Coding Project Part 2

Making Use of the 'Diamond Prices' Dataset

Jake Merry

12.06.24

Contents

Introduction

For Part 1 of the Diamond Data Set project, we took a random sample of 500 observations from our data set, we described each variable in the data set, provided summary statistics, and made note of an discrepancies that could potentially influence our analysis.

Now, for Part 2, we aim to address any issues that may arise in our sample from Part 1 due to multicollinearity and to find the best fitting model for said sample. To do this, we will run model diagnostics is order to ensure that residuals are homoscedastic and normally distributed, and check for any influential points within our sample.

Following, we will make use of the packages `readr` to access our dataset, `car` for use of the `vif` function to check for multicollinearity, and `MASS` for the `boxcox` function to help us find a proper transformation for our response variable.

```
knitr::opts_chunk$set(echo = TRUE, fig.height=4)

library(readr)
library(car)
library(MASS)

diamondData <- read_csv("cd2_dataset.csv")
```

To maintain consistency, we parse the same observations from the data set that we used in Part 1.

```
set.seed(5)

diamondSample <- diamondData[
  sample(1:nrow(diamondData), 500),
```

```
  c("carat", "color", "clarity", "depth", "table", "price")
]
diamondSample
```

```
## # A tibble: 500 x 6
##    carat color clarity depth table price
##    <dbl> <chr> <chr>   <dbl> <dbl> <dbl>
##  1  0.53 F     VS1      61.7    56  1630
##  2  1.46 H     SI2      61.4    59  7604
##  3  0.56 H     VVS2     59.8    57  1723
##  4  0.7  G     VS2      61.8    54  2593
##  5  1.13 I     SI2      59.9    57  4195
##  6  0.32 F     VVS2     62      55   786
##  7  0.43 F     SI1      63.6    53   948
##  8  1.29 J     VS1      61.7    59  5463
##  9  0.71 I     VS2      64      59  1840
## 10  1.09 H     VS1      60.2    61  5951
## # i 490 more rows
```

Recall from Part 1 that our sample is dealing specifically with the variables `carat`, `color`, `clarity`, `depth`, `table`, and `price`. Here, our dependent variable is `price`; it represents that value we aim to estimate.

Fitting a Multiple Regression Model

To create a model for the data set, we will use the `lm` function that will create a multiple linear regression model for the data. Note that our sample includes categorical variables, for which the `lm` function will automatically add dummy variables for us.

```
mlr <- lm(formula=price~., data=diamondSample)
summary(mlr)
```

```
##
## Call:
## lm(formula = price ~ ., data = diamondSample)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2517.6  -626.3  -121.3   447.5  5014.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5918.99    2938.89   2.014 0.044561 *
## carat        8855.67     114.81  77.130  < 2e-16 ***
## colorE       -345.67     176.86  -1.954 0.051225 .
## colorF       -565.17     174.74  -3.234 0.001303 **
## colorG       -528.30     171.93  -3.073 0.002241 **
## colorH      -1004.52     179.26  -5.604 3.53e-08 ***
## colorI      -1490.16     199.64  -7.464 3.94e-13 ***
## colorJ      -2289.90     249.16  -9.190  < 2e-16 ***
## clarityIF    3905.09     619.40   6.305 6.52e-10 ***
## claritySI1   2462.28     581.18   4.237 2.72e-05 ***
```

```
## claritySI2   1714.38     582.24   2.944 0.003391 **
## clarityVS1   3529.49     586.50   6.018 3.50e-09 ***
## clarityVS2   3248.36     582.37   5.578 4.06e-08 ***
## clarityVVS1  3615.90     606.33   5.964 4.77e-09 ***
## clarityVVS2  3865.72     595.92   6.487 2.17e-10 ***
## depth        -121.57      36.29  -3.350 0.000872 ***
## table         -65.66      21.71  -3.024 0.002626 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 988.1 on 483 degrees of freedom
## Multiple R-squared:  0.9365, Adjusted R-squared:  0.9344
## F-statistic: 445.3 on 16 and 483 DF,  p-value: < 2.2e-16
```

