R Notebook

# code to download ALSM  
#download.file("https://cran.r-project.org/src/contrib/Archive/ALSM/ALSM\_0.2.0.tar.gz", destfile = "ALSM\_0.2.0.tar.gz")  
#install.packages("ALSM\_0.2.0.tar.gz", repos = NULL, type = "source")  
#install.packages(c("leaps", "SuppDists"))  
library(ALSM)

## Loading required package: leaps

## Loading required package: SuppDists

## Loading required package: car

## Loading required package: carData

# code to download dplyr  
#download.file("https://cran.r-project.org/src/contrib/dplyr\_1.1.4.tar.gz", destfile = "dplyr\_1.1.4.tar.gz")  
#install.packages("dplyr\_1.1.4.tar.gz", repos = NULL, type = "source")  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:car':  
##   
## recode

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

# code to download MASS  
#download.file("https://cran.r-project.org/src/contrib/MASS\_7.3-61.tar.gz", destfile = "MASS\_7.3-61.tar.gz")  
#install.packages("MASS\_7.3-61.tar.gz", repos = NULL, type = "source")  
library(MASS)

##   
## Attaching package: 'MASS'

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

# code to download car  
#download.file("https://cran.r-project.org/src/contrib/car\_3.1-3.tar.gz", destfile = "car\_3.1-3.tar.gz")  
#install.packages("car\_3.1-3.tar.gz", repos = NULL, type = "source")  
library(car)  
# code to download lmtest  
#download.file("https://cran.r-project.org/src/contrib/lmtest\_0.9-40.tar.gz", destfile = "lmtest\_0.9-40.tar.gz")  
#install.packages("lmtest\_0.9-40.tar.gz", repos = NULL, type = "source")  
library(lmtest)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

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

# code to download boot  
#download.file("https://cran.r-project.org/src/contrib/boot\_1.3-31.tar.gz", destfile = "boot\_1.3-31.tar.gz")  
#install.packages("boot\_1.3-31.tar.gz", repos = NULL, type = "source")  
library(boot)

##   
## Attaching package: 'boot'

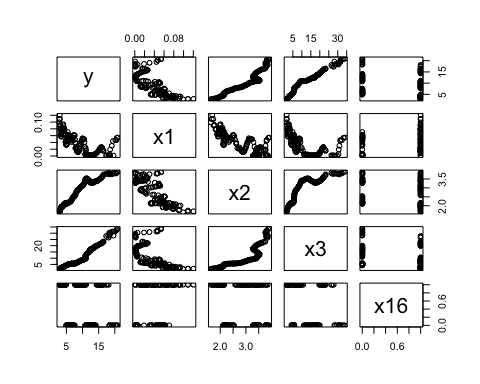
## The following object is masked from 'package:car':  
##   
## logit

# grab code that scaled the units to trillions from past file  
data = read.csv("~/Dropbox/Mac/Desktop/Main/Purdue/STAT 512/Project/FinalDataset.csv")  
df = data[,c('DPI','GFDEBTN', 'Government.Purchases','Nominal.Interest','Party.in.Office')]  
y = df$DPI / 1000 # convert to trillions  
x1 = df$Nominal.Interest / 100 # convert to raw num instead of %   
x2 = df$Government.Purchases / 1000 # convert to trillions  
x3 = df$GFDEBTN \* 1000000 / 1000000000000 # convert to trillions  
x16 = df$Party.in.Office  
newDF = data.frame(y=y, x1=x1, x2=x2, x3=x3, x16=x16)  
write.csv(newDF, "~/Dropbox/Mac/Desktop/Main/Purdue/STAT 512/Project/RealFinalDataset.csv")

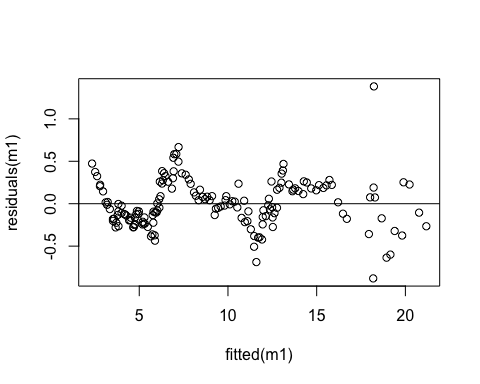
# read to new dataframe  
df = read.csv("~/Dropbox/Mac/Desktop/Main/Purdue/STAT 512/Project/RealFinalDataset.csv")  
df = df[2:6]  
# make x4 numeric: 0 = D, 1 = R  
df$x16 = ifelse(df$x16=='D',0,1)  
df$x16

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0  
## [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1  
## [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0  
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [149] 0 0 0 0 0 0 0 0 0 0 0 0

# assess intial full model   
m1 = lm(df$y~df$x1+df$x2+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16))  
# run diagnostics   
# scatter table  
pairs(df)



# residual plot  
plot(fitted(m1), residuals(m1))  
abline(h=0)



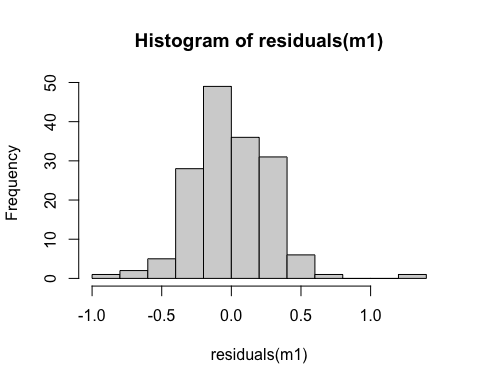
# bp test for non constant variance  
bptest(m1) # rerun with normal errors

