3),data = trainingData)

## GAMscale CV-mse-GAM   
## 0.2097 0.2172

Best model combining techniques was model.all with mse 0.2122.

model.all <- gam(Y~V16+V6+s(V10, k=2)+poly(V14,4)+poly(V3,4)+V7, data = trainingData)

## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible

summary(model.all)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V16 + V6 + s(V10, k = 2) + poly(V14, 4) + poly(V3, 4) + V7  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.444e+00 6.733e-02 110.560 < 2e-16 \*\*\*  
## V16 4.565e-01 7.146e-02 6.389 8.82e-10 \*\*\*  
## V6 2.806e-02 8.485e-03 3.307 0.00109 \*\*   
## poly(V14, 4)1 -8.158e-01 4.519e-01 -1.805 0.07232 .   
## poly(V14, 4)2 9.693e-01 4.517e-01 2.146 0.03292 \*   
## poly(V14, 4)3 -9.716e-01 4.553e-01 -2.134 0.03389 \*   
## poly(V14, 4)4 9.160e-01 4.520e-01 2.026 0.04385 \*   
## poly(V3, 4)1 -6.632e-01 4.506e-01 -1.472 0.14236   
## poly(V3, 4)2 -4.812e-01 4.493e-01 -1.071 0.28530   
## poly(V3, 4)3 -6.942e-01 4.563e-01 -1.521 0.12947   
## poly(V3, 4)4 4.294e-01 4.503e-01 0.954 0.34129   
## V7 -9.749e-07 7.630e-07 -1.278 0.20262   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(V10) 1.79 1.956 6.588 0.005 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.249 Deviance explained = 28.8%  
## GCV = 0.21077 Scale est. = 0.19914 n = 250

CVgam(formula(model.all),data = trainingData)

## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible

## GAMscale CV-mse-GAM   
## 0.1991 0.2122

# Ridge and LASSO

Now trying ridge

library(glmnet)

## Loading required package: Matrix

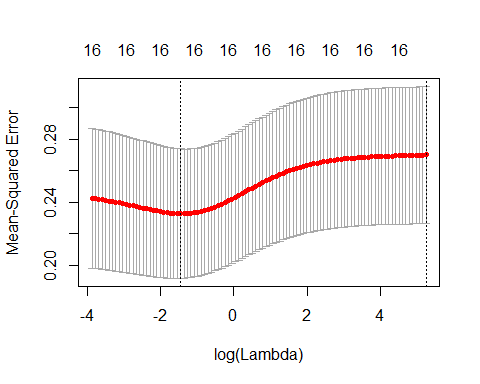
##   
## Attaching package: 'Matrix'

## The following object is masked from 'package:reshape':  
##   
## expand

## Loading required package: foreach

## Loaded glmnet 2.0-3

ridge.cv <- cv.glmnet(x=as.matrix(trainingData[,-17]),  
 y=as.matrix(trainingData[,17]),alpha=0)  
plot(ridge.cv)



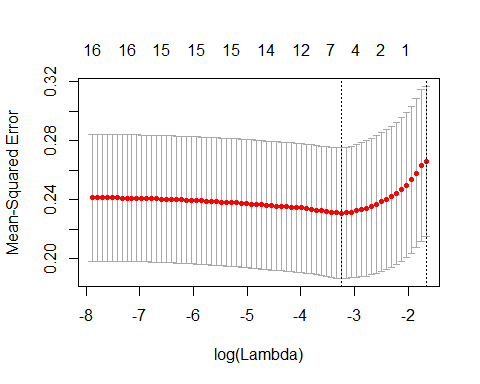
summary(ridge.cv)

## Length Class Mode   
## lambda 99 -none- numeric   
## cvm 99 -none- numeric   
## cvsd 99 -none- numeric   
## cvup 99 -none- numeric   
## cvlo 99 -none- numeric   
## nzero 99 -none- numeric   
## name 1 -none- character  
## glmnet.fit 12 elnet list   
## lambda.min 1 -none- numeric   
## lambda.1se 1 -none- numeric

Lowest mse is about 0.24. Not useful.