Now that we have a base model for comparison, we can use the Akaike Information Criterion method for variable selection to attempt to find a model that better fits our data. We will also use backward selection to attempt to remove any variable from the current model that are inhibiting its ability to accurately predict the response variable.

```
step(mlr, direction="backward")
```

```
## Start:  AIC=6912.45
## price ~ carat + color + clarity + depth + table
##
##           Df  Sum of Sq        RSS    AIC
## <none>                    471534356 6912.4
## - table    1    8928171  480462527 6919.8
## - depth    1   10955480  482489836 6921.9
## - color    6  127498266  599032622 7020.1
## - clarity  7  231465505  702999861 7098.1
## - carat    1 5807810711 6279345067 8205.0

##
## Call:
## lm(formula = price ~ carat + color + clarity + depth + table,
##     data = diamondSample)
##
## Coefficients:
## (Intercept)        carat       colorE       colorF       colorG       colorH
##     5918.99      8855.67      -345.67      -565.17      -528.30     -1004.52
##      colorI       colorJ    clarityIF   claritySI1   claritySI2   clarityVS1
##    -1490.16     -2289.90      3905.09      2462.28      1714.38      3529.49
##  clarityVS2  clarityVVS1  clarityVVS2        depth        table
##     3248.36      3615.90      3865.72      -121.57       -65.66
```

As we can see, AIC backward selection did not remove any variable from our model. This means that the predictors that we currently have in our model form the best model according to this method.

Now, a few notes:

- Notice that the number of dummy variables added for each categorical variable is one less that the number of unique categories that each variable contains.

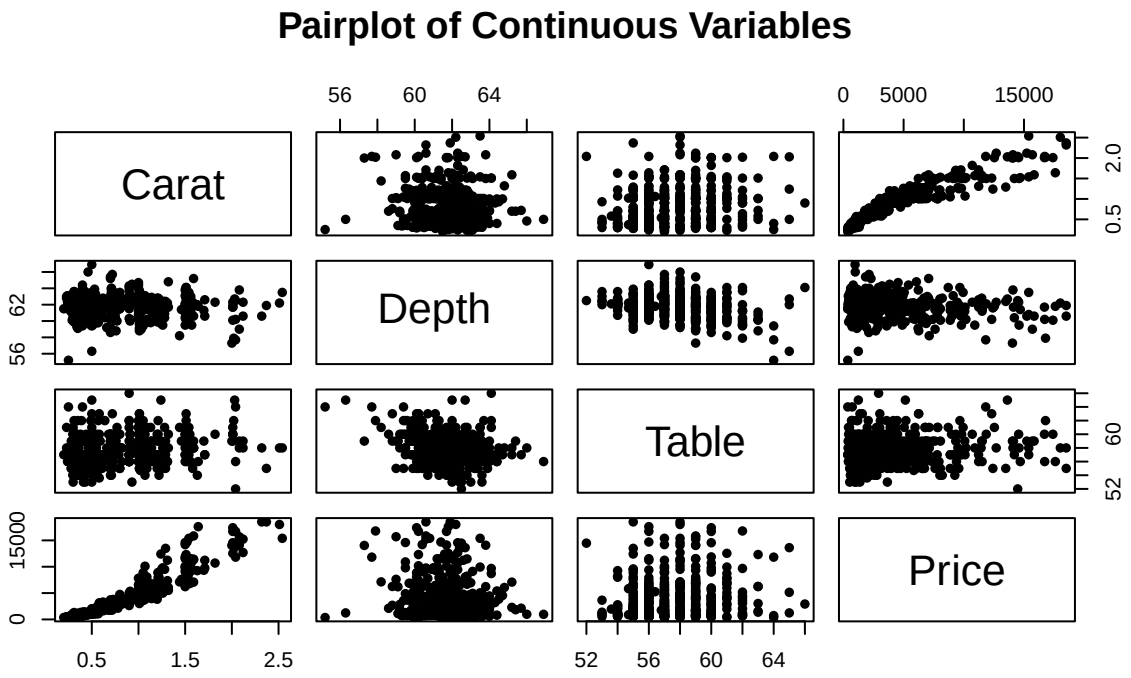
- At this point, it is also worth noting that the Adjusted R^2 value for the model is 0.9344, which is already very close to 1. However, we shall continue to attempt to increase this value to ensure that we find the best model for our data. -Also notice that the standard error of our residuals has an unexpectedly high value of 988.1. We should keep an eye on this value as we attempt to improve our model in an attempt to lower this value.