##   
## studentized Breusch-Pagan test  
##   
## data: m1  
## BP = 8.2517, df = 7, p-value = 0.3109

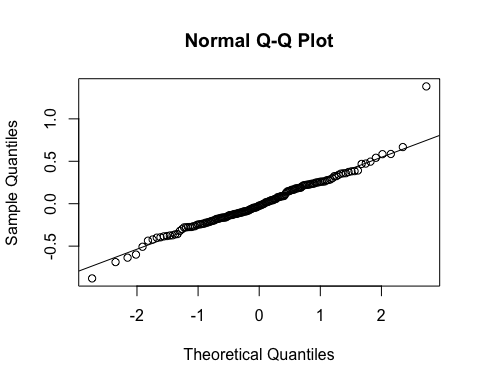
# shapiro test for normal errors  
shapiro.test(residuals(m1)) # says not normal

##   
## Shapiro-Wilk normality test  
##   
## data: residuals(m1)  
## W = 0.96595, p-value = 0.0005621

# qqplot, hist of residuals for normal errors  
hist(residuals(m1))



# QQ Plot of residuals  
qqnorm(residuals(m1))  
qqline(residuals(m1))

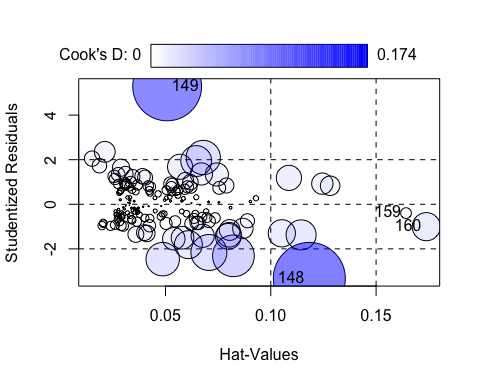


# VIF for multicollinearity  
vif(m1) # clearly some multicollinearity with x2 and x3

## there are higher-order terms (interactions) in this model  
## consider setting type = 'predictor'; see ?vif

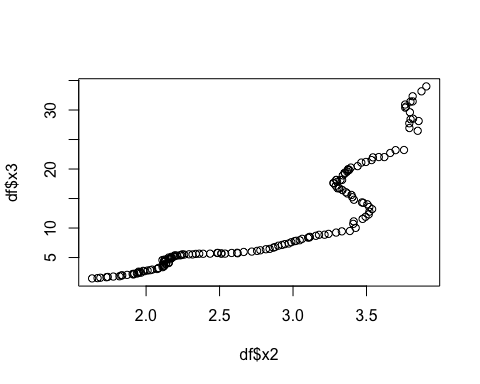
## df$x1 df$x2 df$x3 df$x16 df$x1:df$x16 df$x2:df$x16   
## 8.520576 27.529784 13.837204 167.266506 17.507039 186.081340   
## df$x3:df$x16   
## 15.627349

# influence plot - dffits and cook's dist for influential pts  
  
influencePlot(m1)

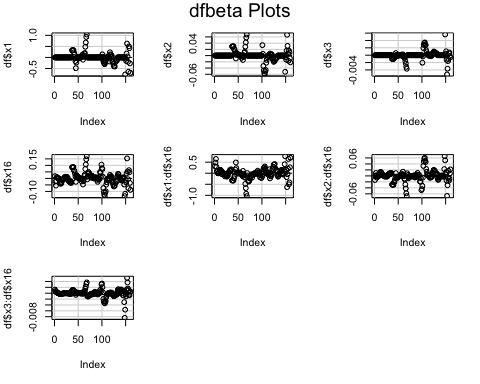


## StudRes Hat CookD  
## 148 -3.3243648 0.1182975 0.173848849  
## 149 5.2875534 0.0507813 0.158799467  
## 159 -0.3951232 0.1642145 0.003855758  
## 160 -1.0040328 0.1738515 0.026515686

# scatterplot to show relation between x2 and x3  
plot(df$x2, df$x3)

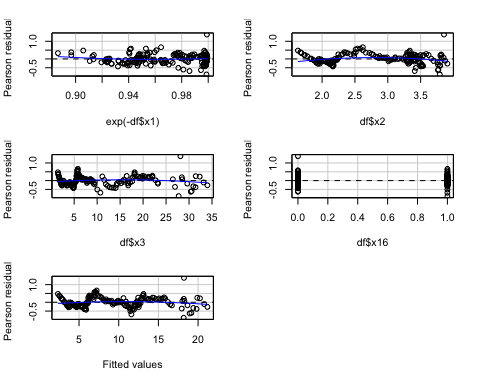


dfbetaPlots(m1)



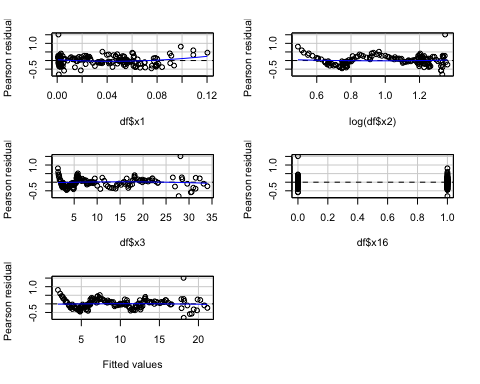
# there are maybe two points with minor influence but that's it.  
# so not gonna worry about that

# try to fix linearity  
# transform x1 by e^-x  
residualPlots(lm(df$y~exp(-df$x1)+df$x2+df$x3+df$x16+(exp(-df$x1)\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



