Analyzing Murder Population Rate

Alejandro Pesantez

Introduction:

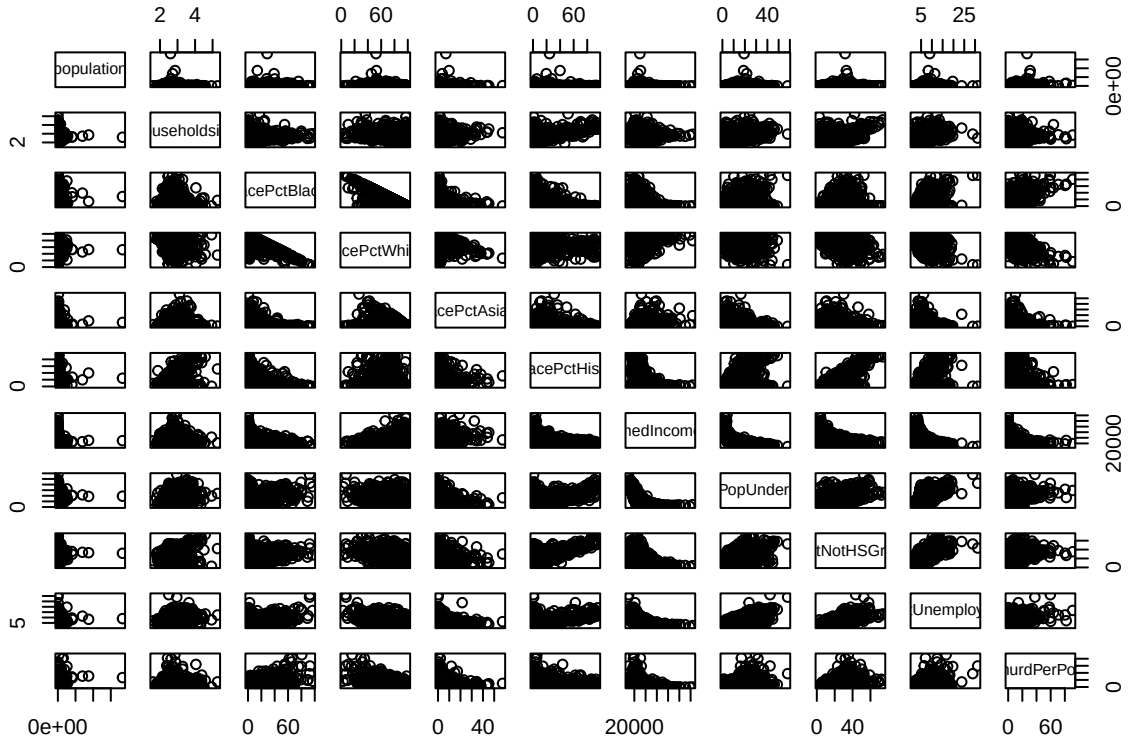
The objective of this report is to figure out what variables are best at predicting the murder per population rate. This is important because people do not want to be or live in areas where there is a high murder per population rate. Figuring out what variables effect the murder per population rate will give an understanding of what kind of areas have a low or high rate, and can also give ideas on ways to prevent murders which would be beneficial in a multitude of ways. This analysis was done in a step by step process which will be outlined and explained below.

Step 1:

In step 1 the data was read in and the column names needed to be changed in order for the data to make sense. Once this was done I picked a few variables that I thought were correlated or in other words good predictors of the murder per population rate. Below is a pairs plot which shows if variables are correlated with each other and if they are correlated with the murder per population rate.

```
library(tidyverse)
data <- read_delim("CommViolPredUnnormalizedData.txt",na = "?", col_names = FALSE,
                   show_col_types = FALSE)
column_names <- c("communityName","state","countyCode","communityCode","fold","population"
                  ,"householdsize","racePctBlack","racePctWhite","racePctAsian","racePctHisp"
                  ,"medIncome","PctPopUnderPov","PctNotHSGrad","PctUnemployed","murdPerPop")
data_mod <- data[,c(1:11,18,34,36,38,131)]
colnames(data_mod) <- column_names
```

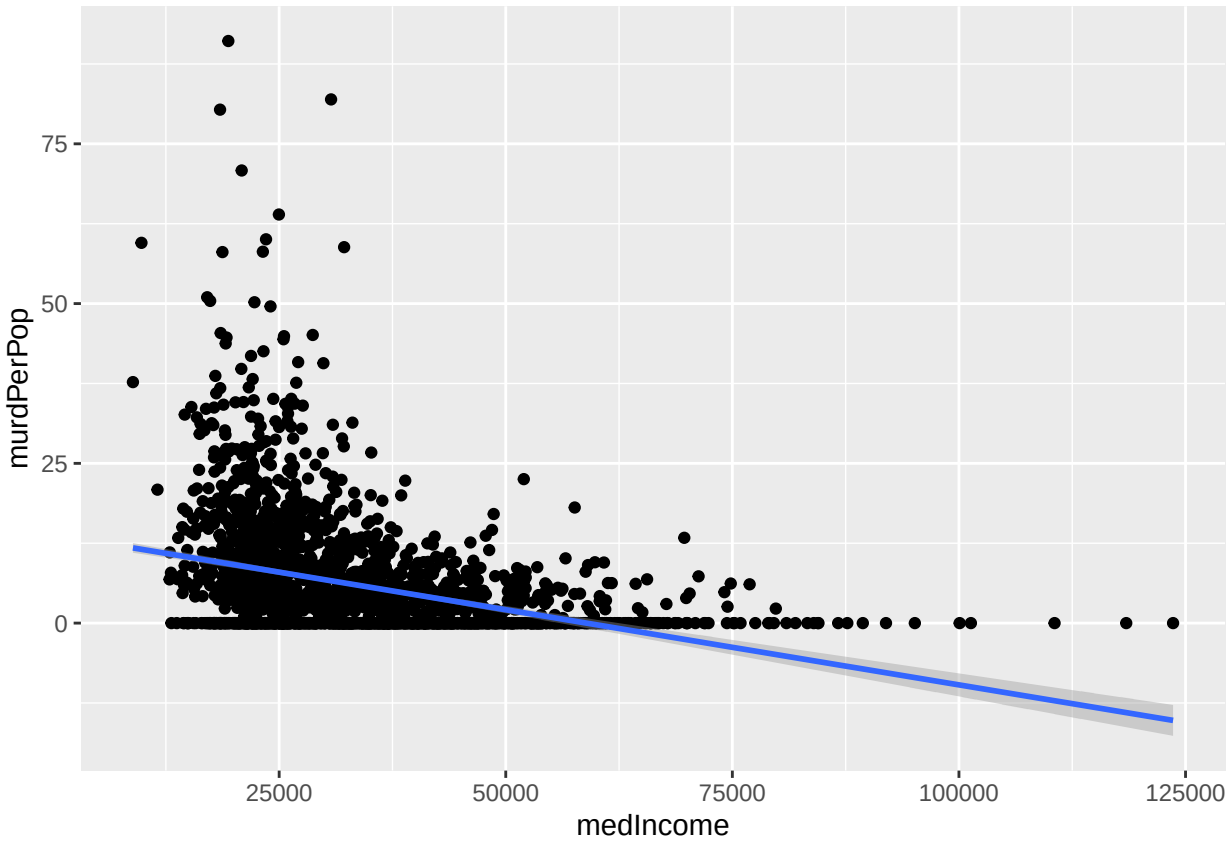
```
pairs(data_mod[,6:16])
```



After the pairs plot, the four predictors I felt were most important were put against the murder per population rate in scatter plots. I provided a brief explanation to these plots as well.