Addressing Multicollinearity

The next step to ensuring that we find the best model is by analyzing the variable present in our model to ensure that none of them are highly correlated with each other which would cause unreliable estimates of our response.

We can first analyze the correlation of each of the quantitative variables in the model by creating a pair plot and correlation table.

```
pairs(
  diamondSample[c("carat", "depth", "table", "price")],
  pch = 16,
  label = c("Carat", "Depth", "Table", "Price"),
  main = "Pairplot of Continuous Variables"
)
```




```
cor(diamondSample[c("carat", "depth", "table", "price")])
```

```
##             carat       depth      table      price
## carat  1.00000000 -0.08147112  0.2227081  0.9421419
## depth -0.08147112  1.00000000 -0.2815579 -0.1286931
## table  0.22270806 -0.28155791  1.0000000  0.1598044
## price  0.94214192 -0.12869305  0.1598044  1.0000000
```

As we can see, the only variables that are highly correlated are `carat` and `price`, however, since `price` is our response variable, this is to be expected and will provide better predictions rather than hinder our model.

We will as well use the `vif` function to directly analyze any multicollinearity within the model. Any variable with a value greater than 5 should be dropped from the model.

```
vif(mlr)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## carat    1.475817  1        1.214832
## color    1.328838  6        1.023975
## clarity 1.495424  7        1.029161
## depth    1.139411  1        1.067432
## table    1.201328  1        1.096051
```

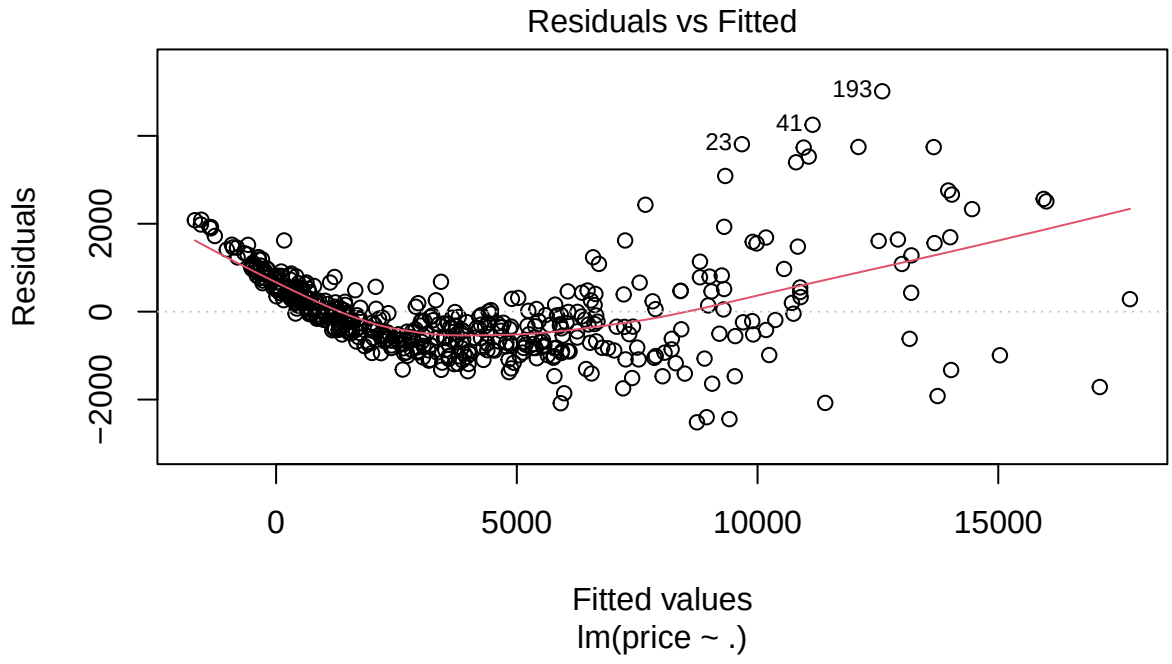
Using the Variance Inflation Factor, we can see again that there is not any significant multicollinearity within our model. Notice, however, that the `vif` function also includes our categorical variables so that we can analyze multicollinearity between all explanatory variables, rather than just the variables that have quantitative values.