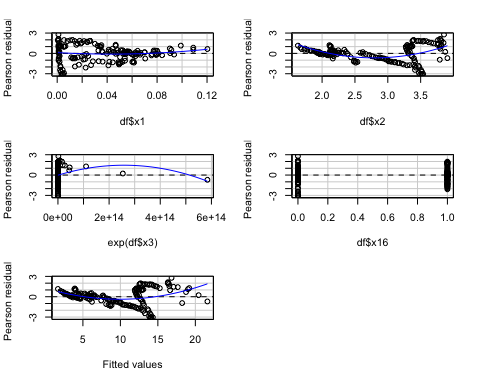
## Test stat Pr(>|Test stat|)   
## exp(-df$x1) 1.6473 0.1016   
## df$x2 -4.9242 2.198e-06 \*\*\*  
## df$x3 -4.2243 4.128e-05 \*\*\*  
## df$x16 -0.0349 0.9722   
## Tukey test -4.7190 2.370e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# did not help with overall linearity problem, but did help with x1  
# transform x2 by log(x2)  
residualPlots(lm(df$y~df$x1+log(df$x2)+df$x3+df$x16+(df$x1\*df$x16)+(log(df$x2)\*df$x16)+(df$x3\*df$x16)))



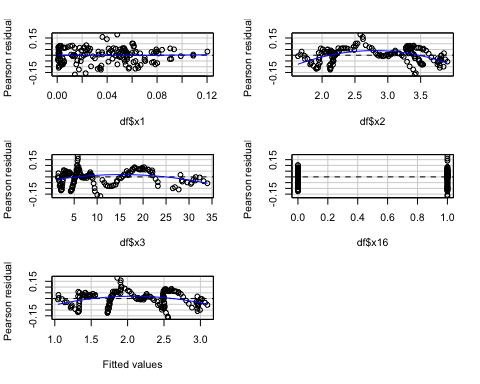
## Test stat Pr(>|Test stat|)   
## df$x1 3.1295 0.002102 \*\*  
## log(df$x2) 1.1004 0.272912   
## df$x3 -2.1181 0.035804 \*   
## df$x16 1.6677 0.097448 .   
## Tukey test -1.1893 0.234317   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# did not help with linearity problem overall or with x2  
# transform x3 by exp(x3)  
residualPlots(lm(df$y~df$x1+df$x2+exp(df$x3)+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(exp(df$x3)\*df$x16)))



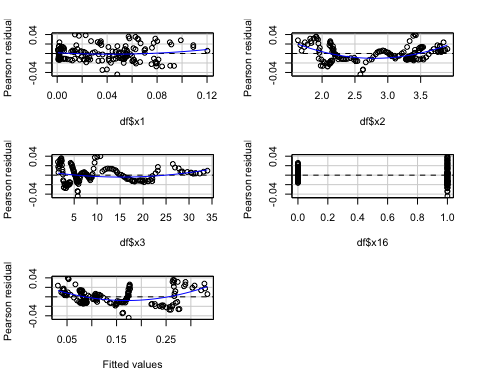
## Test stat Pr(>|Test stat|)   
## df$x1 1.6708 0.09683 .   
## df$x2 8.2115 9.03e-14 \*\*\*  
## exp(df$x3) -2.2191 0.02797 \*   
## df$x16 0.4647 0.64280   
## Tukey test 8.3134 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# made everything way worse  
# natural log y  
residualPlots(lm(log(df$y)~df$x1+df$x2+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



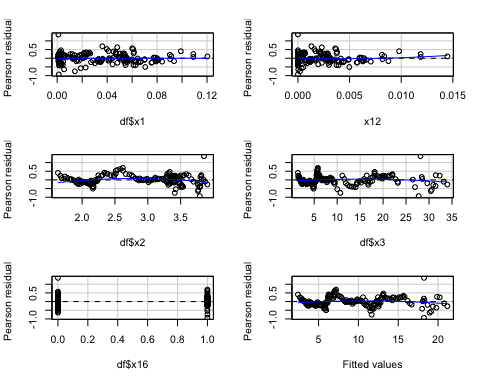
## Test stat Pr(>|Test stat|)   
## df$x1 0.1898 0.8497   
## df$x2 -22.0232 <2e-16 \*\*\*  
## df$x3 -10.2622 <2e-16 \*\*\*  
## df$x16 0.0136 0.9892   
## Tukey test -16.9674 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# made everything worse but made x1 much better  
# reciprocal y  
residualPlots(lm(1/(df$y)~df$x1+df$x2+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



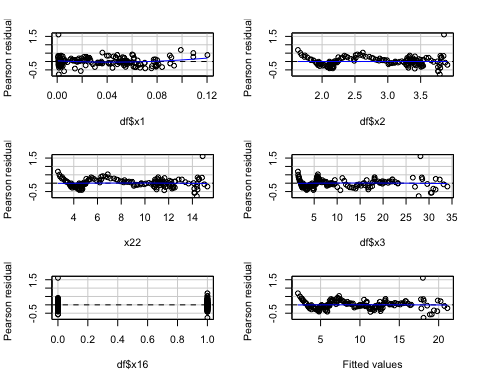
## Test stat Pr(>|Test stat|)   
## df$x1 1.7816 0.07682 .   
## df$x2 22.5616 < 2.2e-16 \*\*\*  
## df$x3 7.3300 1.298e-11 \*\*\*  
## df$x16 0.5671 0.57150   
## Tukey test 15.8851 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# nope, same as last try  
# add polynomial terms  
# add x12  
x12 = (df$x1)^2  
residualPlots(lm(df$y~df$x1+x12+df$x2+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