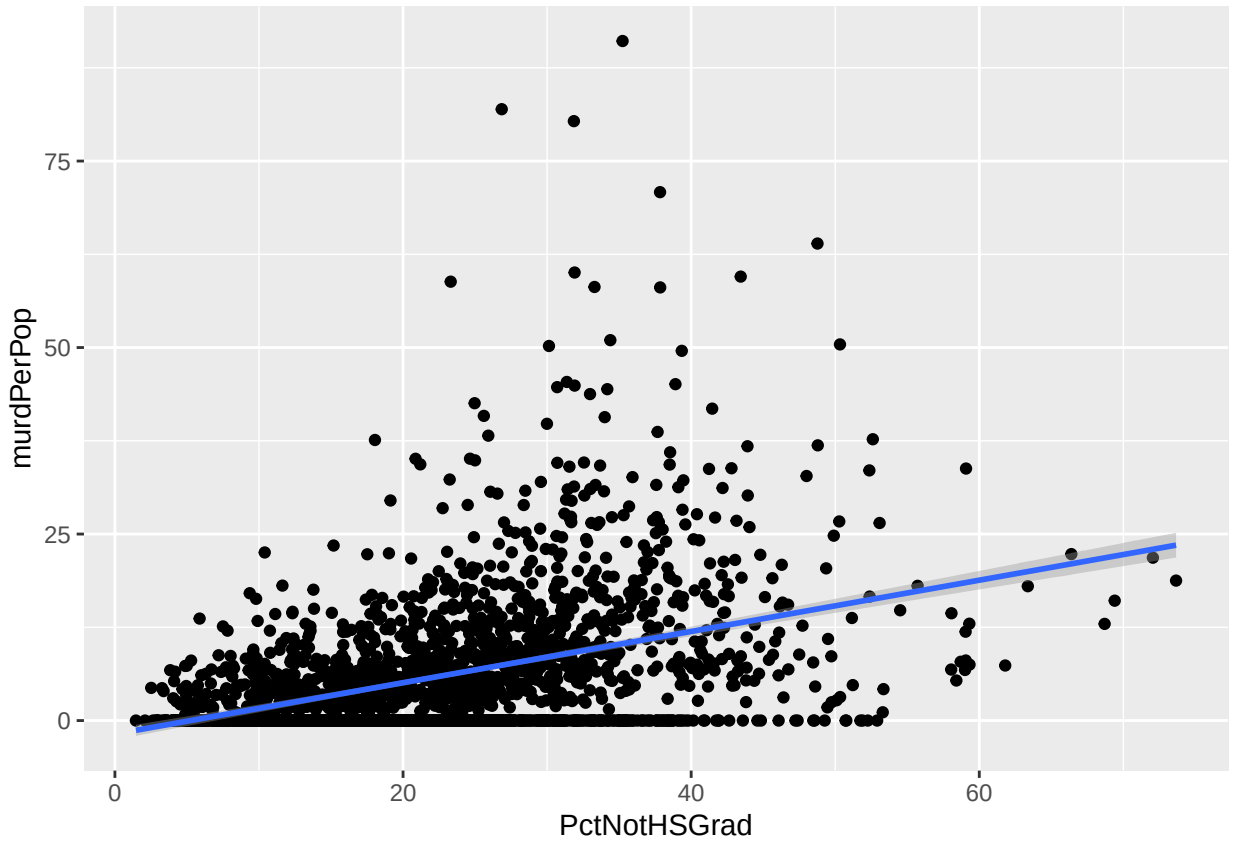
In the first scatter plot where it shows murder per population versus median income, there seems to be a general trend that as median income increases, the murder per population decreases. Which seems to mean the wealthier the area is, the lower the murder per population rate will be.

```
med_inc_plot <- ggplot(data_mod, aes(medIncome,murdPerPop)) + geom_point() + geom_smooth(method=lm)
suppressMessages(print(med_inc_plot))
```



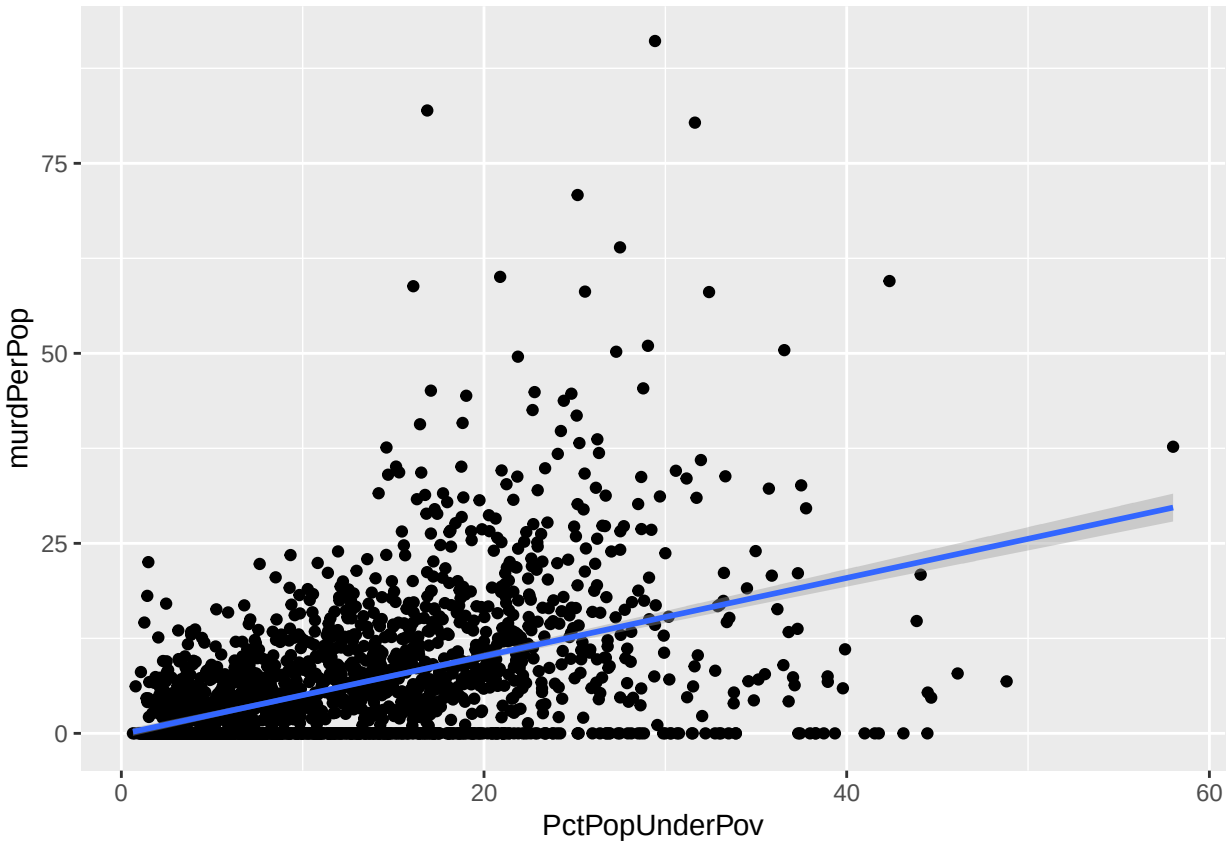
In the second scatter plot where it shows murder per population versus the percentage of people not being a high school graduate, there seems to be a general trend that as the percentage of people not being a high school graduate increases the murder per population rate increases. This seems to mean that in areas where people aren't very educated, there will most likely be a higher murder per population rate.

```
hs_grad_plot <- ggplot(data_mod, aes(PctNotHSGrad,murdPerPop)) + geom_point() + geom_smooth(method='lm'
suppressMessages(print(hs_grad_plot))
```



