6372: Unit 2 Homework

Matt Farrow

HW Instructions

The weekly HW assignments are designed to accomplish 2 goals for the MSDS student. The first is to provide a series of conceptual and analtical questions so the student can get a feel for their current understanding of the unit. The second goal is to introduce the students to standard functions and routines in R that effectively do the same things that the “Procs” do in SAS.

R and SAS are both wonderful tools and as we go through the assignments, students will begin to recognize very quickly that they both have pros and cons.

The formatting of the HW is as follows:

1. A series of high level questions will be asked with either short answers or simple multiple choice responses.
2. Analytical questions will be provided but a short vignette example of how R functions work for a given topic or method will be given. The student will then be asked a follow up question or two based on the output provided.

3.Thirdly, a new data set will be given to allow the student to gain some experience with a new data set from start to finish. This part of the homework is for your own practice and is not due at the time of HW submission.

Solutions to the HW will be provided a day or two after the HW is submitted. It is up to the student to “shore up” any confusion or missunderstanding of a topic. Grading will be based on a combination of correctness, completion, and overall conciseness.

MLR Conceptual questions

1. State the necessary assumptions for a multiple linear regression model to be valid in terms of conducting hypothesis tests and providing prediction intervals.

- **The assumptions for a multiple linear regression model to be valid include: residuals that are normally distributed, constant variance, and independent observations.**

2. True or False? It is required that the response variable of an MLR model be normally distributed.

- **True**

3. Feature selection such as forward, backward, stepwise, or penalized techniques provide analysts the ability to do…what? For MLR, what can a feature selection method not do without the guidance of the analyst?

- **Feature selection allows analysts to determine which predictors are relevant in a regression model. One thing that a feature selection method not do with out the guidance of the analyst is perform transformations on the variables being considered.**

4. Supose that the the true relationship between a response variable ,y, and explanatory variable x is $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$. If the analysts final model includes a third, fourth, and fifth polynomial term, state whether the test ASE will be higher or lower than the training ASE. Is this an example of a model having high bias or high variance?

- **In this example, if additional polynomial terms are added, the ASE will be higher than the training ASE. This is an example of a model having high variance.**

Exercise #1 EDA

The first step of any model building process is to conduct a thorough exploratory data analysis (EDA) once the main study question and goals are defined. To motivate this topic using R we will use a widely known data set involving modeling miles per gallon (mpg) of cars with numerous makes and models. The data set is available in the package “ISLR”.

Let’s view the data first below. The goal of this study is to understand what explanatory variables contribute to the variation in the mpg values. Among the variables in the data set, we have information such as how many cylinders the vehicle has, year it was made, horsepower, displacement, acceleration, and origin. Cylinders and origin are categorical variables. Origin is the only one that is a little cryptic. There are 3 categorical levels: 1-US, 2-Europe, and 3-Japan/Asia.

```
library(ISLR)
head(Auto)
```

…	cylinders	displacement	horsepower	wei…	acceleration	y…	origin	name
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fctr>
1 18	8	307	130	3504	12.0	70	1	chevrolet chevelle
2 15	8	350	165	3693	11.5	70	1	buick skylark 320
3 18	8	318	150	3436	11.0	70	1	plymouth satellite
4 16	8	304	150	3433	12.0	70	1	amc rebel sst
5 17	8	302	140	3449	10.5	70	1	ford torino
6 15	8	429	198	4341	10.0	70	1	ford galaxie 500

6 rows

When categorical variables have levels that are coded as numeric it is important to make sure that R is treating them as categorical. The following code ensures that origin and cylinders are indeed categorical and then the “attach” function allows us to call the variable names in the data set Auto without having to use the dollar sign convention.

```
Auto$cylinders <- as.factor(Auto$cylinders)
Auto$origin <- as.factor(Auto$origin)
attach(Auto)
```

To better understand what variables we have access to, some simple summary statistics for each variable are helpful for identifying any obvious outliers that could be recording errors or just simply understanding the range of each predictor so we do not extrapolate any predictions if that is something we are asked to do later.

A quick and dirty summary is to just use summary.

```
summary(Auto)
```

```
##       mpg         cylinders  displacement     horsepower        weight
##  Min.   : 9.00   3:  4     Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   4:199     1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   5:  3     Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   6: 83     Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   8:103     3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60             Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year       origin                  name
##  Min.   : 8.00   Min.   :70.00   1:245   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   2: 68   ford pinto        :  5
##  Median :15.50   Median :76.00   3: 79   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98           amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00           amc hornet        :  4
##  Max.   :24.80   Max.   :82.00           chevrolet chevette:  4
##                                          (Other)           :365
```

For categorical explanatory variables it can be helpful to view summary statistics by each categorical level. Examining cylinders below we can quantify the shifts a little better using means. Note we could swap out the “summary” function for any other function we would want to statistics on.

```
# Summary statistics by categorical level
t(aggregate(mpg ~ cylinders, data = Auto, summary))
```

```
##             [,1]       [,2]       [,3]       [,4]       [,5]
## cylinders   "3"        "4"        "5"        "6"        "8"
## mpg.Min.    "18.00000" "18.00000" "20.30000" "15.00000" " 9.00000"
## mpg.1st Qu. "18.75000" "25.00000" "22.85000" "18.00000" "13.00000"
## mpg.Median  "20.25000" "28.40000" "25.40000" "19.00000" "14.00000"
## mpg.Mean    "20.55000" "29.28392" "27.36667" "19.97349" "14.96311"
## mpg.3rd Qu. "22.05000" "32.95000" "30.90000" "21.00000" "16.00000"
## mpg.Max.    "23.70000" "46.60000" "36.40000" "38.00000" "26.60000"
```

```
# Means of each cylinder
t(aggregate(mpg~cylinders,data=Auto,mean))
```

```
##           [,1]       [,2]       [,3]       [,4]       [,5]
## cylinders "3"        "4"        "5"        "6"        "8"
## mpg       "20.55000" "29.28392" "27.36667" "19.97349" "14.96311"
```

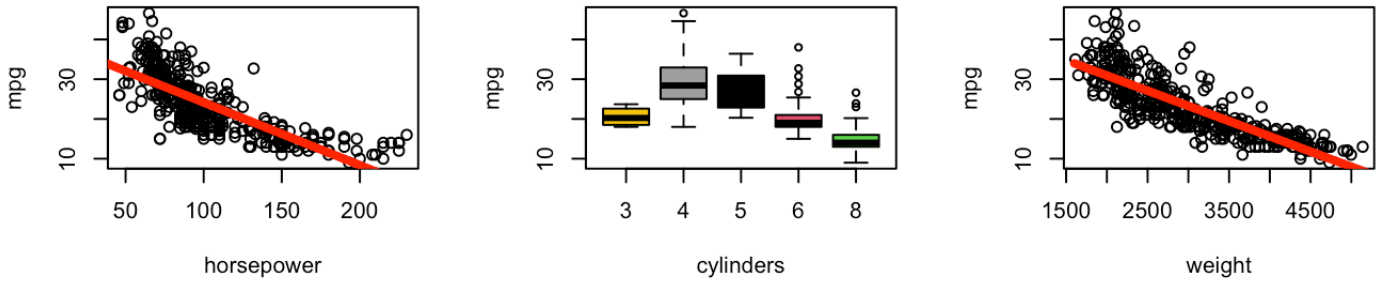
```
# Standard deviations of each cylinder
t(aggregate(mpg~cylinders,data=Auto,sd))
```

```
##           [,1]       [,2]       [,3]       [,4]       [,5]
## cylinders "3"        "4"        "5"        "6"        "8"
## mpg       "2.564501" "5.670546" "8.228204" "3.828809" "2.836284"
```

From the summary statistics, we can see that origin and cylinders are treated as categorical. It's interesting to note that there are only a few cars that have 3 and 5 cylinders. For building a model for interpretation this is probably fine as is, however if we are doing something a little more complex and utilizing cross validation or training and test set splits, not all categorical levels may be available in the training set so removing of those rare cars may be worth doing. For now we will keep them.

It is also useful to explore how the potential explanatory variables may be correlated to the response.