Model Diagnostics

There are still a few things that we need to check in our model to ensure that we create the best model for our data. These include checking for heteroscedasticity, linearity, normality, and linearity, as well as ensuring that there are no influential points swaying our model away from the true best-fit line.

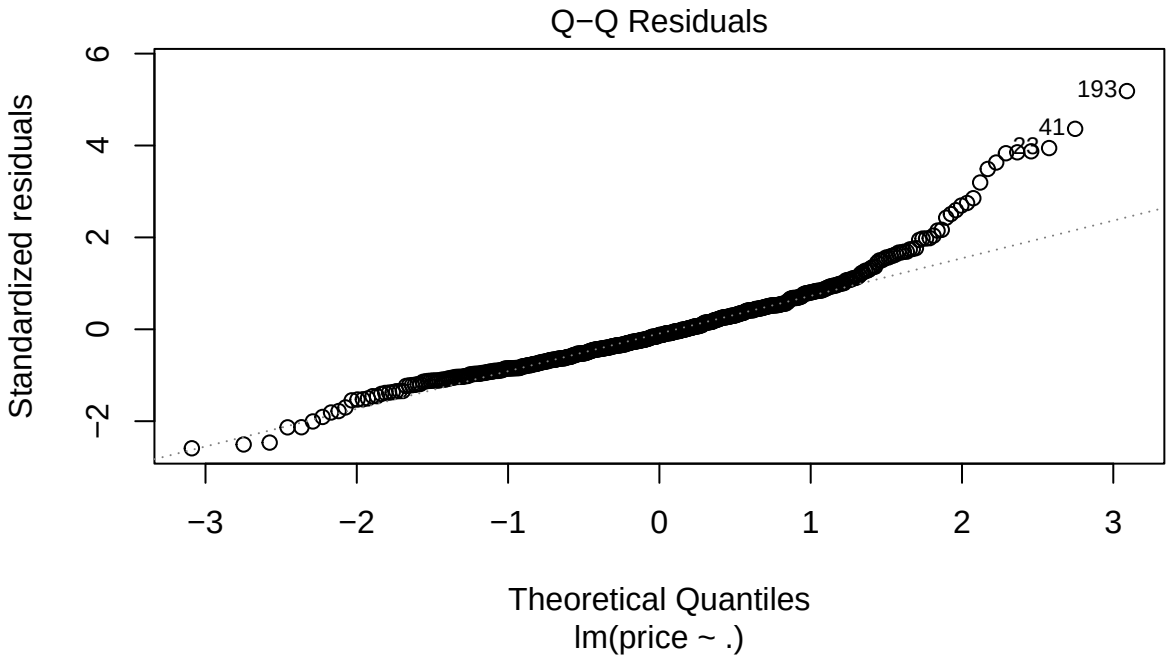
Let's create some plots of our model to make this a bit easier. Each of the following graphs will tell us something different about our model.

```
plot(mlr, 1)
```



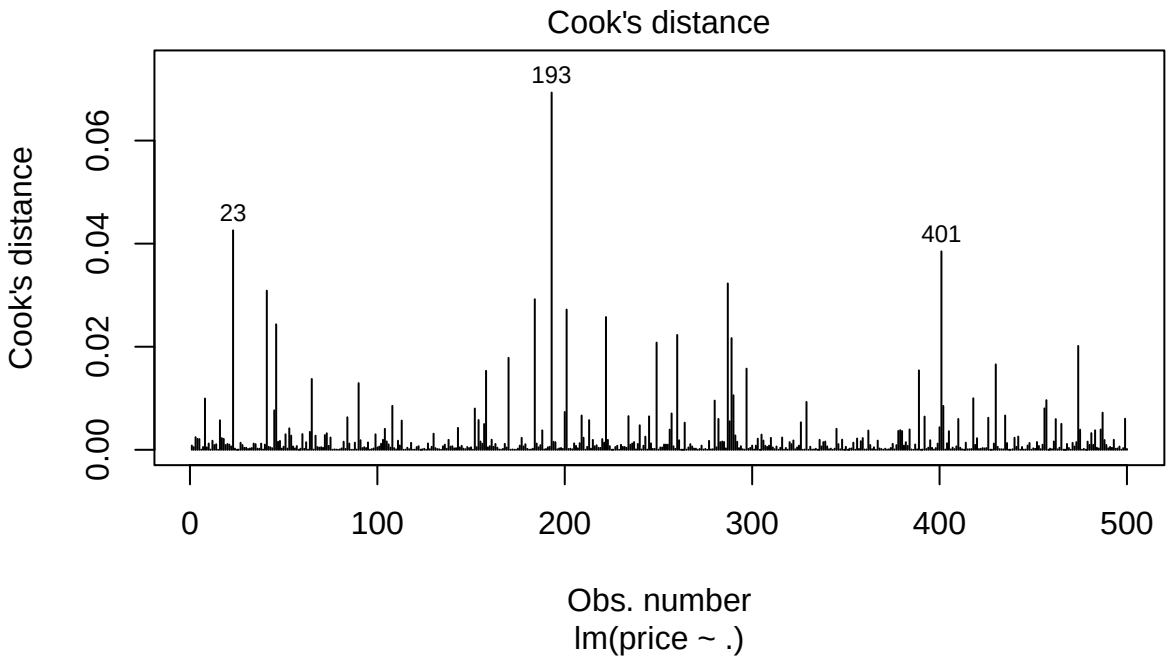
Here, we can see that the variance of our residuals are non-constant, i.e. our model is heteroscedastic. Specifically, our residuals form an outward opening funnel. Ideally, we would be able to find some vertical limits that contain our residual values. This is the first problem with our model.

```
plot(mlr, 2)
```



