Midterm

Anthony Wilson

2023-03-05

library(MASS)  
library(dplyr)  
library(caret)  
library(glmnet)  
library(corrplot)  
library(RColorBrewer)  
library(car)  
library(explore)  
library(ggplot2)

### Question 3

################################################  
## Load data  
################################################  
  
myAuto = read.csv("04cars.csv")  
  
# See the data summary  
summary(myAuto)

## Type Retailprice Sport SUV   
## Length:428 Min. : 10280 Min. :0.0000 Min. :0.0000   
## Class :character 1st Qu.: 20334 1st Qu.:0.0000 1st Qu.:0.0000   
## Mode :character Median : 27635 Median :0.0000 Median :0.0000   
## Mean : 32775 Mean :0.1145 Mean :0.1402   
## 3rd Qu.: 39205 3rd Qu.:0.0000 3rd Qu.:0.0000   
## Max. :192465 Max. :1.0000 Max. :1.0000   
##   
## Wagon Minivan Pickup Engine   
## Min. :0.00000 Min. :0.00000 Min. :0.00000 Min. :1.300   
## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:2.375   
## Median :0.00000 Median :0.00000 Median :0.00000 Median :3.000   
## Mean :0.07009 Mean :0.04673 Mean :0.05607 Mean :3.197   
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:3.900   
## Max. :1.00000 Max. :1.00000 Max. :1.00000 Max. :8.300   
##   
## Cylinders Horsepower CityMPG HwyMPG   
## Min. : 3.000 Min. : 73.0 Min. :10.00 Min. :12.00   
## 1st Qu.: 4.000 1st Qu.:165.0 1st Qu.:17.00 1st Qu.:24.00   
## Median : 6.000 Median :210.0 Median :19.00 Median :26.00   
## Mean : 5.808 Mean :215.9 Mean :20.09 Mean :26.91   
## 3rd Qu.: 6.000 3rd Qu.:255.0 3rd Qu.:21.00 3rd Qu.:29.00   
## Max. :12.000 Max. :500.0 Max. :60.00 Max. :66.00   
## NA's :2 NA's :14 NA's :14   
## Weight Wheelbase Length Height   
## Min. :1850 Min. : 89.0 Min. :143.0 Length:428   
## 1st Qu.:3102 1st Qu.:103.0 1st Qu.:177.0 Class :character   
## Median :3474 Median :107.0 Median :186.0 Mode :character   
## Mean :3577 Mean :108.2 Mean :185.1   
## 3rd Qu.:3974 3rd Qu.:112.0 3rd Qu.:193.0   
## Max. :7190 Max. :144.0 Max. :227.0   
## NA's :2 NA's :2 NA's :26

head(myAuto)

## Type Retailprice Sport SUV Wagon Minivan Pickup Engine Cylinders Horsepower  
## 1 Other 10280 0 0 0 0 0 1.6 4 104  
## 2 Other 10539 0 0 0 0 0 1.6 4 103  
## 3 Other 10760 0 0 0 0 0 1.5 4 108  
## 4 Other 10995 0 0 0 0 0 2.2 4 140  
## 5 Other 11155 0 0 0 0 0 1.6 4 104  
## 6 Other 11290 0 0 0 0 0 1.5 4 108  
## CityMPG HwyMPG Weight Wheelbase Length Height  
## 1 26 33 2403 95 167 66  
## 2 29 33 2255 96 167 66  
## 3 35 43 2035 93 163 65  
## 4 26 35 2692 103 185 67  
## 5 25 32 2458 95 167 66  
## 6 35 43 2055 93 163 65

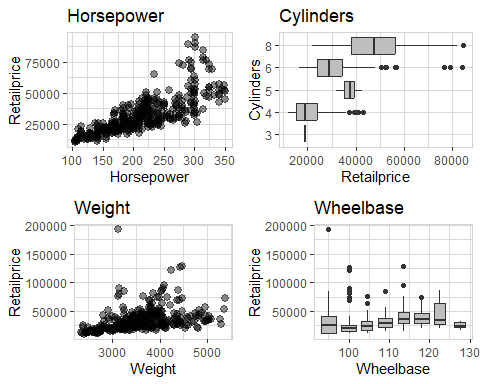
################################################  
## Clean up and EDA  
################################################  
  
# Length and Width of pickup trucks were not given, so remove both.  
# If just do na.omit on length/weight, it will make the pickup no variance,   
# basically removing pickup.  
# If pickup has significantly different characteristics or features compared to   
# the type of cars used in the training data,   
# the model may not perform well in predicting the price of pickup.  
myAuto <- myAuto %>% select(-c(Height,Length))  
  
# I include these three electric cars  
# remove ??? electric cars due abnormal values on HP and mpg  
#electric\_cars <- c(83,100,112)  
#myAuto <- myAuto %>% slice(-electric\_cars)  
  
# Remove the rest of NA  
myAuto <- na.omit(myAuto)  
  
# Check to make sure no more NA  
colSums(is.na(myAuto))

## Type Retailprice Sport SUV Wagon Minivan   
## 0 0 0 0 0 0   
## Pickup Engine Cylinders Horsepower CityMPG HwyMPG   
## 0 0 0 0 0 0   
## Weight Wheelbase   
## 0 0

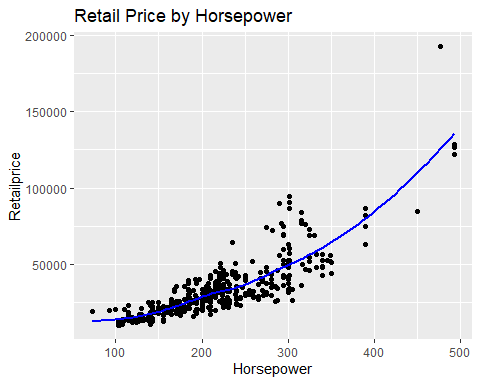
# Check for no zero variances, so good to go  
myAuto.numeric <- select\_if(myAuto, is.numeric)  
apply(myAuto.numeric, 2, var) == 0

## Retailprice Sport SUV Wagon Minivan Pickup   
## FALSE FALSE FALSE FALSE FALSE FALSE   
## Engine Cylinders Horsepower CityMPG HwyMPG Weight   
## FALSE FALSE FALSE FALSE FALSE FALSE   
## Wheelbase   
## FALSE