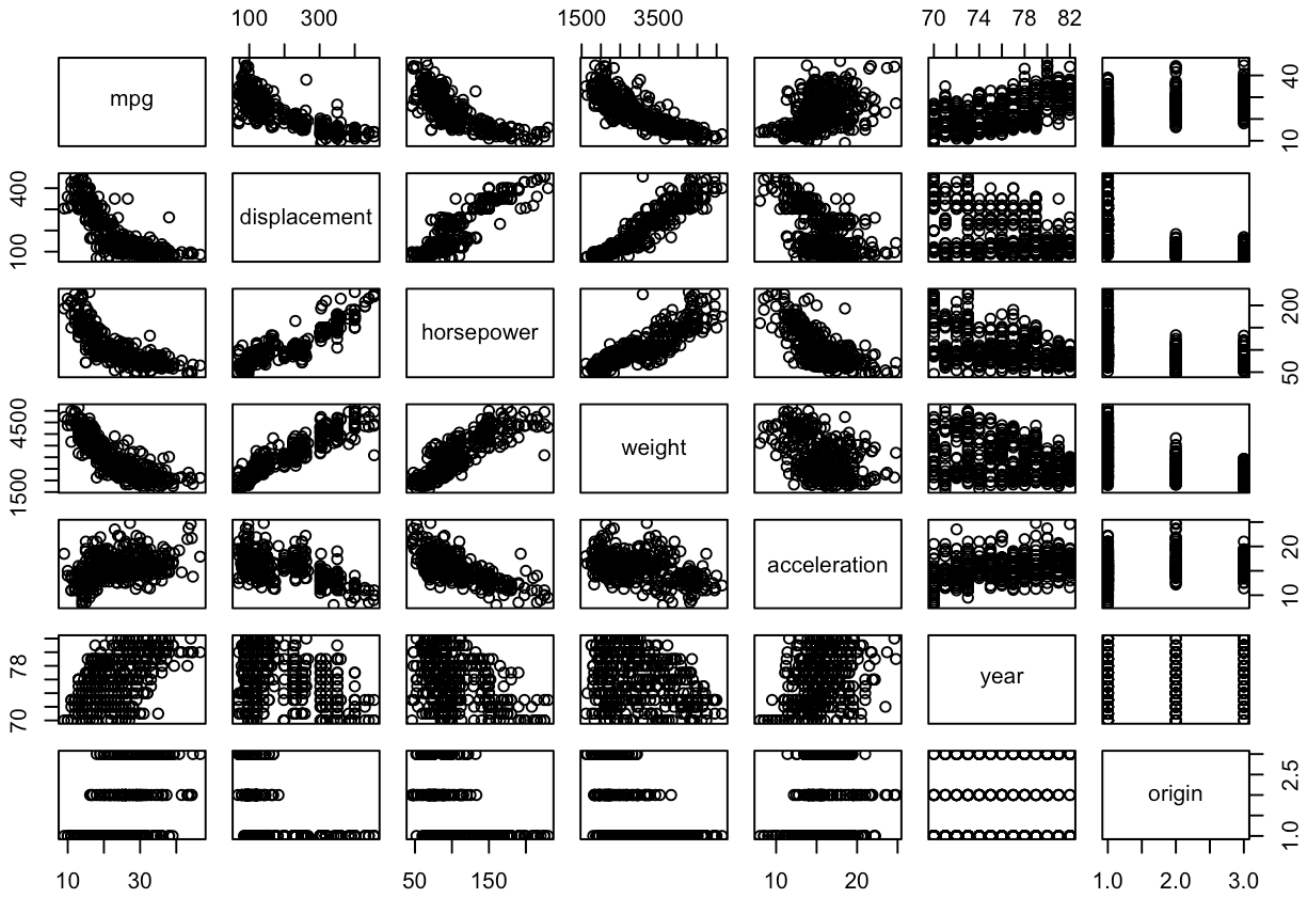
```
par(mfrow = c(1, 3))
plot(horsepower, mpg, xlab = "horsepower", ylab = "mpg")
new <- data.frame(horsepower = seq(30, 300, .1))
lines(seq(30, 300, .1),
   predict(lm(mpg ~ horsepower), newdata = new),
   col = "red",
   lwd = 4)

plot(
 as.factor(cylinders),
 mpg,
 xlab = "cylinders",
 ylab = "mpg",
 title = "Auto Data Set",
 col = c(7, 32, 57, 82, 107)
)

plot(weight, mpg)
new2 <- data.frame(weight = seq(1600, 5200, 1))
lines(seq(1600, 5200, 1),
   predict(lm(mpg ~ weight), newdata = new2),
   col = "red",
   lwd = 4)
```



There are a couple of things to note in the figures above. All three explanatory variables seem to be trending with the mpg. The more cylinders you have tends to decrease mpg (except for 3). There are also downward trends with respect to horsepower and weight which all makes sense. They appear to be nonlinear so that may be something to consider later during the modeling process. Additional graphics could be made for the other variables as well to explore.

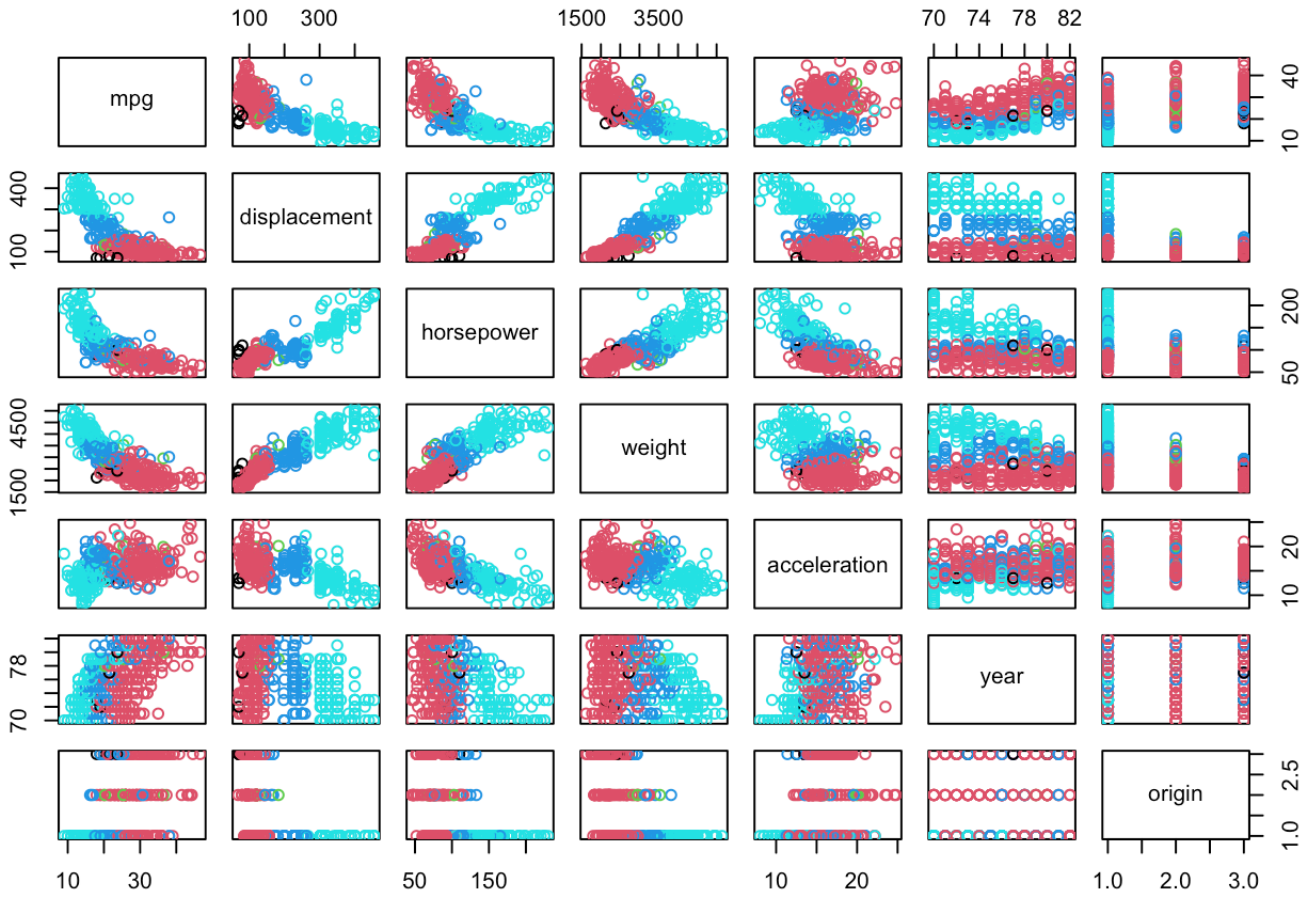
For interpretation of MLR models, it is necessary to examine the issues of multicollinearity. If the explanatory variables are highly associated with each other, regression coefficients can become unreliable and provide inconsistent interpretations. The following code provides the means to make pairwise scatter plots. It is also helpful to examine these plots while color coding for additional categorical explanatory variables.

```
pairs(Auto[, -c(2, 9)])
```