The QQ plot us shows that our residuals are not exactly normally distributed since there are some observations that clearly vary from the mean response much more than others.

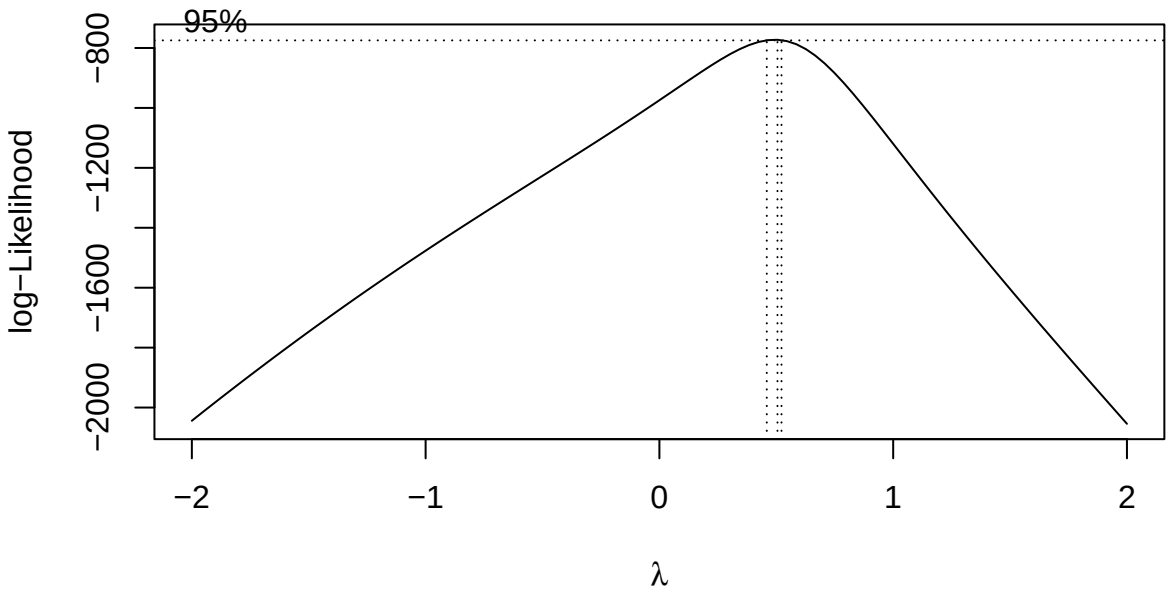
```
plot(mlr, 4)
```



And from this final plot, we see that our model may have some influential points that could be swaying our response prediction away from its true value.