Next we will try lasso.

lasso.cv <- cv.glmnet(x=as.matrix(trainingData[,-17]),  
 y=as.matrix(trainingData[,17]),alpha=1)  
plot(lasso.cv)



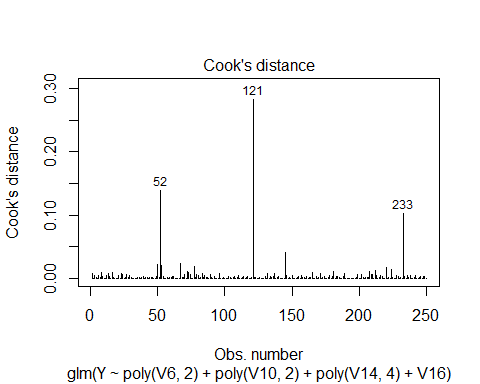
Lowest is about 0.24. Again not useful.

Therefore lm5 is the best model until now.

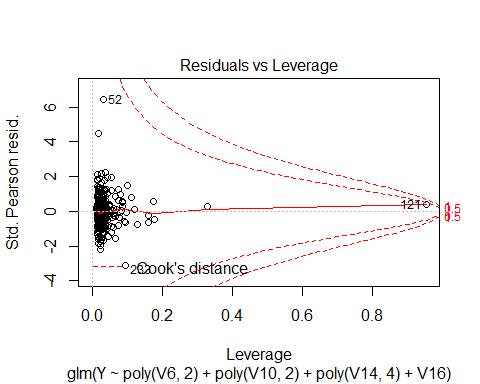
# Outliers

Checking outliers with lm5

plot(lm5,which = 4)



plot(lm5,which = 5)



Removing outliers 121, 52, 233

#taking backup of trainingData  
trainingDataBackup = trainingData  
#restoring backup  
#trainingData = trainingDataBackup  
  
ind <- c(121,52,233)  
trainingData = trainingData[-ind,]

Rechecking mse of all three models after outlier removal.

lm5 <- glm(Y~poly(V6,2)+poly(V10,2)+poly(V14,4)+V16,  
 data=trainingData)  
lm5.cv <- cv.glm(trainingData,lm5)  
lm5.mse <- lm5.cv$delta[1]  
lm5.mse

## [1] 0.1667576

0.1667

model.s3 <- gam(Y~V6+s(V10,k=3)+V14+V16, data = trainingData)  
summary(model.s3)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V6 + s(V10, k = 3) + V14 + V16  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.590190 0.084479 89.847 < 2e-16 \*\*\*  
## V6 0.018950 0.007025 2.698 0.00748 \*\*   
## V14 -0.224050 0.100164 -2.237 0.02621 \*   
## V16 0.499458 0.065722 7.600 6.5e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(V10) 1 1 13.54 0.000287 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.256 Deviance explained = 26.8%  
## GCV = 0.17468 Scale est. = 0.17114 n = 247

CVgam(formula(model.s3),data = trainingData)

## GAMscale CV-mse-GAM   
## 0.1711 0.1765

0.1765

model.all <- gam(Y~V16+V6+s(V10, k=2)+poly(V14,4)+poly(V3,4)+V7, data = trainingData)

## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible

summary(model.all)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Y ~ V16 + V6 + s(V10, k = 2) + poly(V14, 4) + poly(V3, 4) + V7  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.465e+00 6.072e-02 122.943 < 2e-16 \*\*\*  
## V16 4.908e-01 6.464e-02 7.592 7.47e-13 \*\*\*  
## V6 2.239e-02 7.700e-03 2.907 0.00400 \*\*   
## poly(V14, 4)1 -9.011e-01 4.048e-01 -2.226 0.02695 \*   
## poly(V14, 4)2 1.135e+00 4.049e-01 2.803 0.00549 \*\*   
## poly(V14, 4)3 -6.691e-01 4.080e-01 -1.640 0.10235   
## poly(V14, 4)4 1.165e+00 4.062e-01 2.869 0.00450 \*\*   
## poly(V3, 4)1 -5.837e-01 4.044e-01 -1.443 0.15029   
## poly(V3, 4)2 -1.570e-01 4.038e-01 -0.389 0.69772   
## poly(V3, 4)3 -4.654e-01 4.100e-01 -1.135 0.25748   
## poly(V3, 4)4 3.032e-01 4.035e-01 0.751 0.45311   
## V7 -6.933e-07 6.896e-07 -1.005 0.31573   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(V10) 1.116 1.218 11.11 0.000847 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.302 Deviance explained = 33.6%  
## GCV = 0.16967 Scale est. = 0.16066 n = 247

CVgam(formula(model.all),data = trainingData)

## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible  
  
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased to minimum possible

## GAMscale CV-mse-GAM   
## 0.1607 0.1671

0.1671

Therefore we choose lm5 as the best model with the least mse of 0.1667.

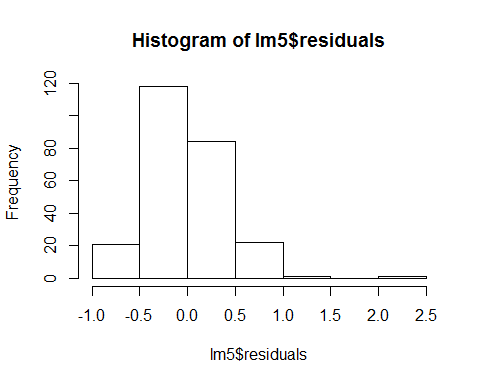
# Assumptions

library(car)

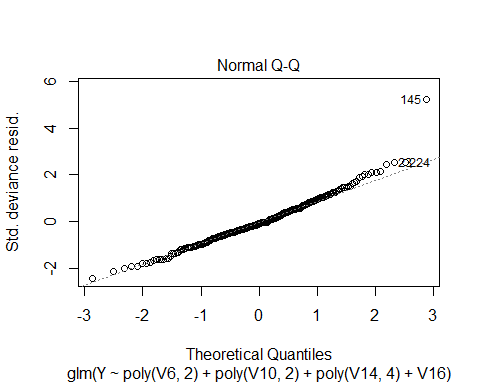
##   
## Attaching package: 'car'

## The following object is masked from 'package:boot':  
##   
## logit

hist(lm5$residuals)



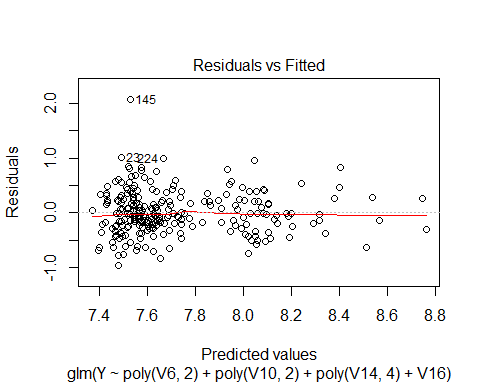
plot(lm5,which = 2)



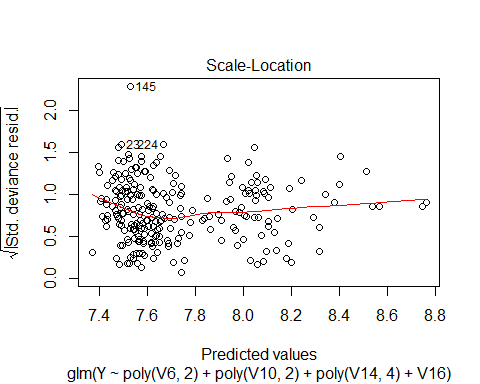
shapiro.test(lm5$residuals)

##   
## Shapiro-Wilk normality test  
##   
## data: lm5$residuals  
## W = 0.97028, p-value = 4.963e-05

#linearity  
plot(lm5,which = 1)



#Homoskedasticity  
plot(lm5,which=3)



From these, we can see that the residuals are not normal. 145 seems to be a clear outlier still. I will remove that as well and retry building model lm5.