The scatter plot matrix above lets us examine the relationship each predictor has with the response (top row) and the remaining plots allow us to assess multicollinearity. If we color code the scatterplot matrix additional insight could be gained. Below is a pairs plot color coded by cylinder. Here we could visually assess if the relationships we see may depend on what cylinder group the observation belongs too. If seperate lines with different slopes for each group looks like a better fit, then an interaction could be included to try to capture that level of complexity.

```
pairs(Auto[, -c(2, 9)], col = cylinders)
```



To obtain the VIFs of each explanatory variable, we just need to run a sigle MLR fit using all the predictors and then make a vif call. *Note: in practice you can manually make decisions to remove redundant variables. You don't need to live and die by VIF assessments.* The code below provides a clever syntax trick. If your data set has nothing but the response and explanatory variables contained in the data frame and you wish to run a model using all of the predictors then you can use the following shorthand syntax.

```
library(car) # where vif function lives
```

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

```
Auto <- Auto[, -9] # removing the car name
full.model <- lm(mpg ~ ., data = Auto) # . means all variable not mpg
vif(full.model)[, 3]^2
```

```
##    cylinders displacement   horsepower       weight acceleration         year
##     1.984867    23.271418    10.559221    11.723047     2.684794     1.323483
##       origin
##     1.531211
```

Homework Question 5

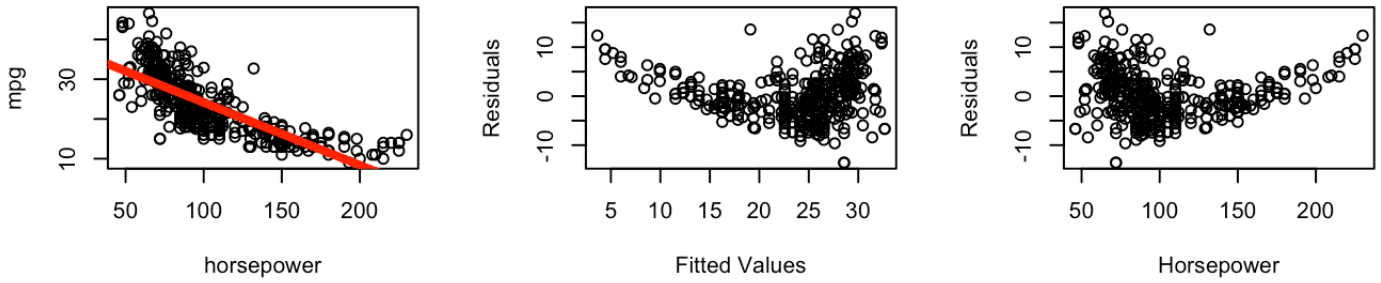
Use this section's EDA output to provide some comments on whether or not multicollinearity may be an issue for this data set. If so, what variables do you think are redundant and could be removed? You do not need to do any additional work, but you are welcome to if you want.

Based on the scatterplots, it appears that weight and displacement may be suffering from multicollinearity. A look at the VIF values indicates that `displacement`, `horsepower`, and `weight` all have values over 10, although `horsepower` is just above that threshold (10.559221).

Model Assessment and Modeling Potential Interaction/Model Complexity

Recall from the exploratory work, it looked like horsepower was associated with mpg but it might be a little nonlinear. Let's suppose for a moment that the only available predictor to us is horsepower and cylinder. How could we tell if additional complexity is needed to model the apparent nonlinear trend? Is there any other way that nonlinear trend could be explained by a MLR model? Let's take a look. Below is a simple linear regression fit of MPG on horsepower. The two right plots are residuals versus fitted and residuals vs horsepower.

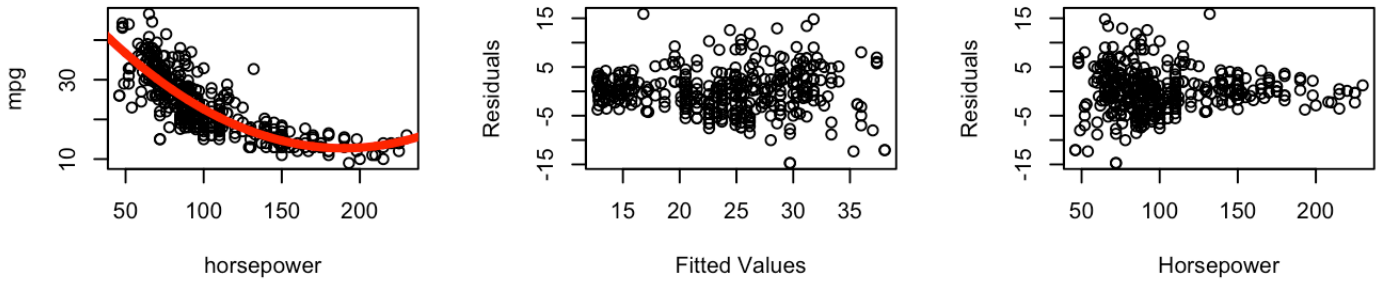
```
par(mfrow = c(1, 3))
plot(horsepower, mpg, xlab = "horsepower", ylab = "mpg")
new <- data.frame(horsepower = seq(30, 300, .1))
horse.model <- lm(mpg ~ horsepower)
lines(seq(30, 300, .1),
   predict(horse.model, newdata = new),
   col = "red",
   lwd = 4)
plot(
 horse.model$fitted.values,
 horse.model$residuals,
 xlab = "Fitted Values",
 ylab = "Residuals"
)
plot(horsepower,
   horse.model$residuals,
   xlab = "Horsepower",
   ylab = "Residuals")
```



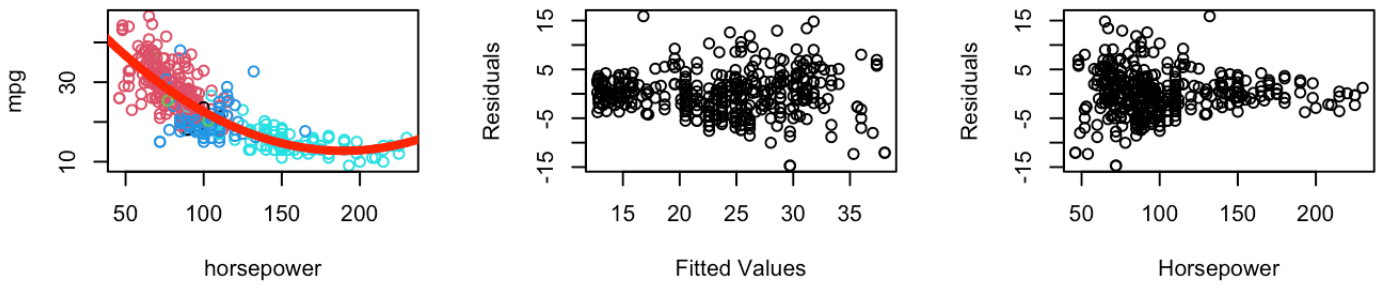
There are two noticable things going on the residual plots. One we can see as horsepower increases the variability about the trend line increases (non-constant variance). Secondly, there is a clear trend in the residuals in which it doesn't appear to be centered at 0 all the way through. This indicates that the model is not complicated enough and are predictions are biased in some areas.