To deal with the problems of heteroscedasticity and non-normality, we can attempt a transformation of one or more of our variables. We will use the Box-Cox method in order to find the transformation that will work best for our model.

```
lambda <- boxcox(mlr, lambda = seq(-2, 2, by = 0.1))$x[
  which.max(boxcox(mlr, lambda = seq(-2, 2, by = 0.1))$y)
]
```




```
lambda
```

```
## [1] 0.5050505
```

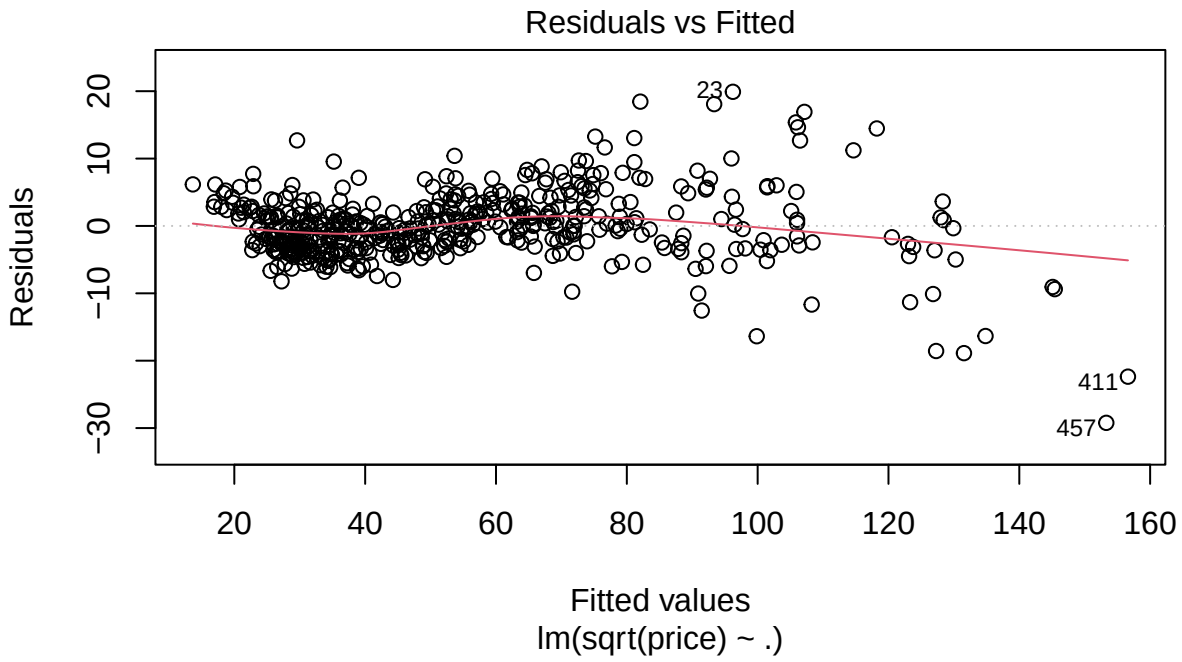
Since the value returned to us is very close to 0.5, we know that the best transformation for our response variable is a square root transformation. Let's apply this now.