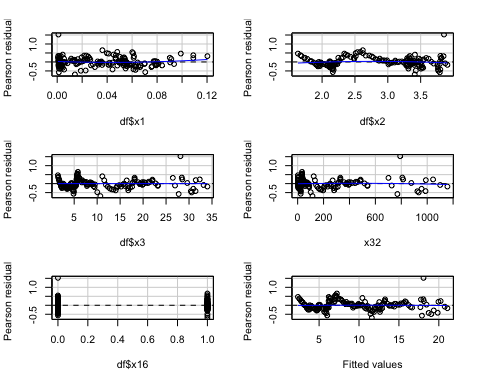
## Test stat Pr(>|Test stat|)   
## df$x1 -0.2548 0.79924   
## x12 2.0298 0.04415 \*   
## df$x2 -5.4275 2.248e-07 \*\*\*  
## df$x3 -4.1937 4.676e-05 \*\*\*  
## df$x16 0.2856 0.77560   
## Tukey test -4.8599 1.174e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# nope, did not help overall linearity   
# add x22  
x22 = (df$x2)^2  
residualPlots(lm(df$y~df$x1+df$x2+x22+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



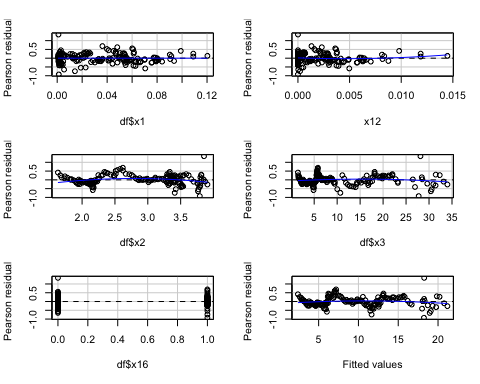
## Test stat Pr(>|Test stat|)   
## df$x1 2.6950 0.007841 \*\*  
## df$x2 0.4400 0.660566   
## x22 -1.8051 0.073069 .   
## df$x3 -1.2181 0.225086   
## df$x16 -0.3397 0.734595   
## Tukey test -0.8568 0.391559   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# also did not help overall linearity  
# add x32  
x32 = (df$x3)^2  
residualPlots(lm(df$y~df$x1+df$x2+df$x3+x32+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



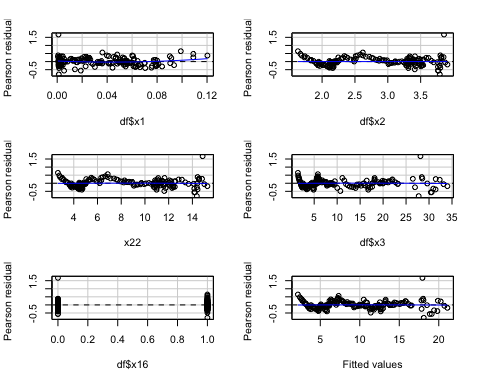
## Test stat Pr(>|Test stat|)   
## df$x1 1.6985 0.091489 .   
## df$x2 -2.7745 0.006233 \*\*  
## df$x3 -0.7362 0.462757   
## x32 -2.7198 0.007304 \*\*  
## df$x16 0.4015 0.688606   
## Tukey test -1.6231 0.104569   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# also did not help linearity  
# add x12 and x12x16  
x12x16 = x12 \* df$x16  
residualPlots(lm(df$y~df$x1+x12+df$x2+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)+(x12\*df$x16)))



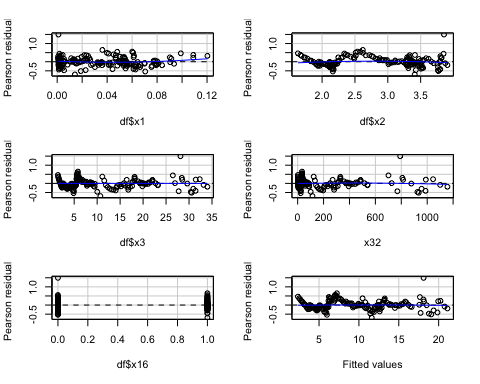
## Test stat Pr(>|Test stat|)   
## df$x1 -0.3545 0.723458   
## x12 2.8270 0.005345 \*\*   
## df$x2 -5.4139 2.414e-07 \*\*\*  
## df$x3 -4.3520 2.493e-05 \*\*\*  
## df$x16 0.2915 0.771092   
## Tukey test -4.9643 6.893e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# also did not help linearity  
# add x22 and x22x16  
x22x16 = x22 \* df$x16  
residualPlots(lm(df$y~df$x1+df$x2+x22+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)+(x22\*df$x16)))



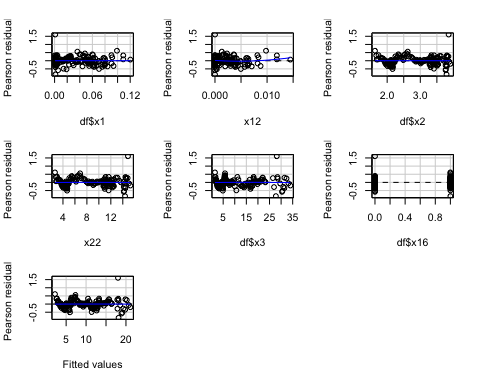
## Test stat Pr(>|Test stat|)   
## df$x1 2.4472 0.01556 \*  
## df$x2 0.4145 0.67908   
## x22 -1.4015 0.16315   
## df$x3 -0.7917 0.42979   
## df$x16 -0.2856 0.77556   
## Tukey test -0.9009 0.36765   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# did not help overall linearity  
# add x32 and x32x16  
x32x16 = x32 \* df$x16  
residualPlots(lm(df$y~df$x1+df$x2+df$x3+x32+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)+(x32\*df$x16)))