One obvious approach is to include a quadratic term and follow up with some additional diagnostic looks. You can see below that the quadratic component eliminates the trending behavior in the diagnostics but constant variance is still an issue. A remedy for that is log transforming mpg.

```
par(mfrow = c(1, 3))
plot(horsepower, mpg, xlab = "horsepower", ylab = "mpg")
new <- data.frame(horsepower = seq(30, 300, .1))
horse.model2 <- lm(mpg ~ horsepower + I(horsepower ^ 2))
lines(seq(30, 300, .1),
   predict(horse.model2, newdata = new),
   col = "red",
   lwd = 4)
plot(
 horse.model2$fitted.values,
 horse.model2$residuals,
 xlab = "Fitted Values",
 ylab = "Residuals"
)
plot(horsepower,
   horse.model2$residuals,
   xlab = "Horsepower",
   ylab = "Residuals")
```

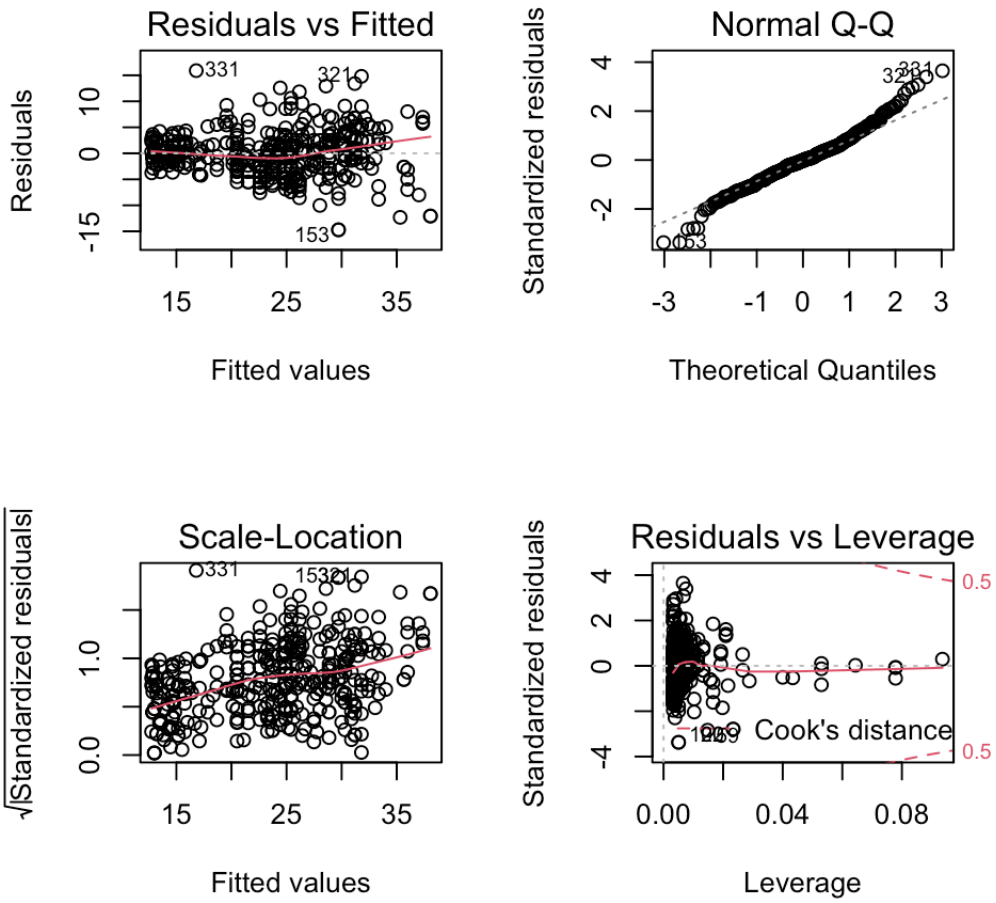


```
par(mfrow = c(1, 3))
plot(horsepower,
   mpg,
   xlab = "horsepower",
   ylab = "mpg",
   col = cylinders)
new <- data.frame(horsepower = seq(30, 300, .1))
horse.model2 <- lm(mpg ~ horsepower + I(horsepower ^ 2))
lines(seq(30, 300, .1),
   predict(horse.model2, newdata = new),
   col = "red",
   lwd = 4)
plot(
 horse.model2$fitted.values,
 horse.model2$residuals,
 xlab = "Fitted Values",
 ylab = "Residuals"
)
plot(horsepower,
   horse.model2$residuals,
   xlab = "Horsepower",
   ylab = "Residuals")
```



In general for any regression model you fit in R, the user can obtain basic residual diagnostic using the plot option. There are four plots total.

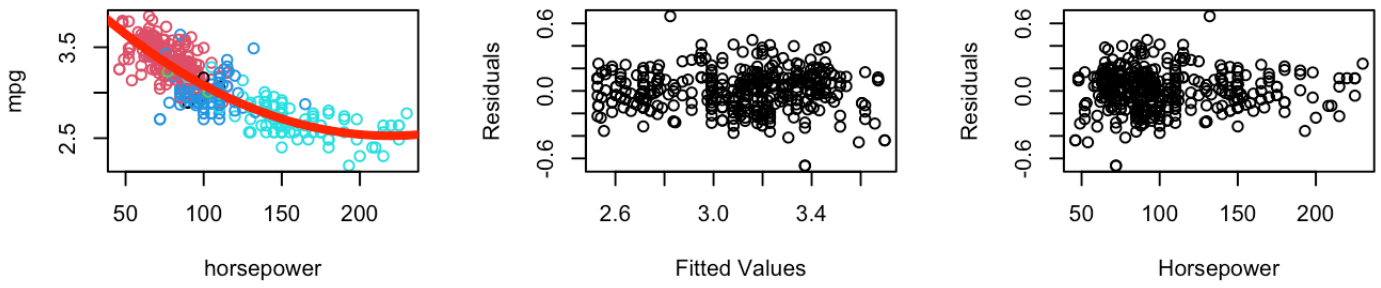
```
par(mfrow = c(2, 2))
plot(horse.model2)
```



Homework Question 6

Reanalyze the regression model (produce the same 3 figures from above) using horsepower (and its quadratic term) alone but log transforming mpg. Does the transformation fix the constant variance issue? How does normality of the residuals look? The plot of log(mpg) vs horsepower looks like the trend may be more linear now and a quadratic term is not needed? Use residual diagnostics and the summary function (summary(yourmodel)) to determine if a quadratic term is needed after log transforming.

```
par(mfrow = c(1, 3))
plot(horsepower,
   log(mpg),
   xlab = "horsepower",
   ylab = "mpg",
   col = cylinders)
new <- data.frame(horsepower = seq(30, 300, .1))
horse.model2 <- lm(log(mpg) ~ horsepower + I(horsepower ^ 2))
lines(seq(30, 300, .1),
   predict(horse.model2, newdata = new),
   col = "red",
   lwd = 4)
plot(
 horse.model2$fitted.values,
 horse.model2$residuals,
 xlab = "Fitted Values",
 ylab = "Residuals"
)
plot(horsepower,
   horse.model2$residuals,
   xlab = "Horsepower",
   ylab = "Residuals")
```



The log transformation of mpg does appear to have fixed the constant variance issue, and the quadratic term may no longer be needed. The residuals appear to be a more normally distributed cloud of points as well which speaks well of the log transformation.

```
summary(horse.model2)
```

```
##
## Call:
## lm(formula = log(mpg) ~ horsepower + I(horsepower^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66460 -0.12041  0.00316  0.11349  0.66376
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.402e+00  7.260e-02  60.639  < 2e-16 ***
## horsepower      -1.711e-02  1.255e-03 -13.632  < 2e-16 ***
## I(horsepower^2)  3.901e-05  4.922e-06   7.925 2.44e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1764 on 389 degrees of freedom
## Multiple R-squared:  0.7324, Adjusted R-squared:  0.731
## F-statistic: 532.2 on 2 and 389 DF,  p-value: < 2.2e-16
```

Feature Selection and the Bias-Variance Trade Off

As discussed during the live session, feature selection is a helpful tool to safeguard our model building processes from building models that are potentially too simple or overly complex. To get a sense of how to replicate tools we have seen in SAS, lets import the golf data set that we used in our prelive assignment and play around. (NOTE: There are more integrated R pacakges for example caret that incorporate other feature selection technques and predictive models. More integrated often translates to more “black box” and I want to make sure we have a good sense of this critical concept.)

The leaps packages has a very straightforward tool to use for forward selection. Regsubsets basically works like a general lm regression call with the added options of selecting the type of selection process and the maximum number of steps you want to go.

```
library(leaps)
golf <- read.csv(here::here("6372", "week 02", "pre-live", "GolfData2.csv"), header =
T)
golf <- golf[, -c(1, 8, 9, 10)]
reg.fwd <- regsubsets(log(AvgWinnings) ~ ., data = golf, method = "forward", nvmax =
20)
```

The object that I created, reg.model, contains the results of not only the final regression model but all of the models through the iterative process. This allows us to make graphics similar to the SAS diagnostics of glmselect.