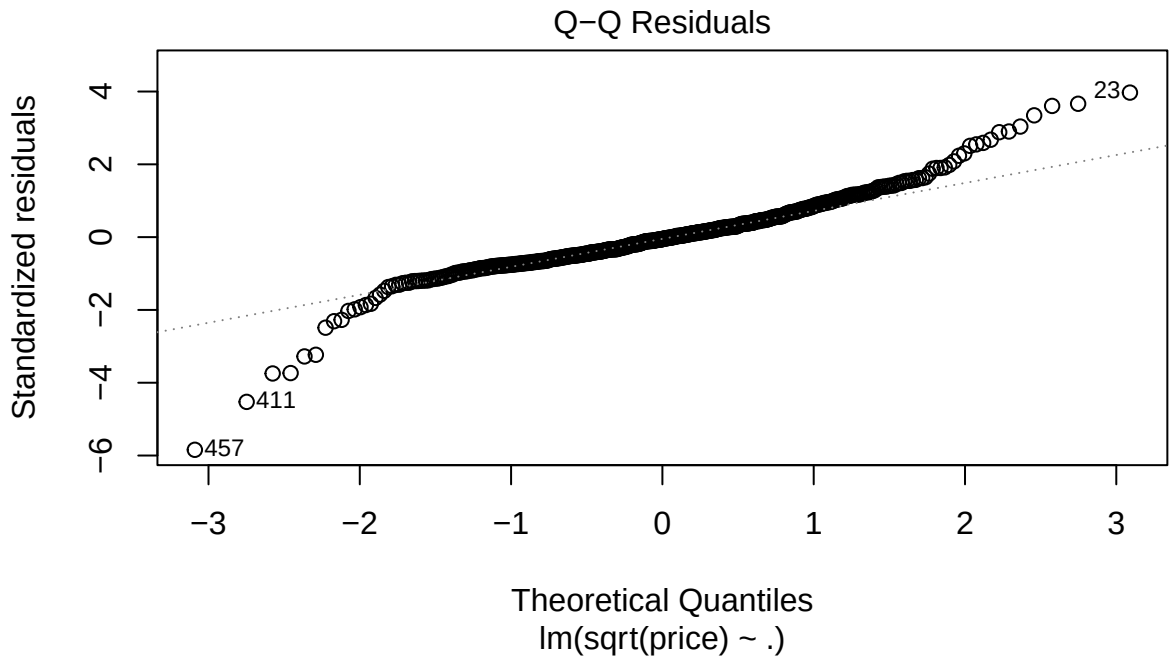
```
mlr.sqrt <- lm(formula=sqrt(price)~., data=diamondSample)
summary(mlr.sqrt)
```

```
##
## Call:
## lm(formula = sqrt(price) ~ ., data = diamondSample)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -29.2240  -2.8581  -0.2698   2.3664  19.9089
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  39.8209    15.2566   2.610 0.009333 **
## carat        65.2059     0.5960 109.399  < 2e-16 ***
## colorE       -2.0463     0.9181  -2.229 0.026294 *
## colorF       -1.5748     0.9071  -1.736 0.083190 .
## colorG       -3.3188     0.8926  -3.718 0.000224 ***
## colorH       -6.7562     0.9306  -7.260 1.56e-12 ***
## colorI      -11.7947     1.0364 -11.380  < 2e-16 ***
## colorJ      -16.1146     1.2935 -12.458  < 2e-16 ***
```

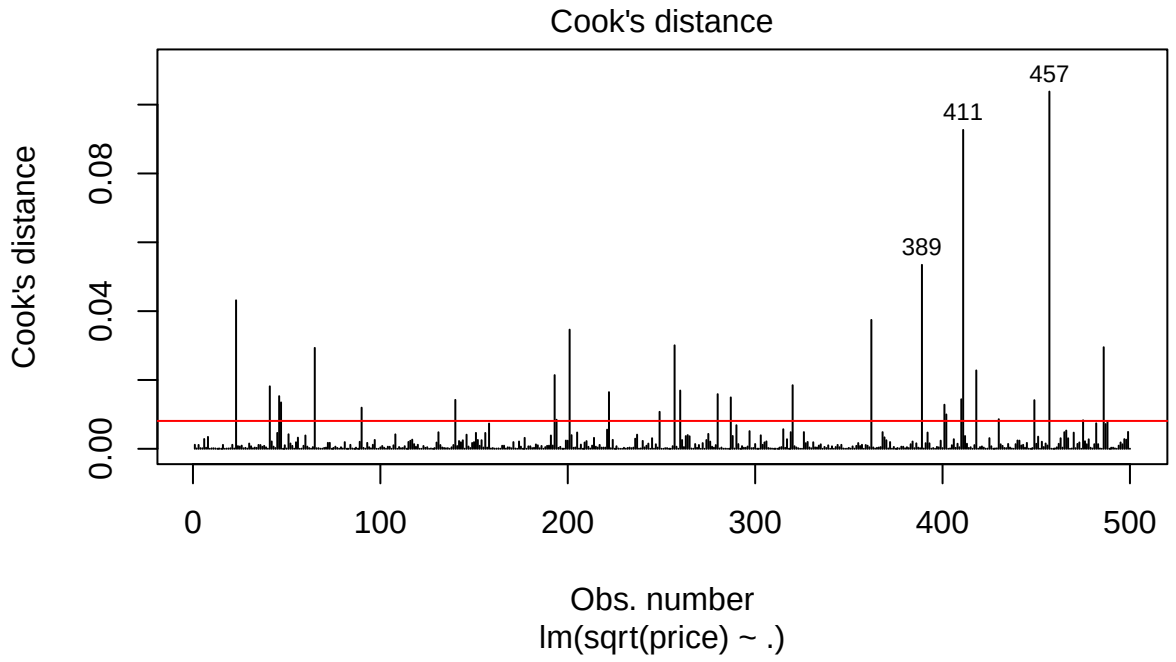
```
## clarityIF    22.5103     3.2155   7.001 8.57e-12 ***
## claritySI1   13.9481     3.0171   4.623 4.86e-06 ***
## claritySI2    8.4291     3.0226   2.789 0.005501 **
## clarityVS1   19.6183     3.0447   6.443 2.83e-10 ***
## clarityVS2   17.3754     3.0233   5.747 1.61e-08 ***
## clarityVVS1  21.1513     3.1476   6.720 5.14e-11 ***
## clarityVVS2  22.1153     3.0936   7.149 3.26e-12 ***
## depth        -0.4945     0.1884  -2.625 0.008941 **
## table        -0.2996     0.1127  -2.659 0.008108 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.129 on 483 degrees of freedom
## Multiple R-squared:  0.9678, Adjusted R-squared:  0.9667
## F-statistic: 906.5 on 16 and 483 DF,  p-value: < 2.2e-16
```

```
plot(mlr.sqrt, c(1, 2, 4))
```






```
n = 500
p = 5
threshold <- 4/(n-p-1)
abline(h=threshold, col="red")
```