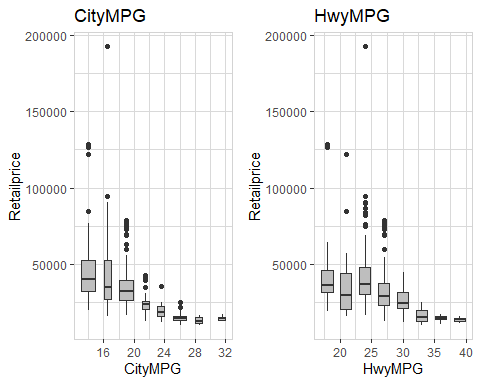
# For violin graph of RetailPrice by Model  
myAuto <- myAuto %>%  
 mutate(Model = case\_when(  
 Sport == 1 ~ 'Sport',  
 SUV == 1 ~ 'SUV',  
 Wagon == 1 ~'Wagon',  
 Minivan == 1 ~'Minivan',  
 Pickup == 1 ~ 'Pickup',  
 TRUE ~ 'Other')) %>%  
 mutate(Model = factor(Model))  
  
# Check relation between Retailprice with predictors  
# Price and HP = positive strong  
# Price and Cylinders = about positive  
myAuto %>% select(Retailprice, Horsepower, Cylinders, Weight, Wheelbase) %>%   
 explore\_all(target = Retailprice, targetpct=TRUE, split=TRUE)



# More detail Retailprice with Horsepower  
ggplot(myAuto, aes(x=Horsepower, y=Retailprice)) +  
 geom\_point() +  
 geom\_smooth(aes(x=Horsepower, y=Retailprice), se=FALSE, color="blue") +  
 ggtitle("Retail Price by Horsepower")

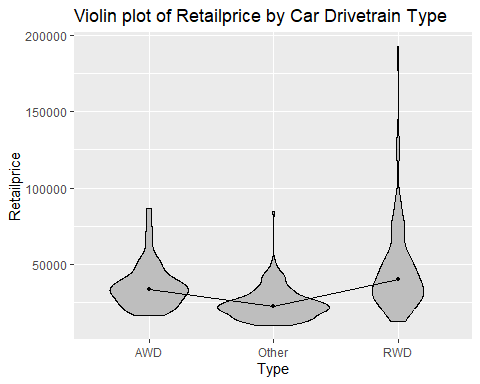


# Price and MPG = negative, interesting??  
myAuto %>% select(Retailprice, CityMPG, HwyMPG) %>%   
 explore\_all(target = Retailprice)

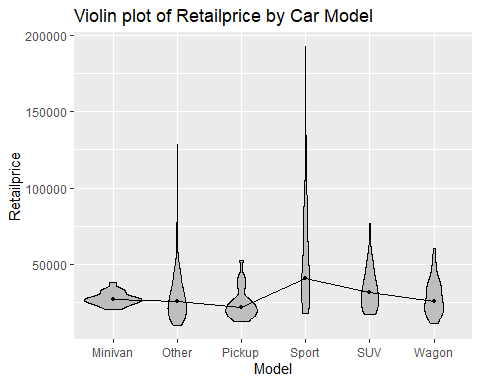


# based on median, the higher prices to lower prices:  
# sport, suv, minivan, wagon, other, pickup  
# rwd has higher prices than awd, other is the lowest  
ggplot(myAuto, aes(x = Type, y = Retailprice)) +  
 geom\_violin(fill = "gray", color = "black") +  
 stat\_summary(fun.y = median, geom = "line", aes(group = 1)) +  
 stat\_summary(fun = median, geom = "point", shape = 20, size = 2) +  
 labs(title = "Violin plot of Retailprice by Car Drivetrain Type")

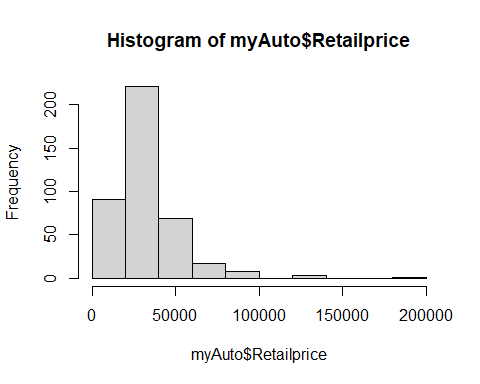
## Warning: The `fun.y` argument of `stat\_summary()` is deprecated as of ggplot2 3.3.0.  
## ℹ Please use the `fun` argument instead.



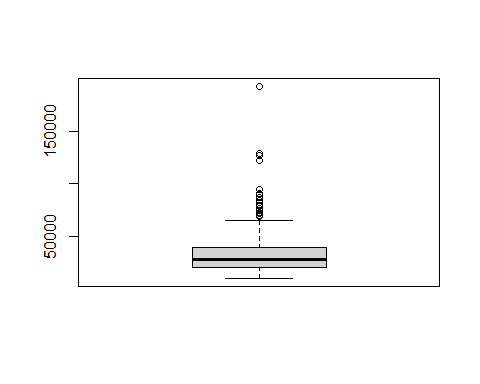
ggplot(myAuto, aes(x = Model, y = Retailprice)) +  
 geom\_violin(fill = "gray", color = "black") +  
 stat\_summary(fun.y = median, geom = "line", aes(group = 1)) +  
 stat\_summary(fun = median, geom = "point", shape = 20, size = 2) +  
 labs(title = "Violin plot of Retailprice by Car Model")



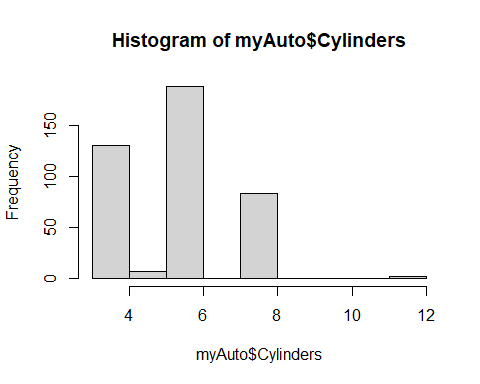
# remove the Model  
myAuto <- myAuto %>% select(-c(Model))  
  
# Check normality  
# right skewed, do norm later  
hist(myAuto$Retailprice)



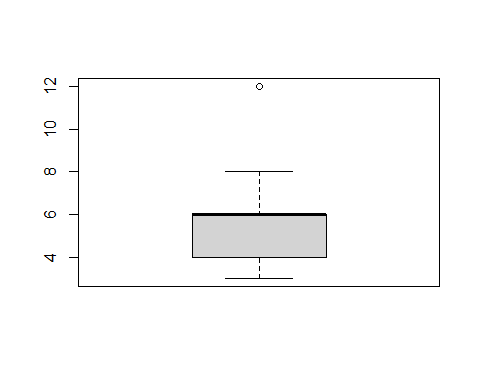
boxplot(myAuto$Retailprice)