Using the summary command we can extract metrics like ASE, AIC, and BIC for each of the steps. The vector of results are listed in order of model complexity (1 predictor up to nvmax). Some examples are below along with a simple visualization which is helpful.Keep in mind at this point, we are not doing any CV or even training and test set splits. We just using the entire data set as a training set.

```
summary(reg.fwd)$adjr2
```

```
##  [1] 0.3297415 0.5660627 0.5851334 0.5939983 0.5997587 0.6069883 0.6134332
##  [8] 0.6173809 0.6205702 0.6224992 0.6235913 0.6247310 0.6258664 0.6262647
## [15] 0.6267603 0.6267177 0.6260190 0.6247145 0.6234114 0.6220942
```

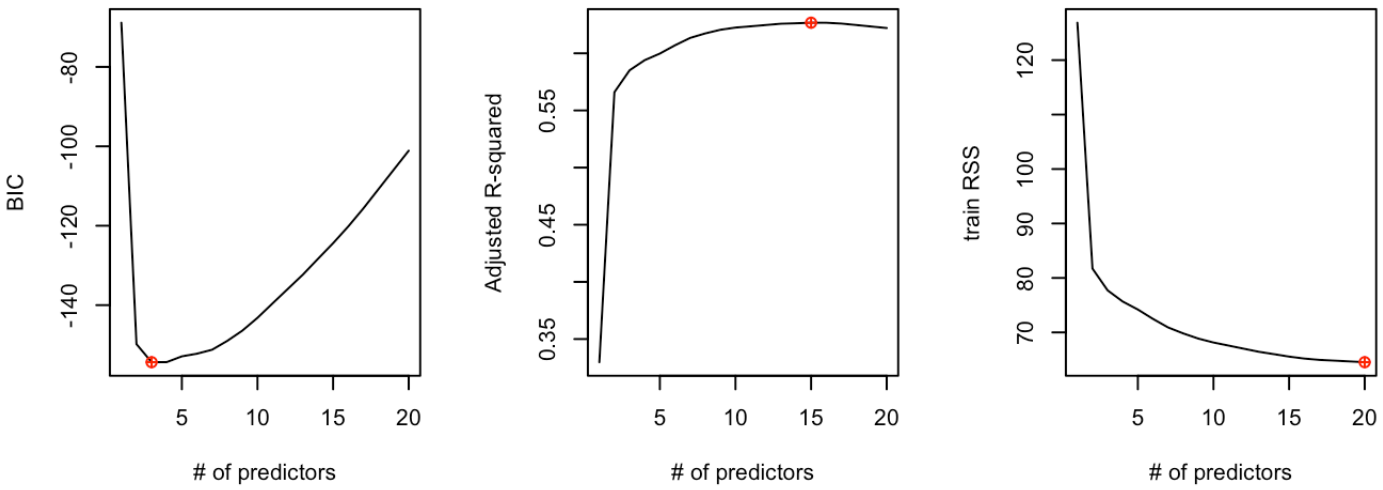
```
summary(reg.fwd)$rss
```

```
##  [1] 126.87220  81.71594  77.71987  75.66302  74.19898  72.47526  70.90956
##  [8]  69.81210  68.85996  68.14156  67.57715  67.00639  66.43861  66.00321
## [15]  65.55152  65.19479  64.95193  64.81231  64.66991  64.52738
```

```
summary(reg.fwd)$bic
```

```
##  [1]  -68.86949 -149.81790 -154.36684 -154.34573 -152.89730 -152.22620
##  [7] -151.22870 -149.00780 -146.42122 -143.19870 -139.55078 -135.93513
## [13] -132.32490 -128.33546 -124.40328 -120.19473 -115.64810 -110.79175
## [19] -105.94475 -101.09908
```

```
par(mfrow = c(1, 3))
bics <- summary(reg.fwd)$bic
plot(1:20, bics, type = "l", ylab = "BIC", xlab = "# of predictors")
index <- which(bics == min(bics))
points(index, bics[index], col = "red", pch = 10)

adjr2 <- summary(reg.fwd)$adjr2
plot(1:20, adjr2, type = "l", ylab = "Adjusted R-squared", xlab = "# of predictors")
index <- which(adjr2 == max(adjr2))
points(index, adjr2[index], col = "red", pch = 10)

rss <- summary(reg.fwd)$rss
plot(1:20, rss, type = "l", ylab = "train RSS", xlab = "# of predictors")
index <- which(rss == min(rss))
points(index, rss[index], col = "red", pch = 10)
```



When dealing with only a training data set metrics like AIC and BIC are the most appropriate as they penelize the more complex models. (See the BIC graph). Now lets visit the train/test set split for checking model adequacy.

First lets simply split the golf data set into training and split and fit a forward selection model on the training data set.

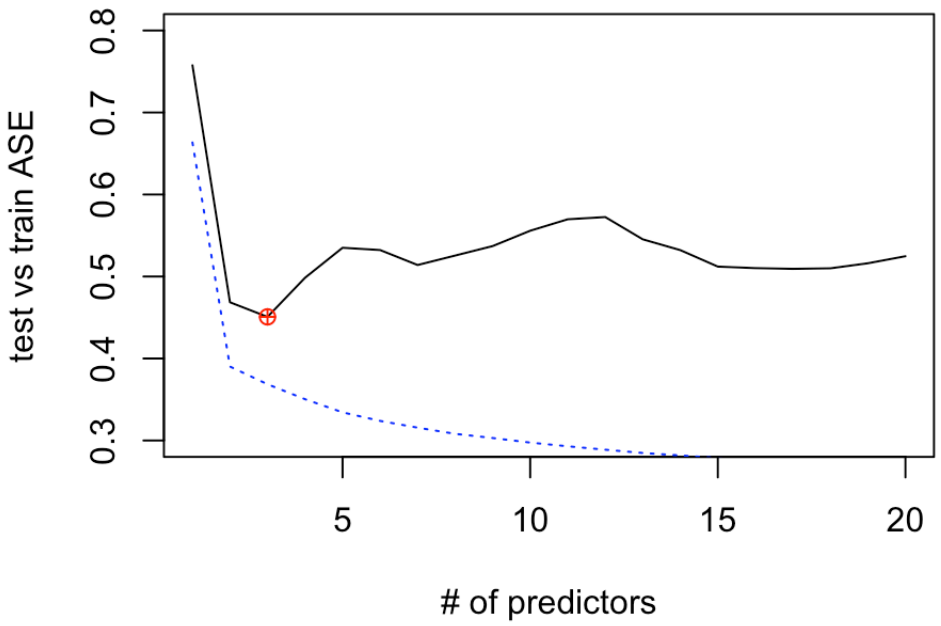
```
set.seed(1234)
index <- sample(1:dim(golf)[1], 100, replace = F)
train <- golf[index, ]
test <- golf[-index, ]
reg.fwd <- regsubsets(log(AvgWinnings) ~ ., data = train, method = "forward", nvmax =
20)
```

Once we have our model fits on the training data set, all we need to do is predict the models onto the training data set and produce ASE type plots for each step of the forward selection. Courtesy of the ISLR texbook, a function is provided to easily predict the forward selection results on a test set.

```
# Really handy predict function
predict.regsubsets <- function(object, newdata, id, ...) {
 form <- as.formula(object$call [[2]])
 mat <- model.matrix(form, newdata)
 coefi <- coef(object, id = id)
 xvars <- names(coefi)
 mat[, xvars] %*% coefi
}
```

With the prediction function read in we can simply write a loop to predicted on each of the 20 models generated from the forward selection procedure and plot the ASE's. I've included the training ASE for comparison.

```
testASE <- c()
# note my index is to 20 since that what I set it in regsubsets
for (i in 1:20) {
 predictions <- predict.regsubsets(object = reg.fwd, newdata = test, id = i)
 testASE[i] <- mean((log(test$AvgWinnings) - predictions)^2)
}
par(mfrow = c(1, 1))
plot(1:20, testASE, type = "l", xlab = "# of predictors", ylab = "test vs train ASE",
ylim = c(0.3, 0.8))
index <- which(testASE == min(testASE))
points(index, testASE[index], col = "red", pch = 10)
rss <- summary(reg.fwd)$rss
lines(1:20, rss / 100, lty = 3, col = "blue") # Dividing by 100 since ASE=RSS/sample
size
```