#restoring backup  
trainingData = trainingDataBackup  
  
ind <- c(121,52,233,145)  
trainingData = trainingData[-ind,]

lm5 <- glm(Y~poly(V6,2)+poly(V10,2)+poly(V14,4)+V16,  
 data=trainingData)  
lm5.cv <- cv.glm(trainingData,lm5)  
lm5.mse <- lm5.cv$delta[1]  
lm5.mse

## [1] 0.1483644

The mse is 0.1483. Removing outlier 145 has resulted in significant improvement.

summary(lm5)

##   
## Call:  
## glm(formula = Y ~ poly(V6, 2) + poly(V10, 2) + poly(V14, 4) +   
## V16, data = trainingData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.94408 -0.22637 -0.02774 0.22411 1.03646   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.60773 0.02713 280.417 < 2e-16 \*\*\*  
## poly(V6, 2)1 0.98883 0.38463 2.571 0.010758 \*   
## poly(V6, 2)2 0.34858 0.38589 0.903 0.367284   
## poly(V10, 2)1 1.44314 0.38374 3.761 0.000214 \*\*\*  
## poly(V10, 2)2 -0.49173 0.38517 -1.277 0.202982   
## poly(V14, 4)1 -1.06438 0.38026 -2.799 0.005549 \*\*   
## poly(V14, 4)2 1.12842 0.38201 2.954 0.003455 \*\*   
## poly(V14, 4)3 -0.50314 0.38460 -1.308 0.192073   
## poly(V14, 4)4 1.21335 0.38308 3.167 0.001741 \*\*   
## V16 0.50770 0.06084 8.345 6.01e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.1434521)  
##   
## Null deviance: 53.041 on 245 degrees of freedom  
## Residual deviance: 33.855 on 236 degrees of freedom  
## AIC: 232.24  
##   
## Number of Fisher Scoring iterations: 2

Second degree of V6 is not significant. Lets try removing it.

lm5 <- glm(Y~V6+poly(V10,2)+poly(V14,4)+V16,  
 data=trainingData)  
lm5.cv <- cv.glm(trainingData,lm5)  
lm5.mse <- lm5.cv$delta[1]  
lm5.mse

## [1] 0.1481979

Removing second degree of V6 improves mse to 0.14819.

I tried removing the second degree of V10 also since it was not significant but mse increased to 0.1483. So i will keep that in.

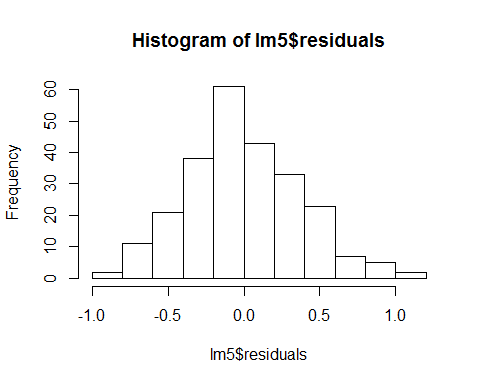
summary(lm5)

##   
## Call:  
## glm(formula = Y ~ V6 + poly(V10, 2) + poly(V14, 4) + V16, data = trainingData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.95594 -0.23342 -0.02659 0.23071 1.02916   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.475800 0.057063 131.011 < 2e-16 \*\*\*  
## V6 0.016818 0.006505 2.586 0.010319 \*   
## poly(V10, 2)1 1.489014 0.380216 3.916 0.000118 \*\*\*  
## poly(V10, 2)2 -0.504176 0.384776 -1.310 0.191360   
## poly(V14, 4)1 -1.053377 0.379921 -2.773 0.006003 \*\*   
## poly(V14, 4)2 1.125436 0.381843 2.947 0.003525 \*\*   
## poly(V14, 4)3 -0.474332 0.383126 -1.238 0.216920   
## poly(V14, 4)4 1.246049 0.381220 3.269 0.001241 \*\*   
## V16 0.505267 0.060754 8.317 7.13e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.1433407)  
##   
## Null deviance: 53.041 on 245 degrees of freedom  
## Residual deviance: 33.972 on 237 degrees of freedom  
## AIC: 231.09  
##   
## Number of Fisher Scoring iterations: 2