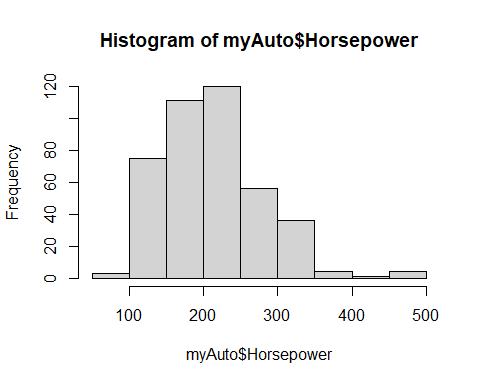
# right skewed, do norm later  
hist(myAuto$Cylinders)



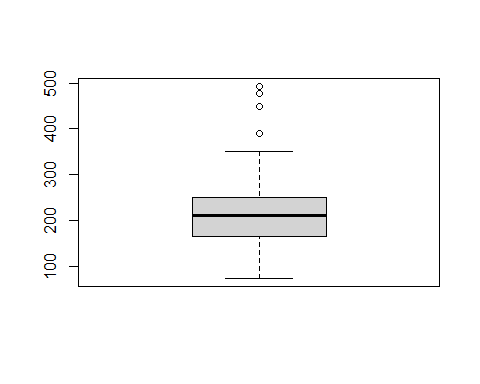
boxplot(myAuto$Cylinders)



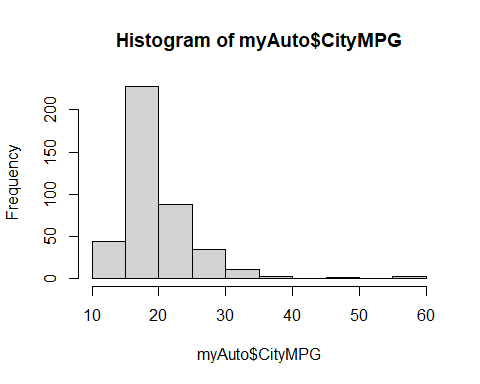
# right skewed, do norm later  
hist(myAuto$Horsepower)



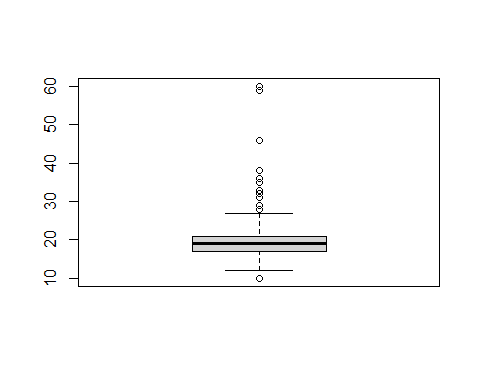
boxplot(myAuto$Horsepower)



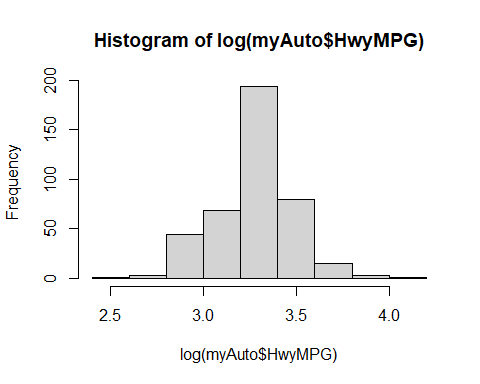
# right skewed, do norm later  
hist(myAuto$CityMPG)



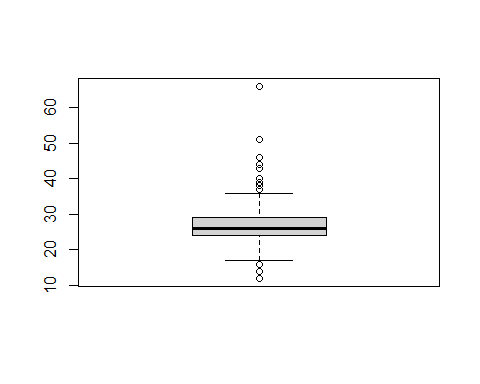
boxplot(myAuto$CityMPG)



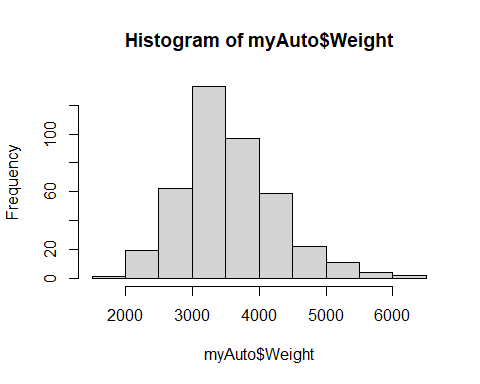
# right skewed, do norm later  
hist(log(myAuto$HwyMPG))



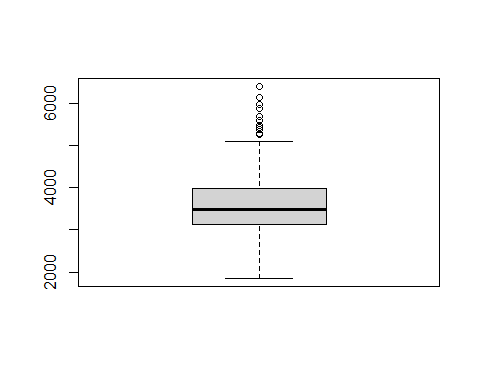
boxplot(myAuto$HwyMPG)



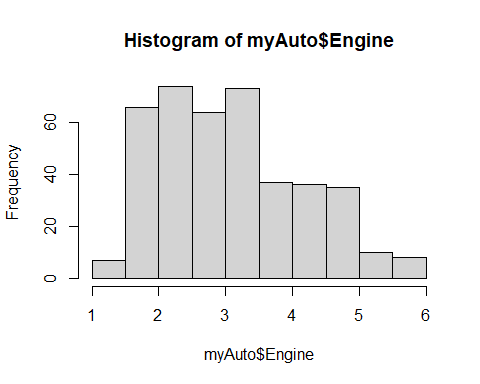
# right skewed, do norm later  
hist(myAuto$Weight)



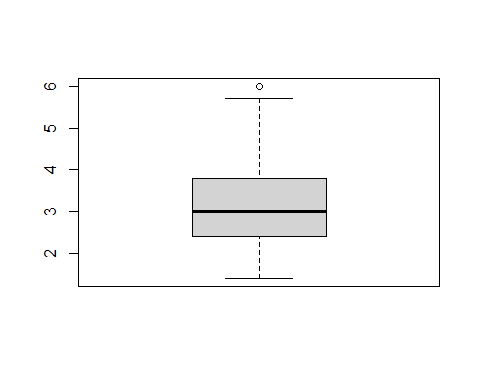
boxplot(myAuto$Weight)