From the test ASE graphic, we see (via the red dot) that the minimum Average Squared Error happens with 3 predictors included in the model and its value is around 0.4. The ASE on the test gives us a metric on how reproducible the models being fitted would be have on a future data set. It doesn't really help us get at how well the prediction accuracy is though. To finish things off and after additional model fitting and trial and error, once a model is deemed the final model it is usually standard practice to fit the entire data set once more and make your final reports and conclusions using all of the data (more information to estimate parameters).

```
reg.final <- regsubsets(log(AvgWinnings) ~ ., data = golf, method = "forward", nvmax
= 4)
coef(reg.final, 3)
```

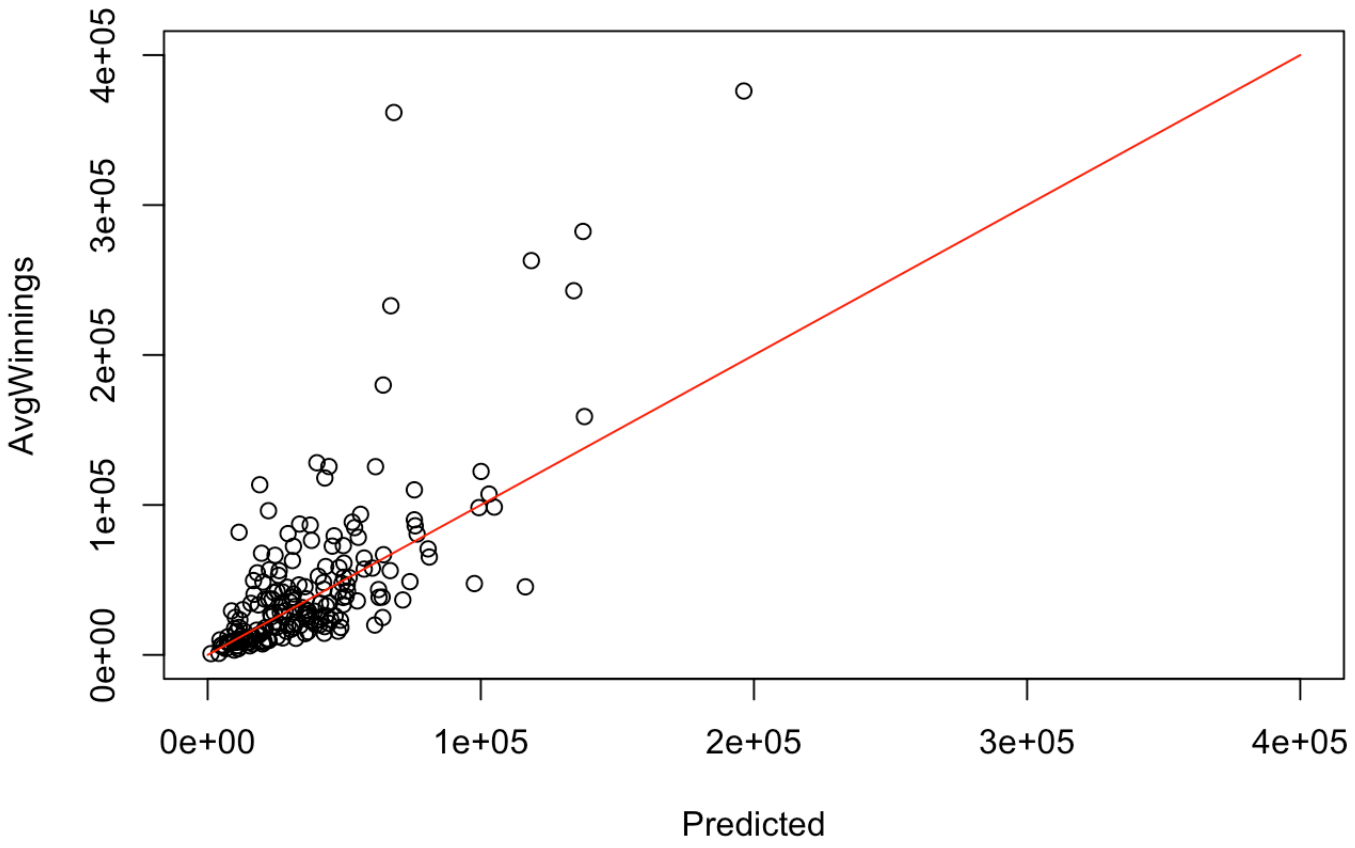
```
##  (Intercept)       Greens     AvgPutts         Save
##  32.43907362   0.17683646 -19.68231695   0.02791842
```

```
final.model <- lm(log(AvgWinnings) ~ Greens + AvgPutts + Save, data = golf)
summary(final.model)
```

```
##
## Call:
## lm(formula = log(AvgWinnings) ~ Greens + AvgPutts + Save, data = golf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3831 -0.4167 -0.0511  0.4411  1.9641
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  32.439074   4.173575   7.772 4.55e-13 ***
## Greens        0.176836   0.016234  10.893  < 2e-16 ***
## AvgPutts    -19.682317   2.080889  -9.459  < 2e-16 ***
## Save          0.027918   0.008886   3.142  0.00194 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6362 on 192 degrees of freedom
## Multiple R-squared:  0.5915, Adjusted R-squared:  0.5851
## F-statistic: 92.68 on 3 and 192 DF,  p-value: < 2.2e-16
```

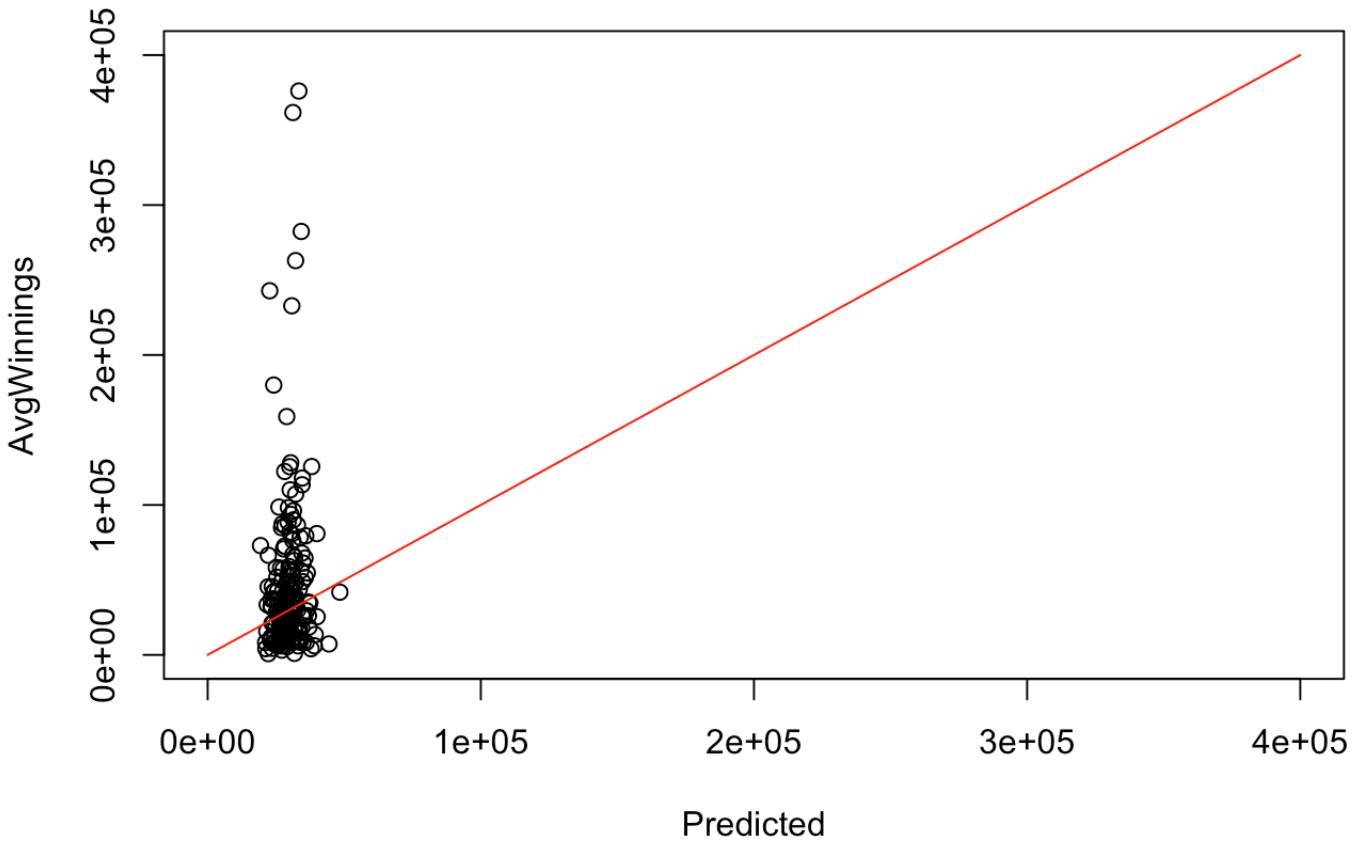
Remember that the output of the final model in terms of reporting p-values and interpretation, you still must check the model assumptions to make sure everything is valid. As mentioned earlier the ASE plots are really great at helping to select a final model that doesn't have too much bias or variance, it is hard to tell how well the predictions are. This depends on the scale of the response variable. A helpful graphic that SAS produces as well is the true response versus the predictions. This can give you a sense of the uncertainty in the prediction. More variability equates to more uncertainty.

```
plot(exp(final.model$fitted.values), golf$AvgWinnings, xlab = "Predicted", ylab = "Av
gWinnings", xlim = c(0, 400000), ylim = c(0, 400000))
lines(c(0, 400000), c(0, 400000), col = "red")
```