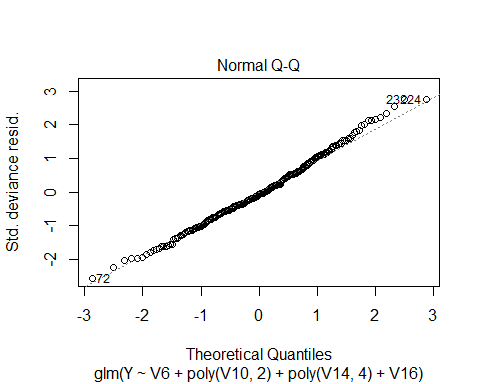
The summary now shows that each variable selected in the model is signifcant. V16 and V10 are the most significant variables in the model. The AIC of the model is 231.09.

Again checking assumptions

hist(lm5$residuals)



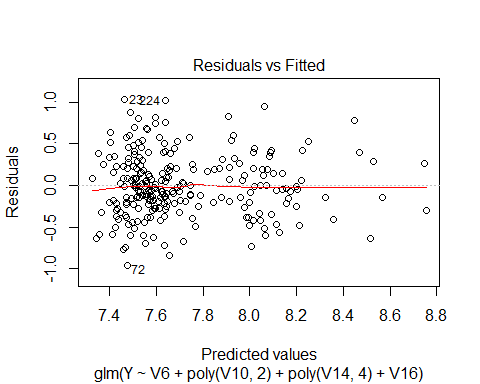
plot(lm5,which = 2)



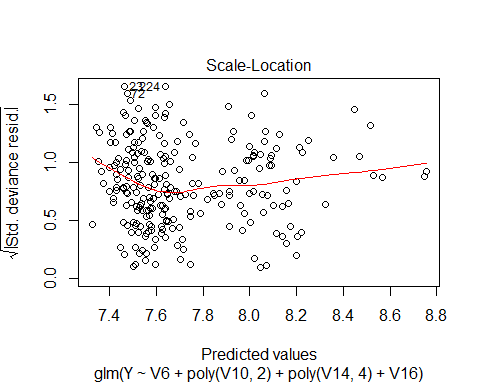
shapiro.test(lm5$residuals)

##   
## Shapiro-Wilk normality test  
##   
## data: lm5$residuals  
## W = 0.99336, p-value = 0.3453

#linearity  
plot(lm5,which = 1)



#Homoskedasticity  
plot(lm5,which=3)



The final model has an MSE of 0.14819. The residuals are normal as per the Shapiro-Wilk test. The model is not reasonably linear. The errors are not homoskedastic although the variation is not too large. The p-values for the individual variables can be considered an indicator to compare the variables. Since all assumptions of the linear model are not met (homoskedasticity is not met), we can not say that the p-values are meaningful.

To improve the prediction further, Greystone Broadcasting can gather more data about the variables that have been included in the model. They can also look at outliers to see if that was an error in data recording. There might be other errors in data which might go unnoticed if they are still within reasonable range of the data but they will influence the model to not be able to predict the target accurately. Greystone broadcasting can also start collecting other data variables. Sometimes unexpected relations exist which help predict the target even though we might not have thought of them as such.

# Hold out Data

In the beginning, I loaded the hold out data into testData. I will now use lm5 to predict target for the hold out data.

predictedY <- predict(lm5, newdata = testData)  
predictedY <- data.frame(predictedY)  
testData$Y <- predictedY$predictedY  
write.csv(testData, file = "testDataPredicted.csv")