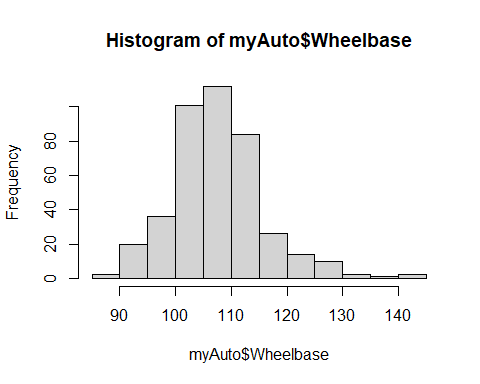
# about normal, no transform  
hist(myAuto$Engine)



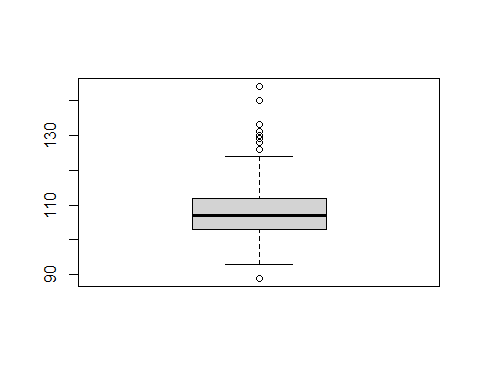
boxplot(myAuto$Engine)



# about normal, no transform  
hist(myAuto$Wheelbase)



boxplot(myAuto$Wheelbase)



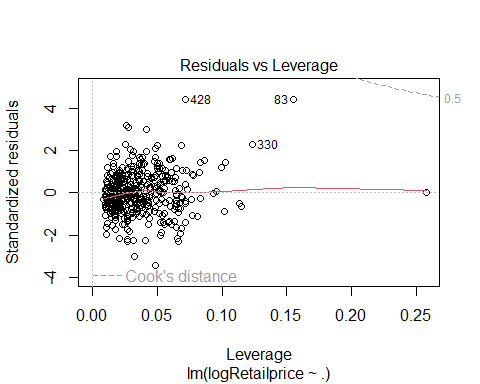
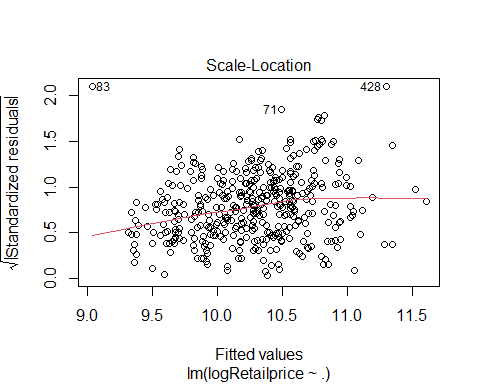
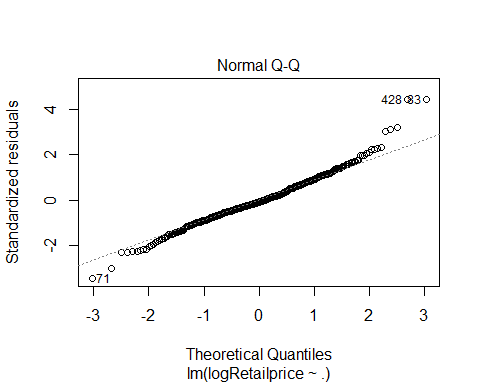
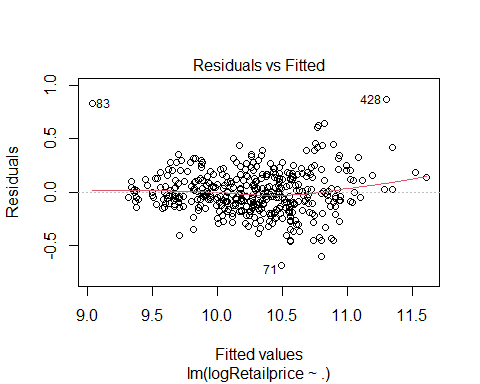
### Question 4

### (There are two segments: evaluate full model and removing couple predictors)

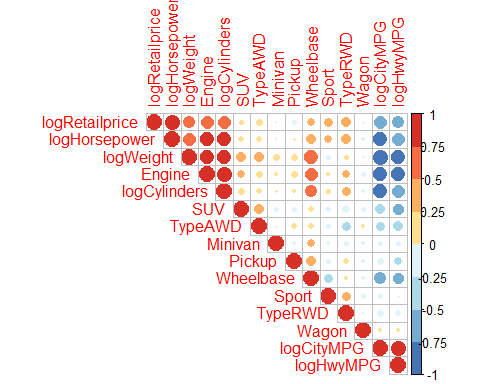
################################################  
## Evaluate full model  
################################################  
  
# Transformations:  
# one hot encode on type  
# log: retailprice, cylinders, weight, hwympg, citympg, horsepower  
myAuto.full <- myAuto %>%  
 mutate(logRetailprice = log(Retailprice),  
 logCylinders = log(Cylinders),  
 logWeight = log(Weight),  
 logCityMPG = log(CityMPG),  
 logHwyMPG = log(HwyMPG),  
 TypeAWD = if\_else(Type == "AWD", 1, 0),  
 TypeRWD = if\_else(Type == "RWD", 1, 0),  
 logHorsepower = log(Horsepower)) %>%   
 select(-c(Type,Weight,CityMPG,HwyMPG,Retailprice,Cylinders,Horsepower))  
  
# Prof Brisbin was ok'ed to use the VIF with full dataset  
# but need to explain about the data leakage  
  
# Check on VIF on full data. It can cause a data leakage as  
# it can lead to overly optimistic performance metrics, as the model has   
# already seen the test data during training, which is not   
# a true representation of its performance on unseen data.  
# logCityMPG, logHwyMPG and logWeight are greater than 10  
myAuto.full.vif = lm(logRetailprice ~ ., data = myAuto.full)  
vif(myAuto.full.vif)

## Sport SUV Wagon Minivan Pickup   
## 2.035818 2.783195 1.121433 1.434838 2.245195   
## Engine Wheelbase logCylinders logWeight logCityMPG   
## 8.235131 4.820978 7.008460 10.590989 17.076833   
## logHwyMPG TypeAWD TypeRWD logHorsepower   
## 16.990269 1.726296 1.769275 5.659965

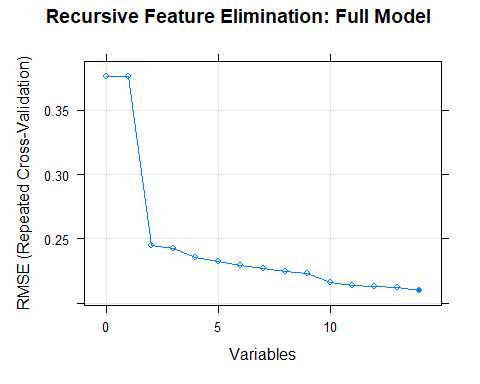
# Check on residuals, QQ, Scale, and Leverage plots  
# Looked OK, some outliers  
plot(myAuto.full.vif)