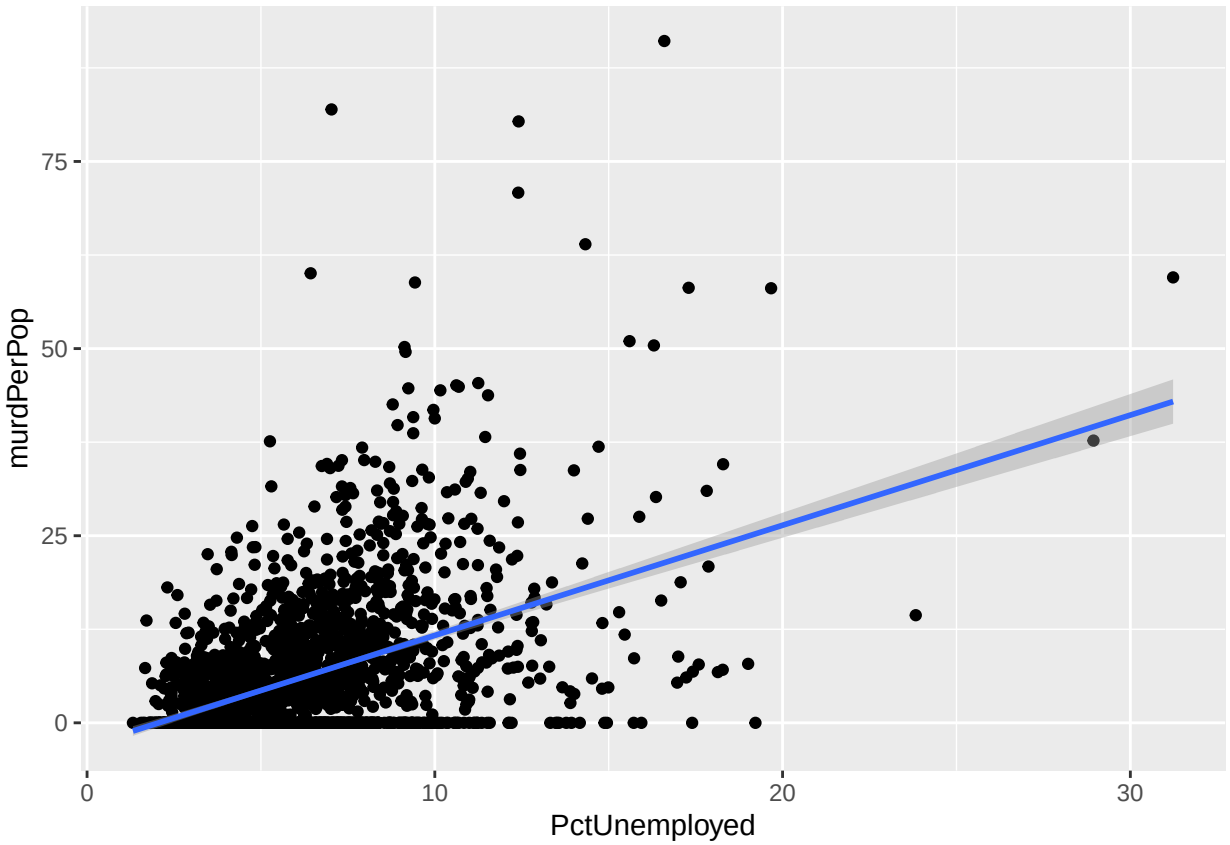
In the third scatter plot where it shows murder per population versus the percentage of people under poverty, there seems to be a general trend that as the percentage of people under poverty increases the murder per population rate increases. This seems to mean that in areas where there are a lot of people in poverty, there will most likely be a higher murder per population rate.

```
pov_plot <- ggplot(data_mod, aes(PctPopUnderPov,murdPerPop)) + geom_point() + geom_smooth(method='lm')
suppressMessages(print(pov_plot))
```



In the fourth and final scatter plot where it shows murder per population versus the percentage of people unemployed, there seems to be a general trend that as the percentage of people unemployed increases the murder per population rate increases. This seems to mean that in areas where there are a lot of people unemployed, there will most likely be a higher murder per population rate.

```
unemp_plot <- ggplot(data_mod, aes(PctUnemployed,murdPerPop)) + geom_point() + geom_smooth(method='lm')
suppressMessages(print(unemp_plot))
```



Step 2:

In step 2 a linear model was created with predictors that I felt were most important in predicting the murder per population rate. These predictors were household size, race percentage Black, race percentage White, race percentage Asian, race percentage Hispanic, median income, percentage of population under poverty, percentage of population without a high school degree, and the percentage of the population that is unemployed. Below is the output and interpretations of the model.

```
mod <- lm(murdPerPop~householdsize+racePctBlack+racePctWhite+
            racePctAsian+racePctHisp+medIncome+PctPopUnderPov+
            PctNotHSGrad+PctUnemployed,data_mod)
summary(mod)
```

```
##
## Call:
## lm(formula = murdPerPop ~ householdsize + racePctBlack + racePctWhite +
##     racePctAsian + racePctHisp + medIncome + PctPopUnderPov +
##     PctNotHSGrad + PctUnemployed, data = data_mod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.145  -2.757  -0.692   2.019  55.127
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.306e+01  4.606e+00   7.177 9.70e-13 ***
## householdsize  -2.367e+00  5.099e-01  -4.642 3.66e-06 ***
## racePctBlack    9.226e-02  4.296e-02   2.148  0.03185 *
```

```
## racePctWhite   -2.876e-01  4.194e-02  -6.857 9.11e-12 ***
## racePctAsian   -1.685e-01  5.578e-02  -3.021  0.00254 **
## racePctHisp    -1.672e-02  2.236e-02  -0.748  0.45477
## medIncome       1.074e-05  1.953e-05   0.550  0.58238
## PctPopUnderPov  7.971e-02  3.299e-02   2.416  0.01575 *
## PctNotHSGrad    1.682e-02  2.253e-02   0.746  0.45545
## PctUnemployed   2.342e-01  8.509e-02   2.752  0.00597 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.466 on 2205 degrees of freedom
## Multiple R-squared:  0.5035, Adjusted R-squared:  0.5014
## F-statistic: 248.4 on 9 and 2205 DF,  p-value: < 2.2e-16
```

The predictors that are statistically significant in the model, or in other words good predictors of the murder rate per population, are household size, race percentage Black, race percentage White, race percentage Asian, percentage of the population under poverty, and percentage of the population unemployed. The R-squared value is 0.5035 and the adjusted R-squared value is 0.5014 which means that around 50% of the variability observed in the murder per population rate is explained by the regression model. I would say that this does match my intuition since I didn't think all of the predictors would be significant but definitely most of them. The variables that increase the murder population rate are race percentage Black, median income, percentage of the population under poverty, percentage of population without high school degree, and percentage of the population unemployed.

Step 3:

In step 3 the process of variable selection was done. The step AIC and fast backward variable selection methods were used.

The first method that was used was fast backward variable selection method. This method uses the fitted complete model and computes approximate Wald statistics by computing conditional (restricted) maximum likelihood estimates assuming multivariate normality of estimates. Which in short means, it tries a bunch of different models until it finds the model with the highest predictive power. We also made the sls .05 which just means the significance level (p-value) a variable needs to stay in the model has to meet the .05 threshold.

```
library(rms)
ols.mod <- ols(murdPerPop~householdsize+racePctBlack+racePctWhite+
            racePctAsian+racePctHisp+medIncome+PctPopUnderPov+
            PctNotHSGrad+PctUnemployed,data_mod)
fastbw(ols.mod, rule = "p", sls = 0.05)
```

```
##
##  Deleted        Chi-Sq d.f. P      Residual d.f. P      AIC   R2
##  medIncome      0.30   1    0.5823 0.30     1    0.5823 -1.70 0.503
##  PctNotHSGrad   0.33   1    0.5649 0.63     2    0.7284 -3.37 0.503
##  racePctHisp    0.32   1    0.5688 0.96     3    0.8113 -5.04 0.503
##  PctPopUnderPov 7.20   1    0.0073 8.16     4    0.0859  0.16 0.502
##
## Approximate Estimates after Deleting Factors
##
##                  Coef    S.E. Wald Z         P
## Intercept     32.9432 3.22215  10.22 0.000e+00
## householdsize -2.4529 0.47265  -5.19 2.106e-07
## racePctBlack   0.1142 0.02365   4.83 1.367e-06
## racePctWhite  -0.2781 0.02400 -11.59 0.000e+00
## racePctAsian  -0.1690 0.04378  -3.86 1.133e-04
```

```
## PctUnemployed  0.3802 0.06243   6.09 1.127e-09
##
## Factors in Final Model
##
## [1] householdsize racePctBlack  racePctWhite  racePctAsian  PctUnemployed
```

As you can see from the output above, this method chose to keep household size, race percentage Black, race percentage White, race percentage Asian, and the percentage of people unemployed in the model. The output of the new model is given below.

```
mod.fastbw <- lm(murdPerPop~householdsize+racePctBlack+racePctWhite+
                   racePctAsian+PctUnemployed, data_mod)
summary(mod.fastbw)
```

```
##
## Call:
## lm(formula = murdPerPop ~ householdsize + racePctBlack + racePctWhite +
##     racePctAsian + PctUnemployed, data = data_mod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.281  -2.736  -0.762   2.011  54.189
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   32.94324    3.22518  10.214  < 2e-16 ***
## householdsize -2.45291    0.47309  -5.185 2.36e-07 ***
## racePctBlack   0.11423    0.02367   4.825 1.49e-06 ***
## racePctWhite  -0.27812    0.02402 -11.579  < 2e-16 ***
## racePctAsian  -0.16900    0.04382  -3.857 0.000118 ***
## PctUnemployed  0.38021    0.06249   6.085 1.37e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.472 on 2209 degrees of freedom
## Multiple R-squared:  0.5016, Adjusted R-squared:  0.5005
## F-statistic: 444.7 on 5 and 2209 DF,  p-value: < 2.2e-16
```