Another helpful thing that this graph shows that also would come up in the residual diagnostics is that our predictive modelis going to under predict those extreme golfers who are making more than everybody else. We would have need some other explantory variable to help capture what is going on out there.

To help with the idea of this graph imagine we fit another predictive model but just used 3 of the random variables. This should clearly have less predictive ability and we could verify that with a test ASE assessment. Below is the predictions of the “bogus” model versus the true Avg Winnnings. We can see the predictions are much worse as they vary much more around the red line.



Probably the best way is to make a few predictions and examine and take a look at their prediction intervals. The tighter the interval the better model you have and you can make some practical sense from there. Another helpful thing to do would be to take a look at some prediction intervals. These are on the log scale.

```
head(predict(final.model, golf, interval = "predict"))
```

```
##         fit      lwr      upr
## 1  9.373349 8.095833 10.65087
## 2 11.113277 9.843192 12.38336
## 3  9.806945 8.545929 11.06796
## 4  9.739179 8.479230 10.99913
## 5  9.946832 8.679149 11.21452
## 6 10.673640 9.412905 11.93437
```

Putting things back on the raw scale, we can see the certainty in our predictions. For example, the first predicted average winnings for a golfer with Green=58.2, AvgPutts=1.767, and Save=50.9 is $11994.16 with a prediction interval of 3247.77 to 44,294. The interval is quite large and illustrates just how variable average winnings are even though there are a few key predictors that are statistically relevent and the model producing results that make some sense.

Homework Question 7

Use the Auto data set and perform the following tasks. Provide your R script for these answers.

1. Split the data set into a training and test set that is roughly 50/50. Before doing so delete the observations that have a 3 or 5 cylinders. This will make your life easier. To keep everyone consistent with each other, make sure you set the seed number “set.seed(1234)” before making the split.

```
##
## 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
```

```
set.seed(1234)

# Create data set
df <- Auto %>%
  filter(!cylinders %in% c(3, 5))

# Build test and training data sets
df_index <- sample(1:dim(df)[1], 192, replace = F)
df_train <- df[df_index, ]
df_test <- df[-df_index, ]
```

2. By choosing to delete the 3 and 5 cylinder cars from the model what does that change, if anything, about the conclusions one could make from an analysis conducted from this new reduced data set?

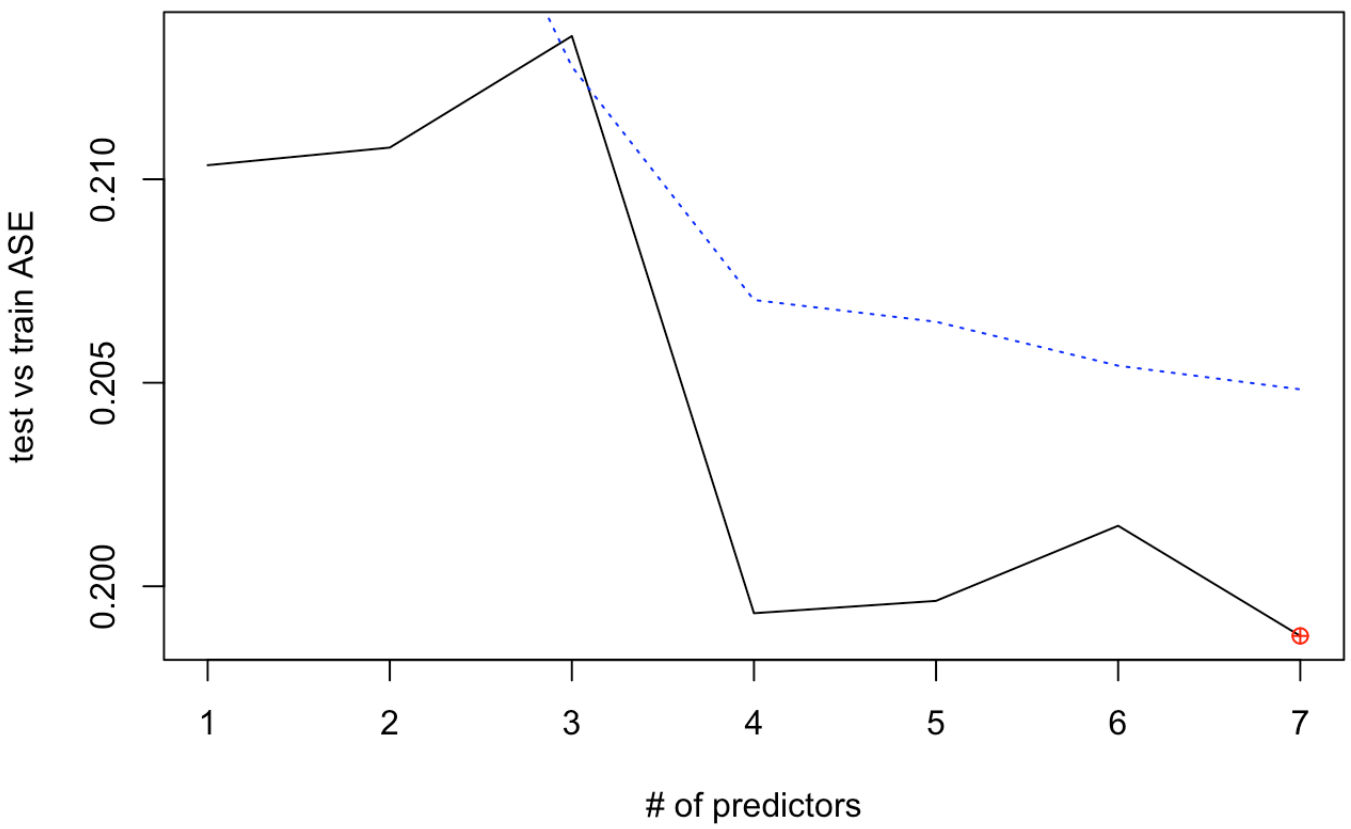
The only conclusions we could draw from the data set would relate to cars with 4, 6, or 8 cylinders since 3- and 5-cylinder cars were excluded.