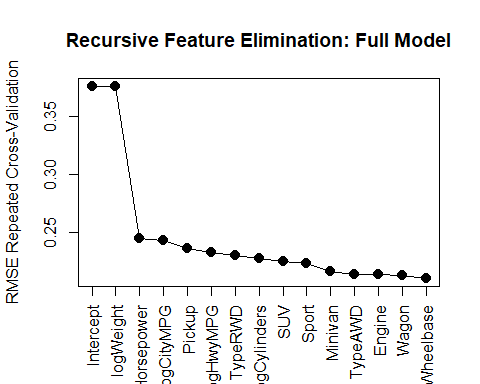
# Check on correlations on full model  
# Retailprice has positive relationships with predictors:   
# Horsepower, Weight, Engine, Cylinders,   
# Retailprice has negative relationships with predictors: CityMPG, HwyMPG  
# There are a bunch of collinearities between predictors  
myAuto.correlations.full <- cor(select\_if(myAuto.full, is.numeric),   
 use = "pairwise.complete.obs")  
corrplot(myAuto.correlations.full, type = "upper", order = "hclust",   
 col = rev(brewer.pal(n = 8, name = "RdYlBu")))



# Check on Recursive Feature Elimination  
# The more predictors the lower RMSE  
set.seed(8)  
control\_rfe <- rfeControl(functions = lmFuncs, method = "repeatedcv",   
 repeats = 5, number = 5, verbose = FALSE)  
Model.rfe = (logRetailprice ~ .)  
X\_train = model.matrix(Model.rfe, data = myAuto.full)[,-1]  
y\_train = myAuto.full$logRetailprice  
result\_rfe = rfe(x = X\_train, y = y\_train,   
 sizes = c(1:dim(myAuto.full)[2]-1),  
 rfeControl = control\_rfe)  
plot(result\_rfe, type = c("g", "o"), main = "Recursive Feature Elimination: Full Model")



plot(result\_rfe$results$RMSE[0:length(result\_rfe$optVariables)+1] ~ c(0:length(result\_rfe$optVariables)),  
 xaxt="n", type = "l", xlab = "", ylab = "RMSE Repeated Cross-Validation", main = "Recursive Feature Elimination: Full Model")  
points(result\_rfe$results$RMSE[0:length(result\_rfe$optVariables)+1] ~ c(0:length(result\_rfe$optVariables)), pch = 20, cex = 2)  
axis(side = 1, at = 0:length(result\_rfe$optVariables), labels = c('Intercept',result\_rfe$optVariables), las = 2)



# FULL MODEL WITH ENET, checking whether we can get a simpler model  
# None of predictors is removed  
alphalist = seq(0, 1, by = 0.1)  
lambdalist = c((1:1000)/10000)  
training = trainControl(method = "cv", number = 5)  
myAuto.full.ENET = train(logRetailprice ~ ., data = myAuto.full,  
 method = "glmnet", trControl = training,  
 tuneGrid = expand.grid(alpha=alphalist,lambda=lambdalist))  
coef(myAuto.full.ENET$finalModel, s = myAuto.full.ENET$bestTune$lambda)

## 15 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) -4.538668724  
## Sport 0.142287048  
## SUV -0.148375088  
## Wagon -0.019956880  
## Minivan -0.109049680  
## Pickup -0.329097361  
## Engine -0.028797714  
## Wheelbase -0.009069972  
## logCylinders 0.147657920  
## logWeight 1.154815844  
## logCityMPG 0.236300108  
## logHwyMPG 0.173153765  
## TypeAWD 0.123910441  
## TypeRWD 0.178378960  
## logHorsepower 0.919727841

myAuto.full.ENET.RMSE <- min(myAuto.full.ENET$results$RMSE)  
paste("ENET RMSE: ",myAuto.full.ENET.RMSE)

## [1] "ENET RMSE: 0.211880790519141"

# FULL MODEL WITH ROBUST  
training = trainControl(method = "cv", number = 5)  
myAuto.full.ROBUST = train(logRetailprice ~ ., data = myAuto.full,  
 method = "rlm", trControl = training)  
coef(myAuto.full.ROBUST$finalModel)

## (Intercept) Sport SUV Wagon Minivan   
## -6.49158246 0.16205764 -0.18202492 -0.02655028 -0.13963263   
## Pickup Engine Wheelbase logCylinders logWeight   
## -0.34104207 -0.06708319 -0.01011313 0.19968222 1.43737805   
## logCityMPG logHwyMPG TypeAWD TypeRWD logHorsepower   
## 0.29514297 0.12001647 0.10750540 0.18269961 0.88116132

myAuto.full.ROBUST.RMSE <- min(myAuto.full.ROBUST$results$RMSE)  
paste("ROBUST RMSE: ",myAuto.full.ROBUST.RMSE)

## [1] "ROBUST RMSE: 0.211086881733444"

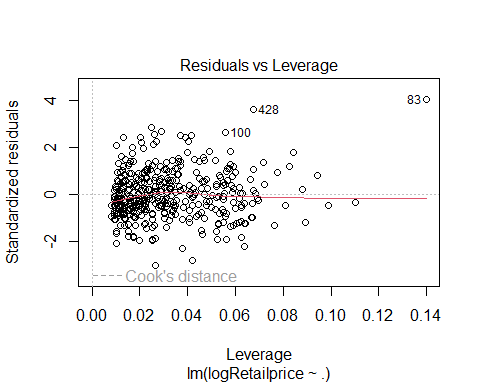
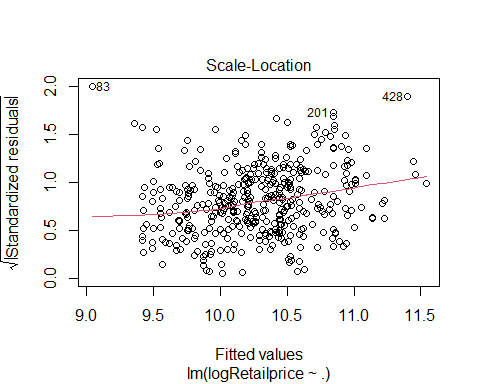
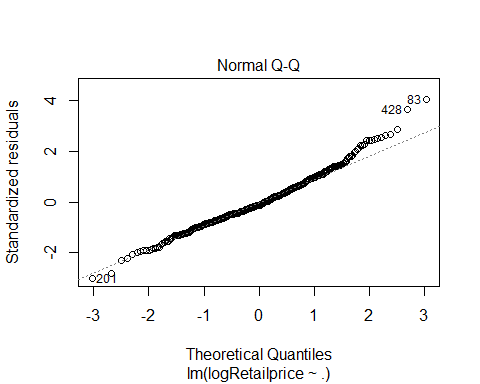
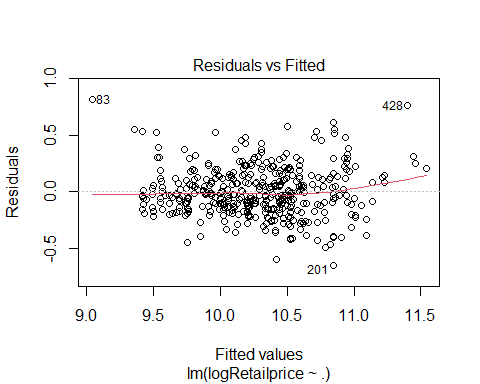
### Question 4