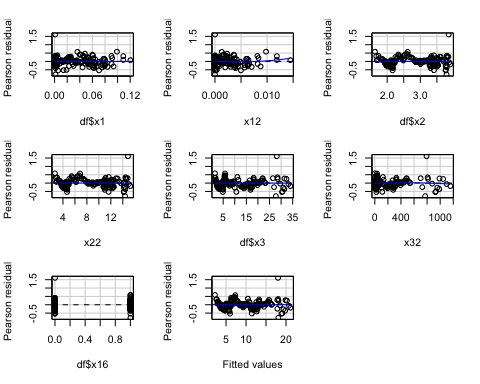
## Test stat Pr(>|Test stat|)   
## df$x1 2.1529 0.032935 \*   
## df$x2 -2.6913 0.007932 \*\*  
## df$x3 -0.7689 0.443198   
## x32 -3.1643 0.001885 \*\*  
## df$x16 0.4475 0.655134   
## Tukey test -1.6363 0.101781   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# add x12 and x22  
residualPlots(lm(df$y~df$x1+x12+df$x2+x22+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



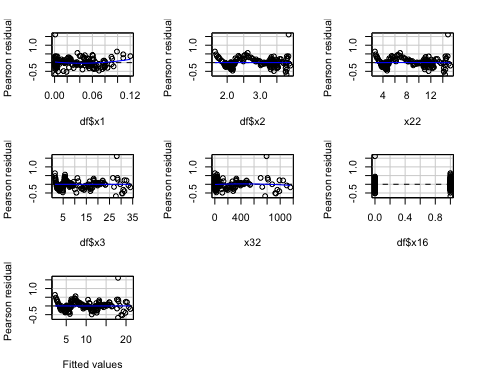
## Test stat Pr(>|Test stat|)   
## df$x1 0.0295 0.97652   
## x12 2.5876 0.01062 \*   
## df$x2 2.7732 0.00626 \*\*  
## x22 -0.9062 0.36628   
## df$x3 -0.7596 0.44870   
## df$x16 -0.2609 0.79449   
## Tukey test -0.3920 0.69503   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# did not help  
# add x12 and x22 and x32  
residualPlots(lm(df$y~df$x1+x12+df$x2+x22+df$x3+x32+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



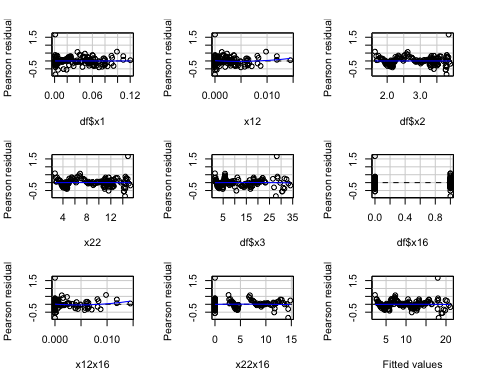
## Test stat Pr(>|Test stat|)   
## df$x1 -0.0168 0.9866156   
## x12 2.5969 0.0103558 \*   
## df$x2 2.7772 0.0061933 \*\*   
## x22 -0.5334 0.5945675   
## df$x3 0.4796 0.6321872   
## x32 -3.6584 0.0003522 \*\*\*  
## df$x16 1.0836 0.2803014   
## Tukey test 0.9540 0.3400643   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# did not help linearity  
# add x22 and x32  
residualPlots(lm(df$y~df$x1+df$x2+x22+df$x3+x32+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)))



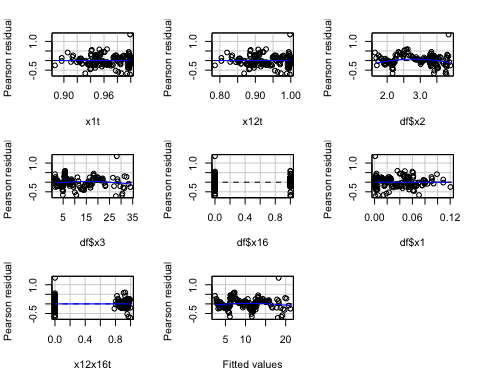
## Test stat Pr(>|Test stat|)   
## df$x1 2.5067 0.0132595 \*   
## df$x2 0.4313 0.6668793   
## x22 -1.3211 0.1884898   
## df$x3 -1.1772 0.2410005   
## x32 -3.7863 0.0002212 \*\*\*  
## df$x16 0.0794 0.9367939   
## Tukey test 0.7469 0.4551329   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# did not help linearity  
# add x12, x22, x12x16, x22x16  
residualPlots(lm(df$y~df$x1+x12+df$x2+x22+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)+x12x16+x22x16))



## Test stat Pr(>|Test stat|)   
## df$x1 0.0990 0.9212589   
## x12 2.6741 0.0083417 \*\*   
## df$x2 2.7582 0.0065513 \*\*   
## x22 -0.6961 0.4874506   
## df$x3 -0.9534 0.3419470   
## df$x16 -0.2180 0.8277272   
## x12x16 2.9932 0.0032396 \*\*   
## x22x16 -3.5940 0.0004436 \*\*\*  
## Tukey test -1.3288 0.1839047   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# did not help   
# try with x1 transformation  
x1t = exp(-df$x1)  
x12t = x1t^2  
x12x16t = x12t\*df$x16  
residualPlots(lm(df$y~x1t+x12t+df$x2+df$x3+df$x16+(df$x1\*df$x16)+(df$x2\*df$x16)+(df$x3\*df$x16)+x12x16t))