Looking at the output of the new model, you can see that all of the variables are now statistically significant, however the adjusted r-squared went down by .0009. We then ran a second variable selection method called step AIC which just performs step wise model selection by AIC. Which means that it looks at the model's AIC and removes variables until the model achieves the best possible AIC.

```
library(MASS)
aic_result <- stepAIC(mod)
```

```
## Start:  AIC=8278.52
## murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian +
##     racePctHisp + medIncome + PctPopUnderPov + PctNotHSGrad +
##     PctUnemployed
##
##                  Df Sum of Sq   RSS    AIC
## - medIncome       1     12.65 92189 8276.8
## - PctNotHSGrad    1     23.29 92199 8277.1
## - racePctHisp     1     23.37 92200 8277.1
## <none>                        92176 8278.5
```

```
## - racePctBlack    1    192.80 92369 8281.1
## - PctPopUnderPov  1    244.11 92420 8282.4
## - PctUnemployed   1    316.67 92493 8284.1
## - racePctAsian    1    381.63 92558 8285.7
## - householdsize   1    900.65 93077 8298.1
## - racePctWhite    1   1965.37 94142 8323.2
##
## Step:  AIC=8276.82
## murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian +
##     racePctHisp + PctPopUnderPov + PctNotHSGrad + PctUnemployed
##
##                   Df Sum of Sq   RSS    AIC
## - PctNotHSGrad    1     13.85 92203 8275.2
## - racePctHisp     1     18.14 92207 8275.3
## <none>                        92189 8276.8
## - racePctBlack    1    212.46 92401 8279.9
## - PctPopUnderPov  1    275.87 92465 8281.4
## - PctUnemployed   1    320.99 92510 8282.5
## - racePctAsian    1    369.01 92558 8283.7
## - householdsize   1    925.72 93115 8297.0
## - racePctWhite    1   1953.72 94143 8321.3
##
## Step:  AIC=8275.15
## murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian +
##     racePctHisp + PctPopUnderPov + PctUnemployed
##
##                   Df Sum of Sq   RSS    AIC
## - racePctHisp     1     13.57 92216 8273.5
## <none>                        92203 8275.2
## - racePctBlack    1    206.73 92409 8278.1
## - PctPopUnderPov  1    307.46 92510 8280.5
## - PctUnemployed   1    415.25 92618 8283.1
## - racePctAsian    1    422.34 92625 8283.3
## - householdsize   1    981.43 93184 8296.6
## - racePctWhite    1   2012.38 94215 8321.0
##
## Step:  AIC=8273.48
## murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian +
##     PctPopUnderPov + PctUnemployed
##
##                   Df Sum of Sq   RSS    AIC
## <none>                        92216 8273.5
## - PctPopUnderPov  1     301.0 92517 8278.7
## - PctUnemployed   1     404.3 92621 8281.2
## - racePctAsian    1     498.6 92715 8283.4
## - racePctBlack    1     977.1 93193 8294.8
## - householdsize   1    1036.1 93252 8296.2
## - racePctWhite    1    5126.3 97343 8391.3
```

```
aic_result$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
```

```
## murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian +
##     racePctHisp + medIncome + PctPopUnderPov + PctNotHSGrad +
##     PctUnemployed
##
## Final Model:
## murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian +
##     PctPopUnderPov + PctUnemployed
##
##
##             Step Df Deviance Resid. Df Resid. Dev      AIC
## 1                                 2205   92176.20 8278.516
## 2    - medIncome  1 12.64541      2206   92188.85 8276.820
## 3 - PctNotHSGrad  1 13.85201      2207   92202.70 8275.152
## 4  - racePctHisp  1 13.57235      2208   92216.27 8273.478
```

As you can see above the step AIC model chose to remove 3 variables. This is different than what the fast backward variable selection gave us. This time the variables included in the model are household size, race percentage Black, race percentage White, race percentage Asian, percentage of population under poverty, and percentage of population unemployed. Below is the output of the new model.

```
mod.aic <- lm(murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, data_mod)
summary(mod.aic)
```

```
##
## Call:
## lm(formula = murdPerPop ~ householdsize + racePctBlack + racePctWhite +
##     racePctAsian + PctPopUnderPov + PctUnemployed, data = data_mod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.073  -2.795  -0.679   2.036  54.985
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    31.82224    3.24761   9.799  < 2e-16 ***
## householdsize  -2.35942    0.47371  -4.981 6.83e-07 ***
## racePctBlack    0.11435    0.02364   4.837 1.41e-06 ***
## racePctWhite   -0.26861    0.02425 -11.079  < 2e-16 ***
## racePctAsian   -0.15265    0.04418  -3.455  0.00056 ***
## PctPopUnderPov  0.07110    0.02648   2.685  0.00731 **
## PctUnemployed   0.24761    0.07958   3.111  0.00189 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.463 on 2208 degrees of freedom
## Multiple R-squared:  0.5032, Adjusted R-squared:  0.5019
## F-statistic: 372.8 on 6 and 2208 DF,  p-value: < 2.2e-16
```

As seen above the adjusted R-squared is higher in this model than both the original and fast backward variable selection models. I would say that this matched my intuition because the step AIC method just removed the non-statistically significant variables from the original model, which in turn gave a higher adjusted r-squared, while the fast backward variable selection method removed one more variable than the step AIC method did and got a lower adjusted r-squared since it removed a statistically significant variable.

The new step AIC model has an adjusted r-squared that is higher by .0005 than the original model from

step2.

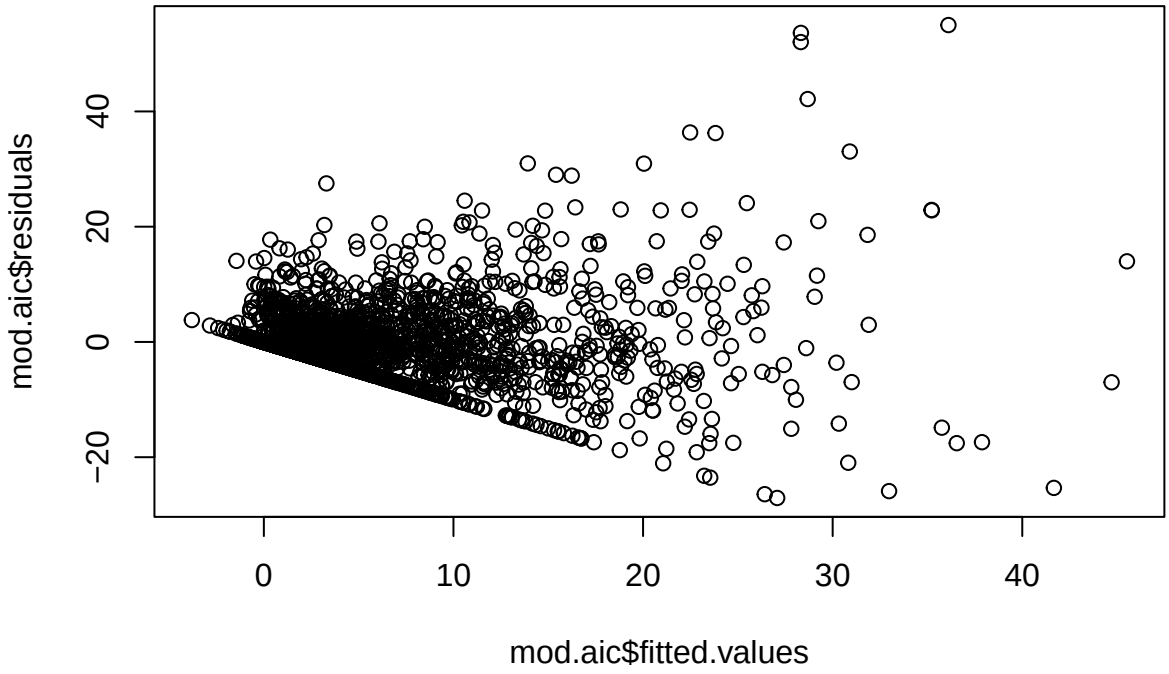
With this being the case the new model that has removed the three predictors that weren't significant from the step AIC variable selection method will be used.

Step 4:

In step 4 diagnostics of the model were done. Below is a diagnostic plot, q-q plot, and lagged residual plot to check if the assumptions of a linear regression model are upheld.