### (There are two segments: evaluate full model and removing couple predictors)

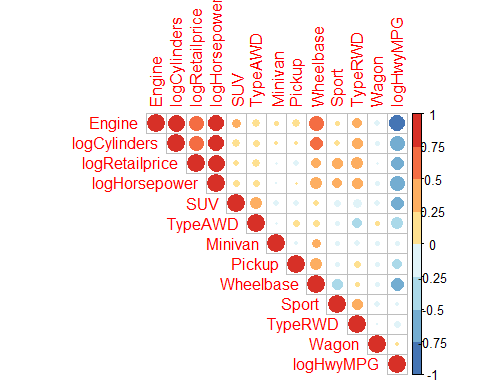
##################################################  
## Re-evaluate model by removing couple predictors  
##################################################  
  
# Transformations:  
# one hot encode on type  
# log: retailprice, cylinders, weight, hwympg, citympg, horsepower  
# Since logCityMPG, logHwyMPG and logWeight are greater than 10  
# remove logCityMPG and logweight  
myAuto.reduced <- myAuto %>%  
 mutate(logRetailprice = log(Retailprice),  
 logCylinders = log(Cylinders),  
 logHwyMPG = log(HwyMPG),  
 TypeAWD = if\_else(Type == "AWD", 1, 0),  
 TypeRWD = if\_else(Type == "RWD", 1, 0),  
 logHorsepower = log(Horsepower)) %>%   
 select(-c(Type,Weight,CityMPG,HwyMPG,Retailprice,Cylinders,Horsepower))  
  
# Prof Brisbin was ok'ed to use the VIF with full dataset  
# but need to explain about the data leakage  
  
# Check on VIF on full data. It can cause a data leakage.  
# it can lead to overly optimistic performance metrics, as the model has   
# already seen the test data during training, which is not   
# a true representation of its performance on unseen data.  
# everything looks great, under 10  
myAuto.reduced.vif = lm(logRetailprice ~ ., data = myAuto.reduced)  
vif(myAuto.reduced.vif)

## Sport SUV Wagon Minivan Pickup   
## 2.004855 2.347245 1.095909 1.350488 2.120592   
## Engine Wheelbase logCylinders logHwyMPG TypeAWD   
## 8.108123 3.653666 6.905691 5.295903 1.648436   
## TypeRWD logHorsepower   
## 1.752775 4.796450

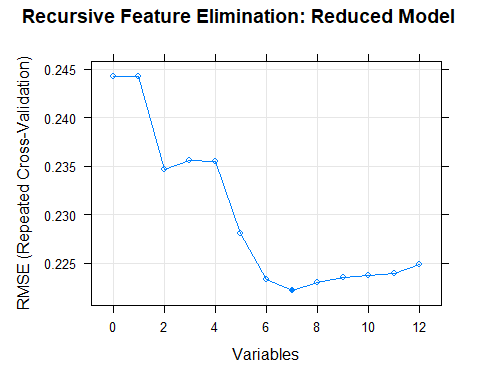
# Check on residuals, QQ, Scale, and Leverage plots  
# Looked OK, some outliers  
plot(myAuto.reduced.vif)



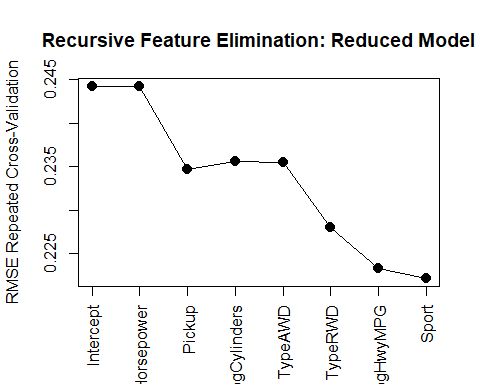
# Check on correlations on the reduced model  
myAuto.correlations.reduced <- cor(select\_if(myAuto.reduced, is.numeric),   
 use = "pairwise.complete.obs")  
corrplot(myAuto.correlations.reduced, type = "upper", order = "hclust",   
 col = rev(brewer.pal(n = 8, name = "RdYlBu")))



# Recursive Feature Elimination  
set.seed(8)  
control\_rfe <- rfeControl(functions = lmFuncs, method = "repeatedcv",  
 repeats = 5, number = 5, verbose = FALSE)  
  
Model.rfe = (logRetailprice ~ .)  
X\_train = model.matrix(Model.rfe, data = myAuto.reduced)[,-1]  
y\_train = myAuto.reduced$logRetailprice  
  
result\_rfe = rfe(x = X\_train, y = y\_train,   
 sizes = c(1:dim(myAuto.reduced)[2]-1),  
 rfeControl = control\_rfe)  
plot(result\_rfe, type = c("g", "o"), main = "Recursive Feature Elimination: Reduced Model")



plot(result\_rfe$results$RMSE[0:length(result\_rfe$optVariables)+1] ~ c(0:length(result\_rfe$optVariables)),  
 xaxt="n", type = "l", xlab = "", ylab = "RMSE Repeated Cross-Validation", main = "Recursive Feature Elimination: Reduced Model")  
points(result\_rfe$results$RMSE[0:length(result\_rfe$optVariables)+1] ~ c(0:length(result\_rfe$optVariables)), pch = 20, cex = 2)  
axis(side = 1, at = 0:length(result\_rfe$optVariables), labels = c('Intercept',result\_rfe$optVariables), las = 2)



# REDUCED MODEL WITH ENET  
# Some predictors are automatically removed  
alphalist = seq(0, 1, by = 0.1)  
lambdalist = c((1:1000)/10000)  
training = trainControl(method = "cv", number = 5)  
myAuto.reduced.ENET = train(logRetailprice ~ ., data = myAuto.reduced,  
 method = "glmnet", trControl = training,  
 tuneGrid = expand.grid(alpha=alphalist,  
 lambda=lambdalist))  
coef(myAuto.reduced.ENET$finalModel, s = myAuto.reduced.ENET$bestTune$lambda)