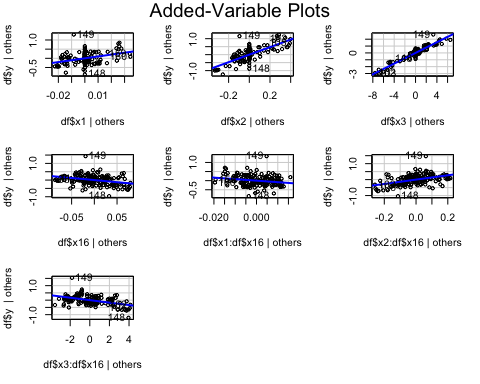
## Test stat Pr(>|Test stat|)   
## x1t 1.0031 0.3174588   
## x12t -0.3296 0.7421511   
## df$x2 -5.2236 5.857e-07 \*\*\*  
## df$x3 -3.6512 0.0003614 \*\*\*  
## df$x16 0.6916 0.4902887   
## df$x1 -0.3823 0.7028130   
## x12x16t -4.2292 4.090e-05 \*\*\*  
## Tukey test -4.2557 2.084e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# nope  
  
# just gonna have to accept that we could not get this ilnear

# the transformations did not help us in any other way either, so we are going to stick with that initial full model  
  
# since we can't deal with the linearity, the next biggest issue was the multicollinearity  
# it wouldn't make sense to drop any of the predictors we already have.   
# we can't drop any of the interaction terms, since that's what we're testing  
# and it's generally good practice to keep the first order terms when the interaction terms are in the model  
# and the best subset model selection backs this up.  
dfForBs = data.frame(y=df$y, x1=df$x1, x2=df$x2, x3=df$x3, x16=df$x16, x1x16=df$x1\*df$x16, x2x16=df$x2\*df$x16, x3x16=df$x3\*df$x16)  
newBS = BestSub(dfForBs[2:8], dfForBs$y, num=1)  
newBS

## p 1 2 3 4 5 6 7 SSEp r2 r2.adj Cp AICp  
## 1 2 0 0 1 0 0 0 0 167.69841 0.9555554 0.9552741 1821.72054 11.51894  
## 2 3 0 1 1 0 0 0 0 29.41902 0.9922032 0.9921038 192.94782 -264.96520  
## 3 4 1 1 1 0 0 0 0 26.91483 0.9928668 0.9927297 165.41511 -277.19943  
## 4 5 0 1 1 1 0 1 0 21.11915 0.9944028 0.9942584 99.06485 -313.99897  
## 5 6 0 1 1 1 0 1 1 14.62343 0.9961244 0.9959986 24.45879 -370.80777  
## 6 7 1 1 1 1 0 1 1 13.51630 0.9964178 0.9962773 13.40202 -381.40439  
## 7 8 1 1 1 1 1 1 1 12.88865 0.9965842 0.9964268 8.00000 -387.01222  
## SBCp PRESSp  
## 1 17.66928 171.85262  
## 2 -255.73968 30.71954  
## 3 -264.89874 28.21776  
## 4 -298.62310 22.96973  
## 5 -352.35673 16.04793  
## 6 -359.87817 14.97826  
## 7 -362.41083 14.50095

# all of the metrics tell us that the model with all the terms is the best to use  
# so we should not drop any for the sake of solving multicollinearity  
# av plots also tell us that we should not drop x1, x2 or x3  
avPlots(m1)



# the only ones that should be considered dropping are the interaction terms, which is what we're testing  
# instead, we will deal with multicollinearity with ridge regression

# get vifs for each ridge model, take the model that did the best job reducing vif  
# code to download lmridge  
#download.file("https://cran.r-project.org/src/contrib/lmridge\_1.2.2.tar.gz", destfile = "lmridge\_1.2.2.tar.gz")  
#install.packages("lmridge\_1.2.2.tar.gz", repos = NULL, type = "source")  
library(lmridge)

##   
## Attaching package: 'lmridge'

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

m3 = lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=df, K= seq(0, 1, .02))  
#plot(m3)  
vif(m3)