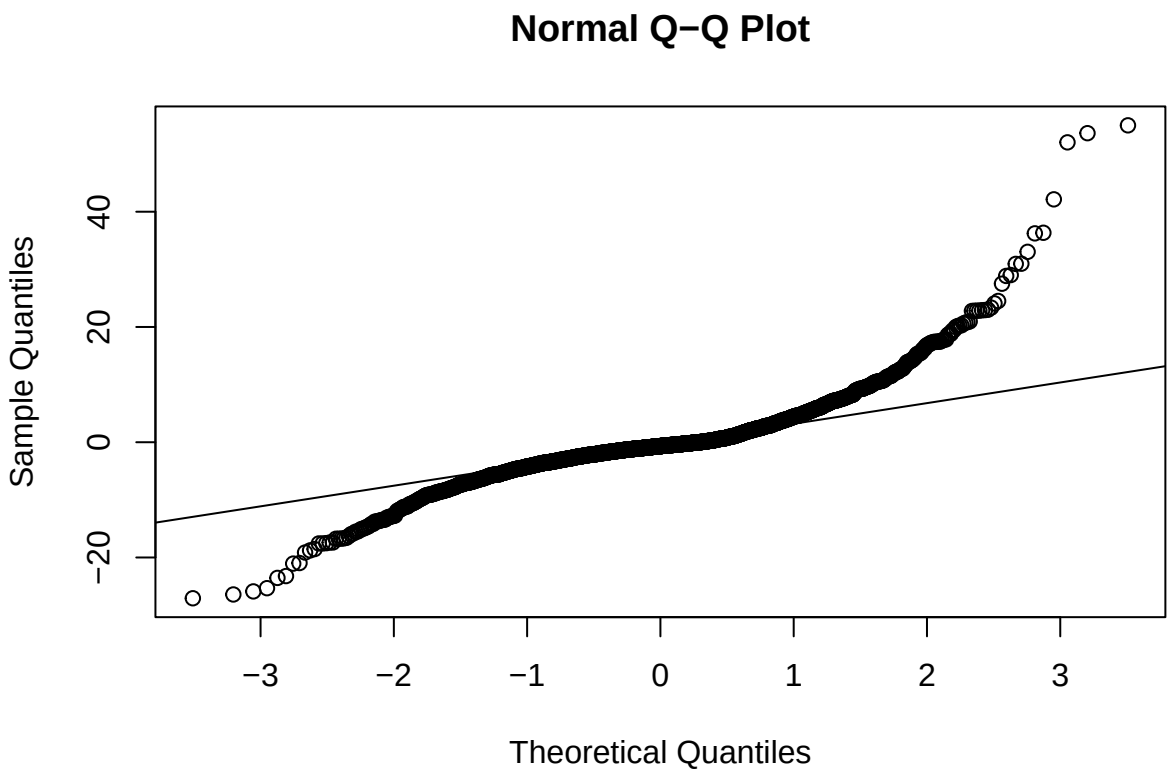
The first plot shown is the diagnostic plot. A diagnostic plot has the fitted values of the model on the x-axis, and the residuals on the y-axis. When looking at the plot I do see some bad patterns that could be examples of model violations, that being said we can say that for the diagnostic plot the assumption of constant error variance is most likely not upheld here. In order to fix this we can transform our data later on in the analysis.

```
plot(mod.aic$fitted.values,mod.aic$residuals)
```



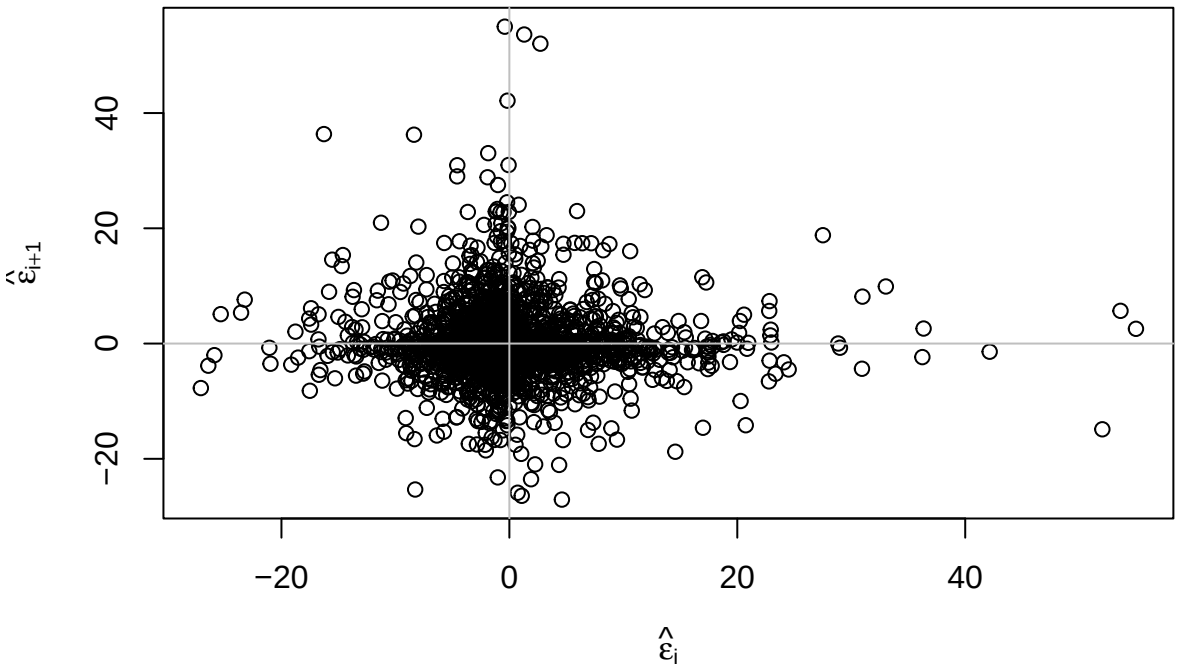
The next plot shown is the Q-Q plot. The Q-Q plot shows the sample quantities on the y-axis and the theoretical quantities on the x-axis. According to the Q-Q plot it does not seem like the assumption of normal errors is upheld since the line isn't straight, and there are some clear outliers influencing the shape of the errors. Since the line isn't straight and instead curved, this means that something is most likely skewing the data. Again we will look more into this further on in the analysis.

```
qqnorm(residuals(mod.aic))
qqline(mod.aic$residuals)
```



The last plot that is shown is the lagged residual plot. The lag residual plot works by plotting each residual value versus the value of the successive residual. In the lag residual plot below it seems that there is a pattern and the points seem to form a diamond shape. Since this is the case the assumption of uncorrelated errors seems to not be upheld since if they were uncorrelated there would be a random scatter of points above and below the e = 0 line, which in this example does not seem to be the case.

```
n <- length(residuals(mod.aic))
plot(tail(residuals(mod.aic),n-1) ~ head(residuals(mod.aic),n-1),
     xlab=expression(hat(epsilon)[i]),
     ylab=expression(hat(epsilon)[i+1]))
abline(h=0,v=0,col=grey(0.75))
```



Step 5:

In step 5, an analysis of the outliers was done. The three methods of detecting outliers that were performed were the cooks distance method, standardize residuals method, and hat value method.

The first method done was looking at cooks distance. Cook's distance is an estimate of the influence of a data point. It takes into account both the leverage and residual of each observation. Cook's Distance is a summary of how much a regression model changes when the ith observation is removed. So once we have all of the cook's distances for each of the observations, it is then put up against a threshold created by the f statistic. We make the f probability .5 so it's 50% and if the cooks distance surpasses this f threshold it is considered an outlier. As you can see below none of the observations passed the threshold which means this method detected zero outliers.

```
n <- dim(model.matrix(mod.aic))[1]
p <- dim(model.matrix(mod.aic))[2]
num_df <- p
den_df <- n - p
Fthresh <- qf(0.5, num_df, den_df)
cook_out <- which(cooks.distance(mod.aic) > Fthresh)
length(cook_out)
```

```
## [1] 0
```

The next method looked at was the standardized residuals method. In this method you standardize the residuals and if an observation is above 3 then that observation is considered an outlier. As you can see below this method detected 31 outliers.

```
stand_out <- which(rstandard(mod.aic) > 3)
length(unique(stand_out))
```

```
## [1] 31
```

The last method looked at was the hat value method. In this method you get the hat values from the model and create a hat threshold that is the number of parameters multiplied by 2, then divided by the number of observations. If the hat value passes the threshold then it is considered an outlier. Below you can see that

this method detected 229 outliers.

```
h_i <- hatvalues(mod.aic)
hat_thresh <- (2*p)/n
hat_out <- which(h_i > hat_thresh)
length(unique(hat_out))
```

```
## [1] 229
```

All of the three different methods gave a different amount of outliers. Cook's method detected 0, the standardized residual method detected 31, and the hat value method detected 229. Since this is the case I decided to not remove any outliers. This is because in real life it's unrealistic to have a perfect pattern, and given that there was a method that detected 0 outliers I decided to keep all the observations so the model would be more realistic. However, in the next step I try to transform the data to minimalize the effects of the outliers that are still present.