## 13 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 4.2977636904  
## Sport 0.0875107374  
## SUV -0.0388585886  
## Wagon 0.0005654419  
## Minivan .   
## Pickup -0.3443451731  
## Engine .   
## Wheelbase .   
## logCylinders 0.1358661943  
## logHwyMPG .   
## TypeAWD 0.1373003699  
## TypeRWD 0.1516313072  
## logHorsepower 1.0688527537

myAuto.reduced.ENET.RMSE <- min(myAuto.reduced.ENET$results$RMSE)  
paste("ENET RMSE: ",myAuto.reduced.ENET.RMSE)

## [1] "ENET RMSE: 0.221421137069169"

# REDUCED MODEL WITH ROBUST  
Model.Robust = (logRetailprice ~ logHorsepower + TypeRWD + TypeAWD + SUV + Sport + Wagon + Minivan + Pickup)  
training = trainControl(method = "cv", number = 5)  
myAuto.reduced.ROBUST = train(Model.Robust, data = myAuto.reduced,  
 method = "rlm", trControl = training)  
coef(myAuto.reduced.ROBUST$finalModel)

## (Intercept) logHorsepower TypeRWD TypeAWD SUV   
## 3.92341680 1.18091277 0.17631851 0.16943621 -0.06421154   
## Sport Wagon Minivan Pickup   
## 0.08072167 0.02024058 -0.01032182 -0.38380771

myAuto.reduced.ROBUST.RMSE <- min(myAuto.reduced.ROBUST$results$RMSE)  
paste("ROBUST RMSE: ",myAuto.reduced.ROBUST.RMSE)

## [1] "ROBUST RMSE: 0.223435220377683"

### Question 5

##################################################  
## DOUBLE CROSS VALIDATION  
##################################################  
  
library(doParallel)  
  
myAuto <- myAuto.reduced  
  
cls = makeCluster(detectCores()-1)  
registerDoParallel(cls)  
  
set.seed(8)  
  
#tuning params for ENET  
lambdalist = c((1:1000)/10000)  
alphalist = seq(0, 1, by = 0.1)  
  
n = dim(myAuto)[1]  
n.alpha = length(alphalist)  
nfolds = 5  
  
groups = rep(1:nfolds,length=n)  
cvgroups = sample(groups,n)  
  
# store predicted values from the double-cross-validation  
allpredictedCV = rep(NA,n)  
allpredictedMethod = rep(NA,n)  
allpredictedMSE = rep(NA,n)  
  
# set up storage to see what models are "best" on the inner loops  
allbestRMSE = rep(NA,nfolds)  
allbestTypes = rep(NA,nfolds)  
allbestPars = vector("list",nfolds)  
allbestModel = vector("list",nfolds)  
  
Model.Full = (logRetailprice ~ .)  
Model.Penalized = (logRetailprice ~ .)  
Model.Robust = (logRetailprice ~ logHorsepower + TypeRWD + TypeAWD + SUV + Sport + Wagon + Minivan + Pickup)  
  
timing <- system.time({  
for (j in 1:nfolds) { # loop through outer splits  
 groupj = (cvgroups == j)  
   
 # train data  
 traindata = myAuto[!groupj,]  
 trainx = model.matrix(Model.Full, data = traindata)[,-1]  
 trainy = traindata$logRetailprice  
   
 # test data  
 validdata = myAuto[groupj,]  
 validx = model.matrix(Model.Full, data = validdata)[,-1]  
 validy = validdata$logRetailprice  
   
 # all model-fitting process with traindata  
 dataused=traindata  
   
 training = trainControl(method = "cv", number = 5, allowParallel = TRUE)  
  
 # cross-validation of penalized regression  
 fit\_caret\_penalized = train(Model.Penalized, data = dataused,  
 method = "glmnet", trControl = training,  
 tuneGrid = expand.grid(alpha=alphalist,lambda=lambdalist))  
   
 # cross-validation of robust regression   
 fit\_caret\_robust = train(Model.Robust, data = dataused, method = "rlm",  
 trControl = training)  
   
 # all best models  
 all\_best\_Types = c("ELASTICNET", "ROBUST")  
 all\_best\_Pars = list(fit\_caret\_penalized$bestTune, fit\_caret\_robust$bestTune)  
 all\_best\_Models = list(glmnet(trainx, trainy, alpha=fit\_caret\_penalized$bestTune$alpha, lambda=lambdalist),  
 fit\_caret\_robust$finalModel)  
 all\_best\_RMSE = c(min(fit\_caret\_penalized$results$RMSE),min(fit\_caret\_robust$results$RMSE))  
   
 # the best model of each fold  
 one\_best\_Type = all\_best\_Types[which.min(all\_best\_RMSE)]  
 one\_best\_Pars = all\_best\_Pars[which.min(all\_best\_RMSE)]  
 one\_best\_Model = all\_best\_Models[[which.min(all\_best\_RMSE)]]  
 one\_best\_RSME = all\_best\_RMSE[[which.min(all\_best\_RMSE)]]  
   
 # for checking later to see what's the best from each fold  
 allbestTypes[j] = one\_best\_Type  
 allbestPars[[j]] = one\_best\_Pars  
 allbestRMSE[j] = one\_best\_RSME  
 allbestModel[[j]] = one\_best\_Model  
   
 allpredictedMethod[groupj] = one\_best\_Type  
 allpredictedMSE[groupj] = one\_best\_RSME  
   
 if (one\_best\_Type == "ELASTICNET") {  
 ENETLAMBDA = one\_best\_Pars[[1]]$lambda  
 allpredictedCV[groupj] = predict(one\_best\_Model,newx=validx,s=ENETLAMBDA)  
 } else if (one\_best\_Type == "ROBUST") {  
 allpredictedCV[groupj] = predict(one\_best\_Model,newdata=validdata)  
 }  
}  
})  
  
paste("Elapsed times: ", timing["elapsed"])

## [1] "Elapsed times: 107.28"

stopCluster(cls)  
  
########################################  
### Print Best Model of Each Fold  
########################################  
allbestTypes

## [1] "ROBUST" "ROBUST" "ROBUST" "ROBUST" "ROBUST"

for (j in 1:nfolds) {  
 writemodel = paste("The best model at fold", j,   
 "is of type", allbestTypes[j],  
 ", its RMSE", round(allbestRMSE[j],5),  
 "with parameter(s)",allbestPars[[j]])  
 print(writemodel, quote = FALSE)  
}

## [1] The best model at fold 1 is of type ROBUST , its RMSE 0.22304 with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at fold 2 is of type ROBUST , its RMSE 0.2196 with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at fold 3 is of type ROBUST , its RMSE 0.22075 with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at fold 4 is of type ROBUST , its RMSE 0.22786 with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at fold 5 is of type ROBUST , its RMSE 0.21994 with parameter(s) list(intercept = TRUE, psi = 3)