3. Perform a forward selection on the training data set and produce the ASE plot comparing both the training and test set ASE metrics using the following set of predictions:

- `displacement`
- `horsepower`
- `weight`
- `acceleration`
- `year`
- `origin`

Determine how many predictors should be included. Set the nvmax to 7 for this problem.

```
# Run forward selection
df_fwd <-
  regsubsets(cylinders ~ .,
             data = df_train,
             method = "forward",
             nvmax = 7)

# Really handy predict function
predict_regsubsets <- function(object, newdata, id, ...) {
 form <- as.formula(object$call [[2]])
 mat <- model.matrix(form, newdata)
 coefi <- coef(object, id = id)
 xvars <- names(coefi)
 mat[, xvars] %*% coefi
}

# Create empty test ASE
testASE <- c()

# For loop
for (i in 1:7) {
  predictions <-
    predict.regsubsets(object = df_fwd,
                       newdata = df_test,
                       id = i)
  testASE[i] <- mean((as.numeric(df_test$cylinders) - predictions) ^ 2)
}

par(mfrow = c(1, 1))
plot(
  1:7,
  testASE,
  type = "l",
  xlab = "# of predictors",
  ylab = "test vs train ASE",
)
index <- which(testASE == min(testASE))
points(index, testASE[index], col = "red", pch = 10)
rss <- summary(df_fwd)$rss
lines(1:7, rss / 192, lty = 3, col = "blue")
```



4. Using your decision from #3, fit a final model using the entire data set and produce the residual diagnostics. Does your model look reasonable or do you think you should consider some additional iterations? Give a brief comment. You do not need to act on the comments.

```
reg_final <- regsubsets(cylinders ~ ., data = df, method = "forward", nvmax = 7)
coef(reg_final, 6)
```

```
##   (Intercept)           mpg  displacement    horsepower        weight
##  7.838402e-01 -3.133506e-02  1.205506e-02 -6.953028e-03  8.296032e-05
##          year       origin3
##  1.673431e-02  1.560137e-01
```

```
final_model <- lm(cylinders ~ mpg + displacement + horsepower + weight + year + origi
n, data = df)
```

```
## Warning in model.response(mf, "numeric"): using type = "numeric" with a factor
## response will be ignored
```

```
## Warning in Ops.factor(y, z$residuals): '-' not meaningful for factors
```

```
summary(final.model)
```

```
##
## Call:
## lm(formula = log(AvgWinnings) ~ Greens + AvgPutts + Save, data = golf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3831 -0.4167 -0.0511  0.4411  1.9641
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  32.439074   4.173575   7.772 4.55e-13 ***
## Greens        0.176836   0.016234  10.893  < 2e-16 ***
## AvgPutts    -19.682317   2.080889  -9.459  < 2e-16 ***
## Save          0.027918   0.008886   3.142  0.00194 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6362 on 192 degrees of freedom
## Multiple R-squared:  0.5915, Adjusted R-squared:  0.5851
## F-statistic: 92.68 on 3 and 192 DF,  p-value: < 2.2e-16
```

5. *(optional)* What happens when you try to do forward selection using all of the variables in the data set? Can you potentially give some reasons why warnings/errors start popping up when doing so? Hint: Using table(train$cylinder$, $train$origin) maybe helpful among other simple summary stats and graphics.

LASSO Calls

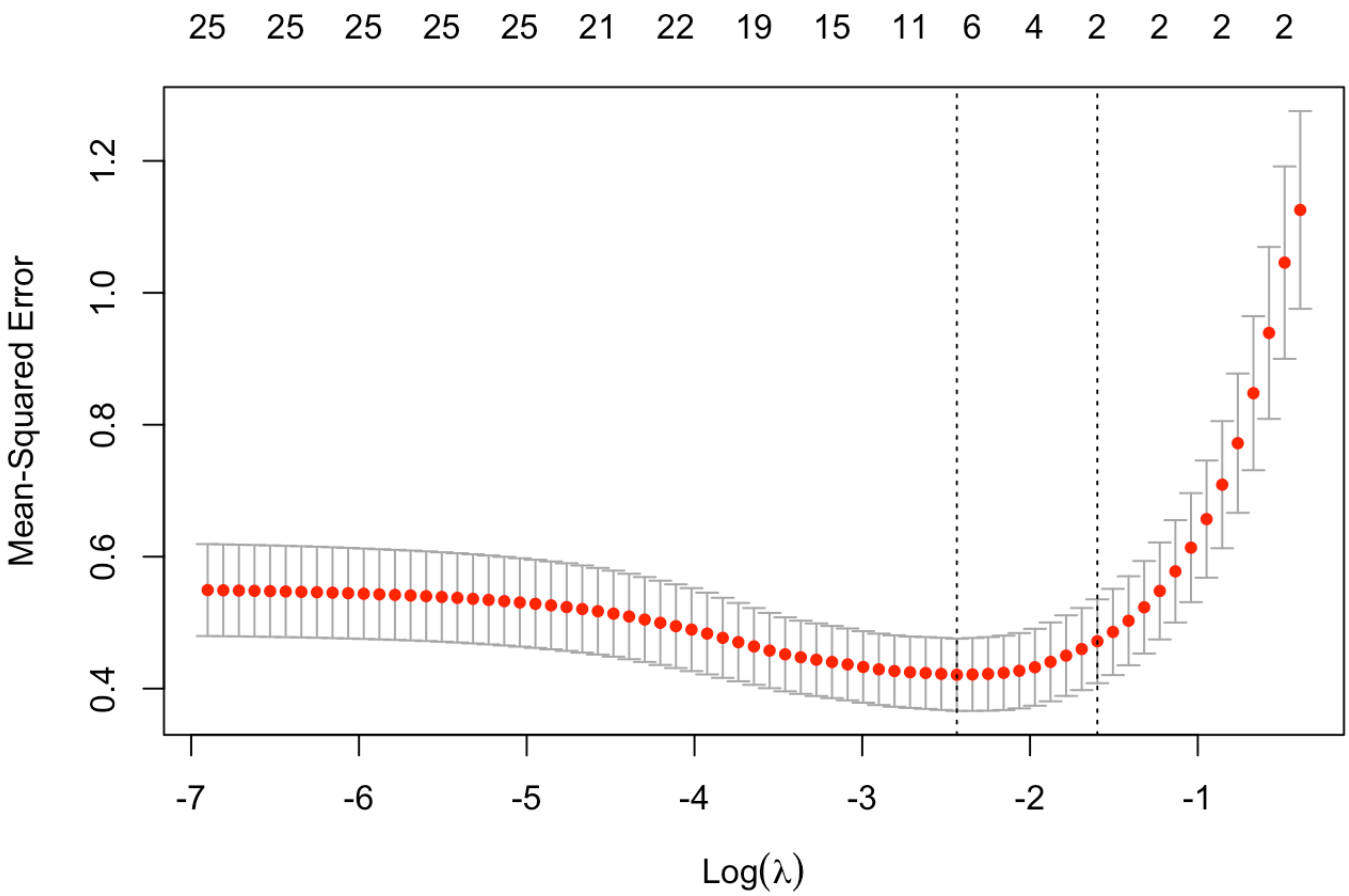
There are no homework questions involving this section but it is here for the sake of your project. Peforming a lasos call is pretty straight forward. GLM-NET peforms 10-fold CV to determine an optimal penalty parameter. The coefficients are easy to extract and making predictions are straight forward. It is possible to make ASE style plots like we did previously but it takes a little extra programming. See me if interested.

```
library(glmnet)
```

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

```
## Loaded glmnet 4.0-2
```

```
# Formatting data for GLM net
x <- model.matrix(AvgWinnings ~ ., train)[, -1]
y <- log(train$AvgWinnings)

xtest <- model.matrix(AvgWinnings ~ ., test)[, -1]
ytest <- log(test$AvgWinnings)

grid <- 10^seq(10, -2, length = 100)
lasso.mod <- glmnet(x, y, alpha = 1, lambda = grid)

cv.out <- cv.glmnet(x, y, alpha = 1) # alpha=1 performs LASSO
plot(cv.out)
```




```
bestlambda <- cv.out$lambda.min # Optimal penalty parameter. You can make this call v
isually.
lasso.pred <- predict(lasso.mod, s = bestlambda, newx = xtest)

testMSE_LASSO <- mean((ytest - lasso.pred)^2)
testMSE_LASSO
```

```
## [1] 0.4462266
```

Note here that the ASE (MSE) on the test set is very similar to the forward selection results previously examined. The number of predictors in this LASSO fit is larger than what forward selection, but examining the coefficients on the predictors we can see that many of them are quite small and not really contributing much.

```
coef(lasso.mod, s = bestlambda)
```

```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                           1
## (Intercept)  39.026083813
## Age           .
## AvgDrive      .
## DriveAcc      .
## Greens        0.178208217
## AvgPutts    -18.398564728
## Save          0.013895993
## V12           .
## V13           .
## V14          -0.002177072
## V15           .
## V16           .
## V17          -0.007051063
## V18           .
## V19           .
## V20          -0.058328895
## V21           0.001492951
## V22           .
## V23           .
## V24           .
## V25           .
## V26           .
## V27          -0.036016824
## V28           .
## V29           .
## V30          -0.015255088
## V31           .
```