Step 6:

In step 6, the data is transformed to see if I can improve the model and fix our model assumptions. As seen below when a Breusch-Pagan test is run on the current model, the model assumption that the errors having constant variance is not upheld since the p-value is statistically significant. Since this is the case a box-cox and yeo-johnson transformation was done.

```
library(lmtest)
bptest(mod.aic)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mod.aic
## BP = 435.16, df = 6, p-value < 2.2e-16
```

To begin the transformations, I started with the yeo-johnson transformation. This transformation is typically done on the outcome variable using the residuals for a statistical model. Here, a simple null model is used to apply the transformation to the predictor variables individually. This can have the effect of making the variable distributions more symmetric.

At the start of the yeo-johnson transformation I created a train and test data set. I did this so I can train the model on a random 80% of the original data and then test the model on a random 20% of the original data.

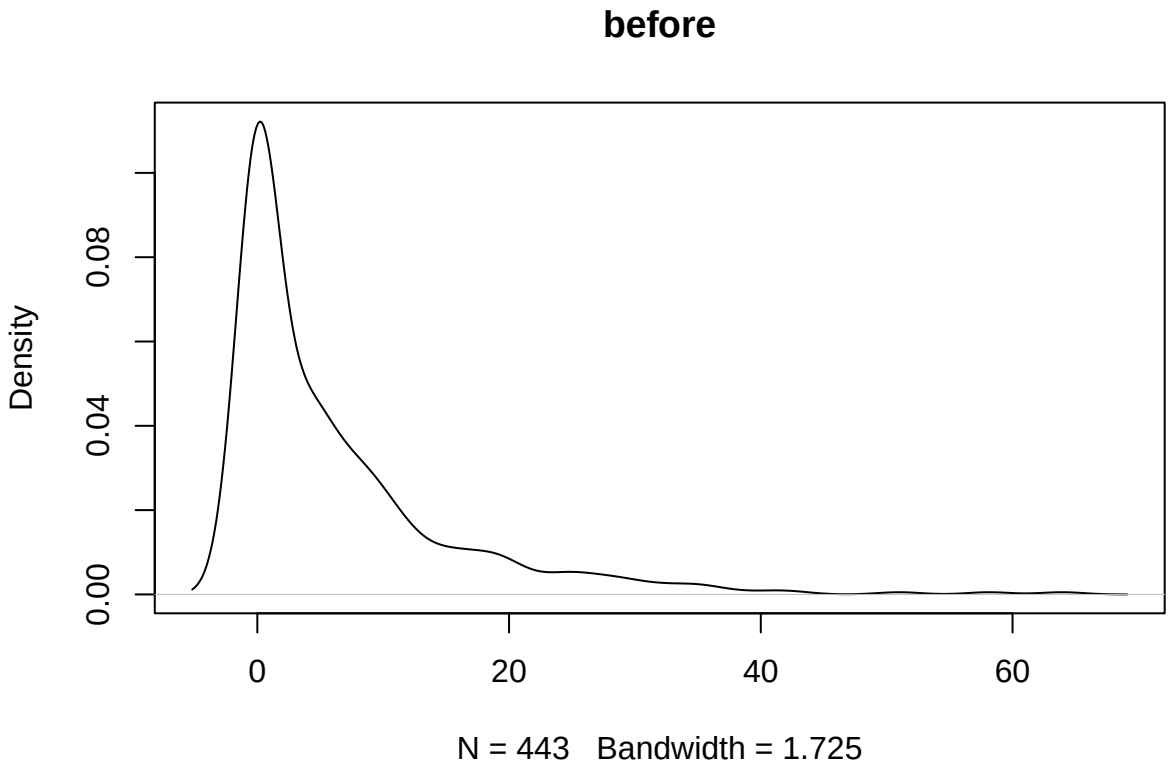
```
n <- nrow(data_mod)
set.seed(1842)
test_index <- sample.int(n,size=round(0.2*n))
train_df <- data_mod[-test_index,]
test_df <- data_mod[test_index,]
```

Below is the step AIC model using the newly created training data set.

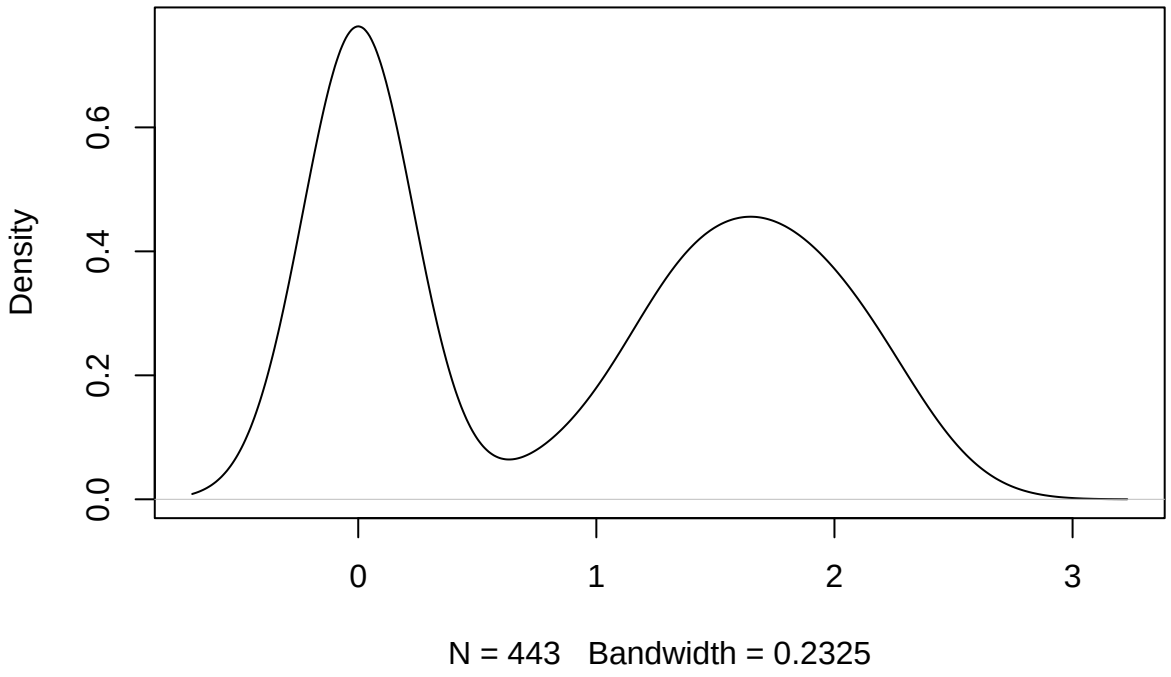
```
library(modeldata)
library(recipes)
mod.train <- recipe(murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian
                    + PctPopUnderPov + PctUnemployed, train_df)
```

Below shows how the data is transformed. As you can see it uses the test and training data set to figure out how to best scale the data in order to get better predictive results and normalize the outcome variable. You can tell that when you look at the murder per population rate variable in the before plot, that it is very much skewed. However after the yeo-johnson transformation you can see that the murder per population rate variable is far less skewed and beginning to look more normal.

```r
yj_transform <- step_YeoJohnson(mod.train,  all_numeric())
yj_estimates <- prep(yj_transform, training = train_df)
yj_te <- bake(yj_estimates, test_df)
plot(density(test_df$murdPerPop), main = "before")
```




```r
plot(density(yj_te$murdPerPop), main = "after")
```