## x1 x2 x3 x16 x1:x16 x2:x16 x3:x16  
## k=0 8.52058 27.52978 13.83720 167.26651 17.50704 186.08134 15.62735  
## k=0.02 3.44395 5.39894 3.96159 4.38289 4.27283 3.90694 3.88807  
## k=0.04 2.63419 3.54297 2.71878 2.09199 2.93751 1.75829 2.73640  
## k=0.06 2.13376 2.56691 2.06737 1.36026 2.23100 1.13363 2.11412  
## k=0.08 1.79041 1.97063 1.66898 0.99727 1.79328 0.83317 1.72026  
## k=0.1 1.53956 1.57528 1.40142 0.78202 1.49638 0.65671 1.44798  
## k=0.12 1.34793 1.29781 1.20949 0.64085 1.28202 0.54135 1.24824  
## k=0.14 1.19659 1.09455 1.06492 0.54185 1.11999 0.46060 1.09528  
## k=0.16 1.07391 0.94057 0.95186 0.46900 0.99315 0.40128 0.97431  
## k=0.18 0.97242 0.82070 0.86079 0.41338 0.89111 0.35607 0.87622  
## k=0.2 0.88704 0.72527 0.78570 0.36967 0.80719 0.32062 0.79507  
## k=0.22 0.81423 0.64785 0.72259 0.33449 0.73693 0.29215 0.72684  
## k=0.24 0.75141 0.58401 0.66871 0.30562 0.67722 0.26884 0.66869  
## k=0.26 0.69668 0.53064 0.62210 0.28154 0.62585 0.24942 0.61857  
## k=0.28 0.64859 0.48547 0.58135 0.26117 0.58119 0.23300 0.57494  
## k=0.3 0.60602 0.44685 0.54537 0.24373 0.54200 0.21896 0.53664  
## k=0.32 0.56810 0.41350 0.51336 0.22863 0.50733 0.20681 0.50277  
## k=0.34 0.53413 0.38446 0.48468 0.21544 0.47647 0.19619 0.47262  
## k=0.36 0.50353 0.35899 0.45881 0.20382 0.44881 0.18682 0.44563  
## k=0.38 0.47584 0.33649 0.43537 0.19351 0.42389 0.17850 0.42133  
## k=0.4 0.45068 0.31650 0.41403 0.18430 0.40134 0.17105 0.39936  
## k=0.42 0.42773 0.29863 0.39450 0.17603 0.38083 0.16434 0.37941  
## k=0.44 0.40673 0.28257 0.37657 0.16854 0.36211 0.15825 0.36121  
## k=0.46 0.38745 0.26808 0.36006 0.16175 0.34495 0.15271 0.34455  
## k=0.48 0.36968 0.25495 0.34479 0.15555 0.32919 0.14763 0.32926  
## k=0.5 0.35328 0.24299 0.33064 0.14987 0.31465 0.14296 0.31517  
## k=0.52 0.33809 0.23207 0.31749 0.14464 0.30120 0.13865 0.30216  
## k=0.54 0.32399 0.22205 0.30523 0.13982 0.28874 0.13464 0.29010  
## k=0.56 0.31088 0.21284 0.29379 0.13535 0.27715 0.13092 0.27891  
## k=0.58 0.29865 0.20434 0.28308 0.13120 0.26636 0.12744 0.26848  
## k=0.6 0.28723 0.19648 0.27304 0.12733 0.25629 0.12419 0.25876  
## k=0.62 0.27655 0.18919 0.26361 0.12371 0.24686 0.12113 0.24967  
## k=0.64 0.26653 0.18241 0.25474 0.12032 0.23803 0.11825 0.24115  
## k=0.66 0.25712 0.17609 0.24637 0.11713 0.22974 0.11553 0.23316  
## k=0.68 0.24827 0.17018 0.23848 0.11414 0.22195 0.11296 0.22564  
## k=0.7 0.23993 0.16465 0.23101 0.11131 0.21461 0.11052 0.21856  
## k=0.72 0.23207 0.15946 0.22395 0.10864 0.20768 0.10820 0.21188  
## k=0.74 0.22464 0.15459 0.21725 0.10611 0.20114 0.10599 0.20557  
## k=0.76 0.21762 0.15000 0.21090 0.10371 0.19495 0.10389 0.19961  
## k=0.78 0.21096 0.14567 0.20486 0.10143 0.18909 0.10188 0.19395  
## k=0.8 0.20465 0.14158 0.19911 0.09926 0.18354 0.09995 0.18858  
## k=0.82 0.19866 0.13771 0.19364 0.09719 0.17826 0.09811 0.18349  
## k=0.84 0.19297 0.13405 0.18843 0.09522 0.17325 0.09634 0.17864  
## k=0.86 0.18756 0.13057 0.18345 0.09333 0.16847 0.09464 0.17402  
## k=0.88 0.18240 0.12727 0.17870 0.09152 0.16393 0.09301 0.16962  
## k=0.9 0.17748 0.12412 0.17416 0.08979 0.15960 0.09144 0.16542  
## k=0.92 0.17279 0.12113 0.16982 0.08813 0.15546 0.08992 0.16141  
## k=0.94 0.16832 0.11827 0.16566 0.08653 0.15151 0.08846 0.15758  
## k=0.96 0.16403 0.11555 0.16167 0.08500 0.14774 0.08705 0.15391  
## k=0.98 0.15994 0.11295 0.15785 0.08352 0.14412 0.08569 0.15039  
## k=1 0.15602 0.11046 0.15418 0.08210 0.14066 0.08436 0.14702

# k = .08 gives all the vifs pretty close to 1, so we're gonna go with that model  
# model summary for k = .08  
summary(lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=df, K=.08))

##   
## Call:  
## lmridge.default(formula = y ~ x1 + x2 + x3 + x16 + (x1 \* x16) +   
## (x2 \* x16) + (x3 \* x16), data = df, K = 0.08)  
##   
##   
## Coefficients: for Ridge parameter K= 0.08   
## Estimate Estimate (Sc) StdErr (Sc) t-value (Sc) Pr(>|t|)   
## Intercept -1.9512 -438.1490 7.3476 -59.6317 <2e-16 \*\*\*  
## x1 2.3371 0.8576 0.6203 1.3826 0.1688   
## x2 2.8551 24.3946 0.6508 37.4864 <2e-16 \*\*\*  
## x3 0.3053 33.4021 0.5989 55.7742 <2e-16 \*\*\*  
## x16 -0.3099 -1.9574 0.4629 -4.2282 <2e-16 \*\*\*  
## x1:x16 -8.4068 -3.3316 0.6208 -5.3667 <2e-16 \*\*\*  
## x2:x16 0.1173 2.1027 0.4231 4.9692 <2e-16 \*\*\*  
## x3:x16 0.0307 2.6325 0.6080 4.3296 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Ridge Summary  
## R2 adj-R2 DF ridge F AIC BIC   
## 0.91750 0.91430 4.47754 2499.73075 -242.49029 583.30674   
## Ridge minimum MSE= 900.6866 at K= 0.08   
## P-value for F-test ( 4.47754 , 154.6605 ) = 1.065625e-142   
## -------------------------------------------------------------------

vif(lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=df, K=.08))