########################################  
### The best model overall  
########################################  
BestTypeOverall = allbestTypes[which.min(allbestRMSE)]  
BestModelOverall = allbestModel[which.min(allbestRMSE)]  
BestParamOverall = allbestPars[which.min(allbestRMSE)]  
BestRMSEOverall = allbestRMSE[which.min(allbestRMSE)]  
  
print(paste("Best Model: ", BestTypeOverall), quote = FALSE)

## [1] Best Model: ROBUST

print(paste("Best Model RMSE: ", BestRMSEOverall), quote = FALSE)

## [1] Best Model RMSE: 0.219602739314875

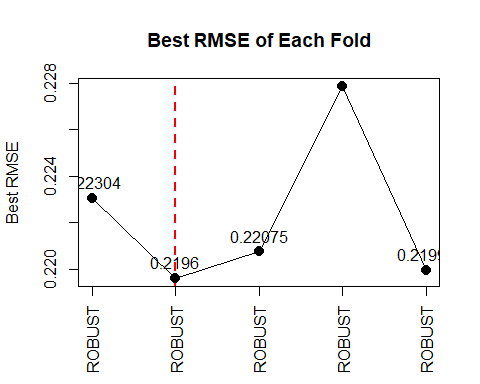
print(BestParamOverall[[1]], quote = FALSE)

## [[1]]  
## intercept psi  
## 5 TRUE psi.hampel

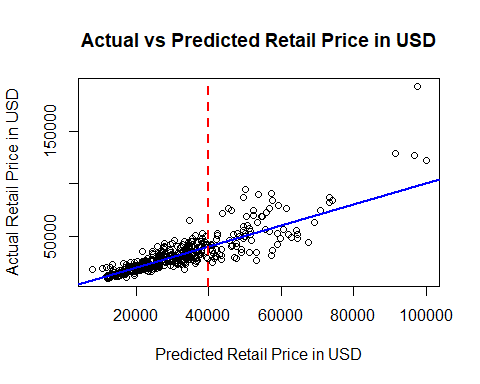
if (BestTypeOverall == "ELASTICNET") {  
 bestcoef = coef(BestModelOverall[[1]], s = one\_best\_Pars[[1]]$lambda)  
} else if (BestTypeOverall == "ROBUST") {  
 bestcoef = coef(BestModelOverall[[1]])  
}  
options(scipen=999)  
round(bestcoef,4)

## (Intercept) logHorsepower TypeRWD TypeAWD SUV   
## 4.0233 1.1615 0.1930 0.1551 -0.0445   
## Sport Wagon Minivan Pickup   
## 0.0467 0.0210 0.0306 -0.3907

# Best Model RMSE for each fold  
plot(c(1:nfolds),allbestRMSE, main = "Best RMSE of Each Fold",  
 xaxt="n", type = "l", xlab = "", ylab = "Best RMSE", pch = 19)   
abline(v = which.min(allbestRMSE), col = "red", lwd = 2, lty = 2)  
points(allbestRMSE ~ c(1:nfolds), pch = 20, cex = 2)  
axis(side = 1, at = 1:length(allbestTypes), labels = allbestTypes, las = 2)  
text(x = c(1:nfolds), y = allbestRMSE, labels = round(allbestRMSE, 5), pos = 3, offset = 0.5)



########################################  
### Outer model assessments  
########################################  
  
y = myAuto$logRetailprice  
  
# Actual VS Predicted Retail Price plot  
plot(exp(y)~exp(allpredictedCV), main = "Actual vs Predicted Retail Price in USD",   
 xlab="Predicted Retail Price in USD", ylab="Actual Retail Price in USD")  
abline(a = 0, b = 1, col = "blue", lwd = 2)  
abline(v = 40000, col = "red", lwd = 2, lty = 2)



# get CV assessment  
CV.assess = mean((allpredictedCV-y)^2)  
print(paste("CV assessment: ", CV.assess), quote = FALSE)

## [1] CV assessment: 0.0493584407507645

# get RMSE assessment  
RMSE = sqrt(mean((allpredictedCV-y)^2))  
print(paste("RMSE assessment: ", RMSE), quote = FALSE)

## [1] RMSE assessment: 0.222167596086298

# get R2 assessment  
R2 = 1-sum((allpredictedCV-y)^2)/sum((y-mean(y))^2)  
print(paste("R2 assessment: ", R2), quote = FALSE)

## [1] R2 assessment: 0.789311152477344

# MAE  
MAE = mean(abs(exp(allpredictedCV)-exp(y)))  
print(paste("Predicted retail price is off +/- USD ",round(MAE,2)), quote = FALSE)

## [1] Predicted retail price is off +/- USD 6131.21

pct\_err.median = round((MAE/summary(exp(y))[3])\*100,2)  
print(paste("Percent error: ",pct\_err.median," from median retail price"), quote = FALSE)

## [1] Percent error: 22.05 from median retail price

pct\_err.mean = round((MAE/summary(exp(y))[4])\*100,2)  
print(paste("Percent error: ",pct\_err.mean," from mean retail price"), quote = FALSE)

## [1] Percent error: 18.72 from mean retail price

About 78.93% of the variability in Retailprice values is explained by this model-fitting process

Predicted retail price is off +/- USD 6131.21 Percent error: 22.05 from median retail price Percent error: 18.72 from mean retail price

### Question 6

print(paste("The best model",one\_best\_Type))

## [1] "The best model ROBUST"

set.seed(8)  
fit\_rlm = rlm(Model.Robust, data=myAuto.reduced, psi=psi.hampel)  
bestcoef =coef(fit\_rlm)  
options(scipen=999)  
round(bestcoef,4)

## (Intercept) logHorsepower TypeRWD TypeAWD SUV   
## 3.9234 1.1809 0.1763 0.1694 -0.0642   
## Sport Wagon Minivan Pickup   
## 0.0807 0.0202 -0.0103 -0.3838

imp <- as.data.frame(varImp(fit\_rlm))  
imp <- data.frame(overall = imp$Overall,  
 names = rownames(imp))  
imp[order(imp$overall,decreasing = T),]

## overall names  
## 1 30.4506997 logHorsepower  
## 8 7.7805710 Pickup  
## 2 5.6996198 TypeRWD  
## 3 5.3872692 TypeAWD  
## 5 2.1247495 Sport  
## 4 1.8285325 SUV  
## 6 0.4750612 Wagon  
## 7 0.2057223 Minivan

The two most important predictors in the model are logHorsepower and Pickup.

A 1% increase in horsepower is associated with an estimated increase of about exp(1.1809) = 3.26 times in the value of Retailprice, when all other variables are held constant.

The retail price of a pickup truck is estimated to be about exp(-0.3838) = 0.681 times or 68.1% of the retail price of a non-pickup vehicle or about 31.9% lower than that of a non-pickup vehicle, when all other variables are held constant.