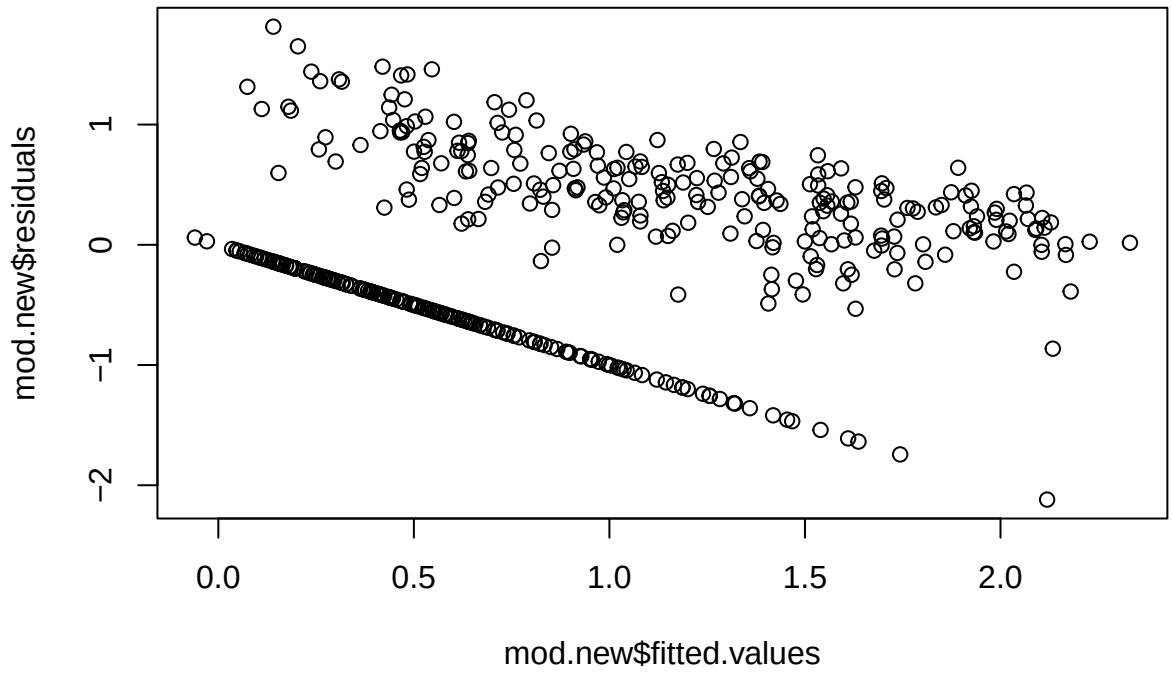
The new step AIC model with the newly transformed data is given below. When the Breusch-Pagan test is done on the model you can see that the p-value significantly increases which means that the model assumption that the errors having constant variance is upheld!

```
mod.new <- lm(murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, yj_te)
bptest(mod.new)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mod.new
## BP = 6.6873, df = 6, p-value = 0.3507
```

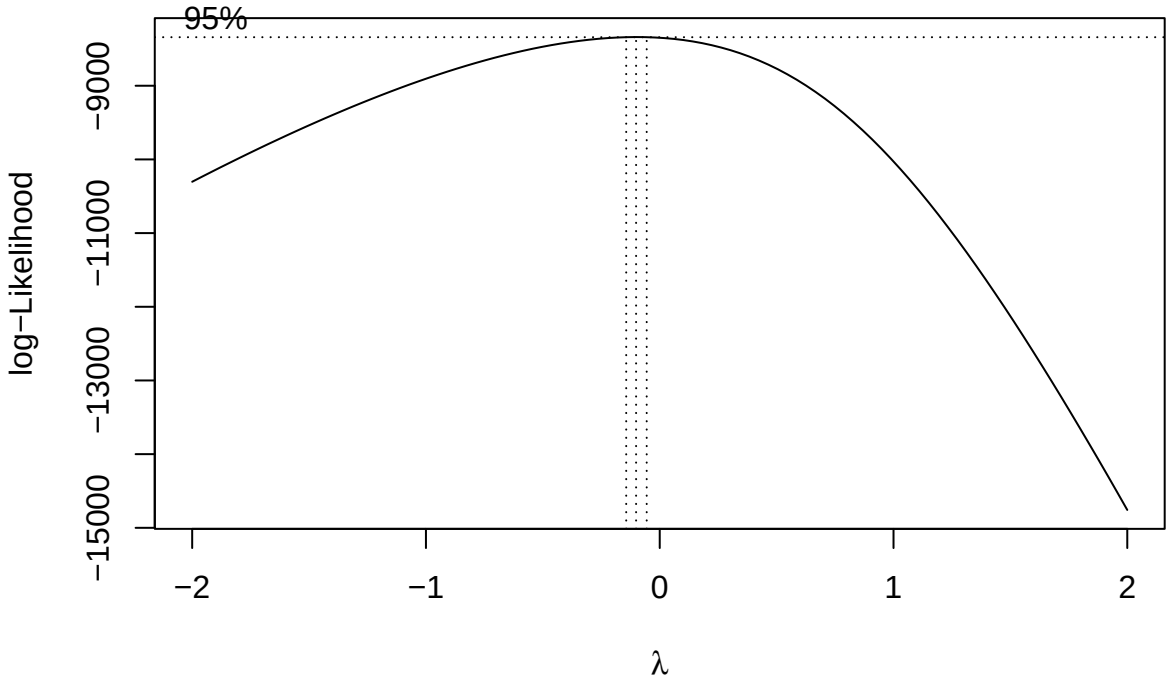
You can also see from the diagnostic plot below that the points are much more scattered and less cluttered now when compared to the original data in step 4 but it could still use some more work. Since this is the case a boxcox transformation was done to see if it could improve the model any better.

```
plot(mod.new$fitted.values, mod.new$residuals)
```



As seen below a box-cox transformation was formed. The box-cox transformation transforms our data so that it closely resembles a normal distribution. Unfortunately it can't handle the response variable being negative unlike the yeo-johnson method. This is why I had to put a plus one after the response variable in the model. The Box-cox method finds an optimal transformation index, denoted by lambda, and it is used when we need to transform the response variable to improve model fit and correct violation of model assumptions. As you can see from the plot below, 0 is not in the 95% confidence interval for lambda. This means that when transforming the response variable in the model we put the response variable to the lambda power.

```
mod.log <- lm(murdPerPop+1 ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, data_mod)
bc <- boxcox(mod.log, plotit=T)
```



```
lambda <- bc$x[which.max(bc$y)]
```

Below is the newly created model using the transformed response variable. Then another Breusch-Pagan test was done on this model. As you can see this model has a much lower p-value than the model using the yeo-johnson transformed data. In fact this model still fails the Breusch-Pagan test and the model assumption that the errors having constant variance is still not upheld.

```
mod.box <- lm((murdPerPop+1)^lambda ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, data_mod)
bptest(mod.box)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  mod.box
## BP = 51.254, df = 6, p-value = 2.633e-09
```

Since the model with the yeo-johnson transformed data was able to uphold the model assumption that the errors having constant variance, I decided to use this model going forward.

Step 7:

In step 7, I report inferences as well as make predictions using my final model.In the final model below you can see that only three variables are statistically significant when alpha equals 0.5. Those variables are household size, race percentage white, and percentage of population under poverty. The variables that increase the murder per population rate as they increase are race percentage Black, percentage of population under poverty, and percentage of population unemployed. The variables that decrease the murder per population rate as they increase are household size, race percentage White, and race percentage Asian.

```
summary(mod.new)$coefficients[,c(1,4)]
```

```
##                    Estimate     Pr(>|t|)
## (Intercept)     1.325755e+01 3.299018e-03
## householdsize  -3.186320e+01 7.554359e-03
## racePctBlack    1.166645e-01 1.474868e-01
## racePctWhite   -9.101138e-08 9.740461e-11
## racePctAsian   -1.478051e-01 3.137550e-01
## PctPopUnderPov  1.529063e-01 2.049716e-02
## PctUnemployed   3.333833e-02 8.822394e-01
```

Below are the final multiple and adjusted r-squared values. The final multiple r-squared ended up being 0.4428577, and the final adjusted r-squared ended up being 0.4351906. I'm pleased with these results because that means that around 43-44% of the variability observed in the murder per population rate is explained by the regression model.

```
summary(mod.new)$r.squared
```

```
## [1] 0.4428577
```

```
summary(mod.new)$adj.r.squared
```

```
## [1] 0.4351906
```

I then wanted to look at a 95% confidence interval for the slope of household size, given that I felt that was the most interesting and one of the most important variables of the model since it's statistically significant and has a larger slope than the rest. However, before I interpret the interval I had to re-scale the numbers back to the original scale in order to be able to actually interpret the slope. This can be seen below as well.

```
tidy(yj_estimates, number = 1)
```

```
## # A tibble: 7 x 3
##   terms            value id
##   <chr>            <dbl> <chr>
## 1 householdsize  -2.58   YeoJohnson_iwNmB
## 2 racePctBlack   -0.300  YeoJohnson_iwNmB
## 3 racePctWhite    3.89   YeoJohnson_iwNmB
## 4 racePctAsian   -0.803  YeoJohnson_iwNmB
## 5 PctPopUnderPov  0.0729 YeoJohnson_iwNmB
## 6 PctUnemployed  -0.230  YeoJohnson_iwNmB
## 7 murdPerPop     -0.263  YeoJohnson_iwNmB
```