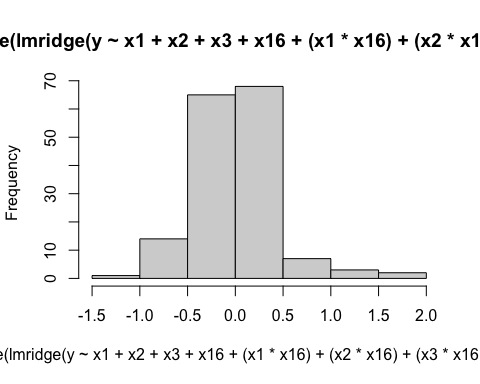
## x1 x2 x3 x16 x1:x16 x2:x16 x3:x16  
## k=0.08 1.79041 1.97063 1.66898 0.99727 1.79328 0.83317 1.72026

# can we verify that the residuals of the ridge model are normal?  
shapiro.test(residuals.lmridge(lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=df, K=.08)))

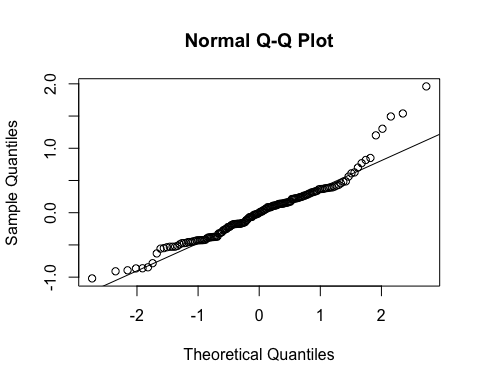
##   
## Shapiro-Wilk normality test  
##   
## data: residuals.lmridge(lmridge(y ~ x1 + x2 + x3 + x16 + (x1 \* x16) + (x2 \* x16) + (x3 \* x16), data = df, K = 0.08))  
## W = 0.9428, p-value = 4.548e-06

# shapiro test says definitely not normal

# what about hist and qqplot  
hist(residuals.lmridge(lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=df, K=.08)))



qqnorm(residuals.lmridge(lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=df, K=.08)))  
qqline(residuals.lmridge(lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=df, K=.08)))



# qqplot shows that it's definitely not normally distributed

# solve with bootstrapping  
  
# ridge Regression Bootstrap  
boot.ridge <- function(data, indices, maxit=100) {  
 data <- data[indices,]  
 colnames(data)<-c("y","x1",'x2','x3','x16')  
 mod = lmridge(y~x1+x2+x3+x16+(x1\*x16)+(x2\*x16)+(x3\*x16), data=data, K=.08)  
 coefficients(mod)  
}  
df.ridge = boot(data=df, statistic=boot.ridge, R=200, maxit=100)  
df.ridge

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = df, statistic = boot.ridge, R = 200, maxit = 100)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* -1.95119 0.05219110 0.393643187  
## t2\* 2.33708 -0.40086065 2.895205725  
## t3\* 2.85510 -0.00603355 0.130461445  
## t4\* 0.30534 -0.00170665 0.007815835  
## t5\* -0.30988 -0.01046780 0.078316518  
## t6\* -8.40685 0.14231710 1.590506681  
## t7\* 0.11728 -0.00327905 0.022443766  
## t8\* 0.03069 0.00104655 0.007592801

summary(df.ridge)

##   
## Number of bootstrap replications R = 200   
## original bootBias bootSE bootMed  
## 1 -1.95119 0.0521911 0.3936432 -1.893890  
## 2 2.33708 -0.4008606 2.8952057 2.221690  
## 3 2.85510 -0.0060336 0.1304614 2.841280  
## 4 0.30534 -0.0017066 0.0078158 0.304680  
## 5 -0.30988 -0.0104678 0.0783165 -0.309320  
## 6 -8.40685 0.1423171 1.5905067 -8.361990  
## 7 0.11728 -0.0032790 0.0224438 0.114330  
## 8 0.03069 0.0010466 0.0075928 0.031755

# get cis for B116, B216, B316  
boot.ci(df.ridge, index=6, type="perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 200 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = df.ridge, type = "perc", index = 6)  
##   
## Intervals :   
## Level Percentile   
## 95% (-11.097, -4.573 )   
## Calculations and Intervals on Original Scale  
## Some percentile intervals may be unstable

boot.ci(df.ridge, index=7, type="perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 200 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = df.ridge, type = "perc", index = 7)  
##   
## Intervals :   
## Level Percentile   
## 95% ( 0.0589, 0.1550 )   
## Calculations and Intervals on Original Scale  
## Some percentile intervals may be unstable

boot.ci(df.ridge, index=8, type="perc")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 200 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = df.ridge, type = "perc", index = 8)  
##   
## Intervals :   
## Level Percentile   
## 95% ( 0.0160, 0.0485 )   
## Calculations and Intervals on Original Scale  
## Some percentile intervals may be unstable

# maybe cross validate these with a bonferroni CI for each of the interaction betas  
  
  
# all three interaction terms were shown to have impact on DPI for republican presidents  
# but since we don't have linearity, we have to take things with a grain of salt  
# and since the interaction terms for x2 and x3 are so close to zero, their impact on DPI might not be substantial enough to phone home about  
# the impact of interest rates on DPI falls more in line with what we'd expect for republicans in office compared to democrats in office.   
# higher interests rate generally indicate a worse economy, which would lead to less disposable income.