As seen below we are 95% confident that the slope of household size is be between -2.352696 and -1.239684.

```
library(VGAM)
confint(mod.new,'householdsize',level = .95)
```

```
##                    2.5 %    97.5 %
## householdsize -55.19678 -8.529619
```

```
yeo.johnson(-55.19678,-2.57579545, inverse = TRUE)
```

```
## [1] -2.352696
```

```
yeo.johnson(-8.529619,-2.57579545, inverse = TRUE)
```

```
## [1] -1.239684
```

I then wanted to figure out what the murder per population rate would be if I used the median of all the variables. In order to do this I made a 95% confidence interval of what the murder per population rate would be when using the median of all the predictors. Below you can see that the 95% confidence interval for the prediction of the mean of the murder per population rate using the medians of the predictor variables should lie in between 1.307664 and 1.737758

```
x <- model.matrix(mod.new)
x0 <- apply(x,2,median)
y0 <- sum(x0*coef(mod.new))
predict(mod.new,new=data.frame(t(x0)),interval="confidence",
        level = .95)
```

```
##         fit       lwr      upr
## 1 0.8176271 0.7505812 0.884673
```

```
yeo.johnson(0.7505812,-0.26327834, inverse = TRUE)
```

```
## [1] 1.307664
```

```
yeo.johnson(0.884673,-0.26327834, inverse = TRUE)
```

```
## [1] 1.737758
```

Lastly, I wanted to do the same thing as I did above however, instead of using a 95% confidence interval, I wanted to use a 95% prediction interval. As seen below, the 95% confidence interval for the prediction of the true value of the murder per population rate using the median of the predictors lies between -0.380429 and 20.75205

```
predict(mod.new,new=data.frame(t(x0)),interval="prediction",
        level = .95)
```

```
##         fit        lwr     upr
```

```
## 1 0.8176271 -0.4747061 2.10996
```

```
yeo.johnson(-0.4747061,-0.26327834, inverse = TRUE)
```

```
## [1] -0.380429
```

```
yeo.johnson(2.10996,-0.26327834, inverse = TRUE)
```

```
## [1] 20.75205
```

Conclusion:

In conclusion you can see that there are many factors that impact the murder per population rate. In our final model the only statistically significant factors were household size, race percentage white, and percentage of population under poverty. Household size and race percentage both had negative slopes which means that as the household size and the percentage of white people in an area increases, the murder per population rate should decrease. While the percentage of population under poverty had a positive slope, which means that as the percentage of population under poverty increases in an area, the murder per population rate should increase as well. This means that from this analysis we can say that in areas where there are majority white people and the house hold sizes are usually on the bigger side, there should be less murders than normal. While if you live in an area where the percentage of the population under poverty is high, then you can expect there to be more murders than normal. This information can help communities that deal with a lot of murders to see in what ways they can improve by seeing what increase and decreases the murder rate. This also helps people that are trying to avoid being murdered by avoiding places where there is a high percentage of the population under poverty.

Appendix:

```
###Step 1

library(tidyverse)
data <- read_delim("CommViolPredUnnormalizedData.txt",na = "?", col_names = FALSE,
                   show_col_types = FALSE)
column_names <- c("communityName","state","countyCode","communityCode","fold","population"
                  ,"householdsize","racePctBlack","racePctWhite","racePctAsian","racePctHisp"
                  ,"medIncome","PctPopUnderPov","PctNotHSGrad","PctUnemployed","murdPerPop")
data_mod <- data[,c(1:11,18,34,36,38,131)]
colnames(data_mod) <- column_names

pairs(data_mod[,6:16])

med_inc_plot <- ggplot(data_mod, aes(medIncome,murdPerPop)) + geom_point() + geom_smooth(method=lm)
suppressMessages(print(med_inc_plot))

hs_grad_plot <- ggplot(data_mod, aes(PctNotHSGrad,murdPerPop)) + geom_point() + geom_smooth(method='lm'
suppressMessages(print(hs_grad_plot))

pov_plot <- ggplot(data_mod, aes(PctPopUnderPov,murdPerPop)) + geom_point() + geom_smooth(method='lm')
suppressMessages(print(pov_plot))

unemp_plot <- ggplot(data_mod, aes(PctUnemployed,murdPerPop)) + geom_point() + geom_smooth(method='lm')
suppressMessages(print(unemp_plot))

###Step 2

mod <- lm(murdPerPop~householdsize+racePctBlack+racePctWhite+
```

```
            racePctAsian+racePctHisp+medIncome+PctPopUnderPov+
            PctNotHSGrad+PctUnemployed,data_mod)
summary(mod)

###Step 3

library(rms)
ols.mod <- ols(murdPerPop~householdsize+racePctBlack+racePctWhite+
            racePctAsian+racePctHisp+medIncome+PctPopUnderPov+
            PctNotHSGrad+PctUnemployed,data_mod)
fastbw(ols.mod, rule = "p", sls = 0.05)

mod.fastbw <- lm(murdPerPop~householdsize+racePctBlack+racePctWhite+
                   racePctAsian+PctUnemployed, data_mod)
summary(mod.fastbw)

library(MASS)
aic_result <- stepAIC(mod)
aic_result$anova

mod.aic <- lm(murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, data_mod)
summary(mod.aic)

###Step 4
plot(mod.aic$fitted.values,mod.aic$residuals)

qqnorm(residuals(mod.aic))
qqline(mod.aic$residuals)

n <- length(residuals(mod.aic))
plot(tail(residuals(mod.aic),n-1) ~ head(residuals(mod.aic),n-1),
     xlab=expression(hat(epsilon)[i]),
     ylab=expression(hat(epsilon)[i+1]))
abline(h=0,v=0,col=grey(0.75))

###Step 5

n <- dim(model.matrix(mod.aic))[1]
p <- dim(model.matrix(mod.aic))[2]
num_df <- p
den_df <- n - p
Fthresh <- qf(0.5, num_df, den_df)
cook_out <- which(cooks.distance(mod.aic) > Fthresh)
length(cook_out)

stand_out <- which(rstandard(mod.aic) > 3)
length(unique(stand_out))


h_i <- hatvalues(mod.aic)
hat_thresh <- (2*p)/n
hat_out <- which(h_i > hat_thresh)
```

```r
length(unique(hat_out))

###Step 6

library(lmtest)
bptest(mod.aic)

n <- nrow(data_mod)
set.seed(1842)
test_index <- sample.int(n,size=round(0.2*n))
train_df <- data_mod[-test_index,]
test_df <- data_mod[test_index,]

library(modeldata)
library(recipes)
mod.train <- recipe(murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian
                    + PctPopUnderPov + PctUnemployed, train_df)

yj_transform <- step_YeoJohnson(mod.train,  all_numeric())
yj_estimates <- prep(yj_transform, training = train_df)
yj_te <- bake(yj_estimates, test_df)
plot(density(test_df$murdPerPop), main = "before")

plot(density(yj_te$murdPerPop), main = "after")

mod.new <- lm(murdPerPop ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, yj_te)
bptest(mod.new)

plot(mod.new$fitted.values, mod.new$residuals)

mod.log <- lm(murdPerPop+1 ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, data_mod)
bc <- boxcox(mod.log, plotit=T)
lambda <- bc$x[which.max(bc$y)]

mod.box <- lm((murdPerPop+1)^lambda ~ householdsize + racePctBlack + racePctWhite + racePctAsian
              + PctPopUnderPov + PctUnemployed, data_mod)
bptest(mod.box)

###Step 7

summary(mod.new)$coefficients[,c(1,4)]

summary(mod.new)$r.squared
summary(mod.new)$adj.r.squared

tidy(yj_estimates, number = 1)

library(VGAM)
confint(mod.new,'householdsize',level = .95)
yeo.johnson(-55.19678,-2.57579545, inverse = TRUE)
yeo.johnson(-8.529619,-2.57579545, inverse = TRUE)
```

```r
x <- model.matrix(mod.new)
x0 <- apply(x,2,median)
y0 <- sum(x0*coef(mod.new))
predict(mod.new,new=data.frame(t(x0)),interval="confidence",
        level = .95)
yeo.johnson(0.7505812,-0.26327834, inverse = TRUE)
yeo.johnson(0.884673,-0.26327834, inverse = TRUE)

predict(mod.new,new=data.frame(t(x0)),interval="prediction",
        level = .95)

yeo.johnson(-0.4747061,-0.26327834, inverse = TRUE)
yeo.johnson(2.10996,-0.26327834, inverse = TRUE)
```