We can see that this transformation has significantly improved our model, however, it is still not perfect. Notice that for the plot of Cook's Distance, we have now added a line representing the threshold that we do not want our residuals to exceed. Any point that falls above this line need to be investigated. Since there seems to be a considerable amount of these points, this could be what is preventing us from finding the best-fitting model for our data.

We can get a list of the Cook's Distances for each of our observations and use this to delete any unwanted data.

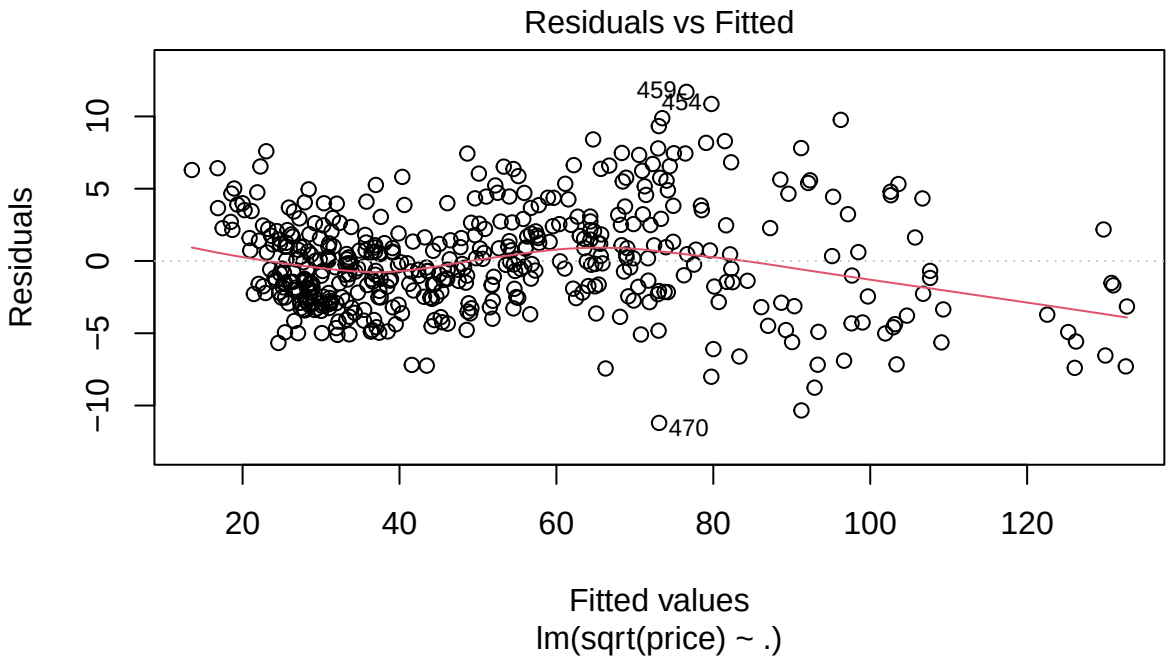
```
cooks <- cooks.distance(mlr.sqrt)
rev(sort(round(cooks, 5)))[1:10]
```

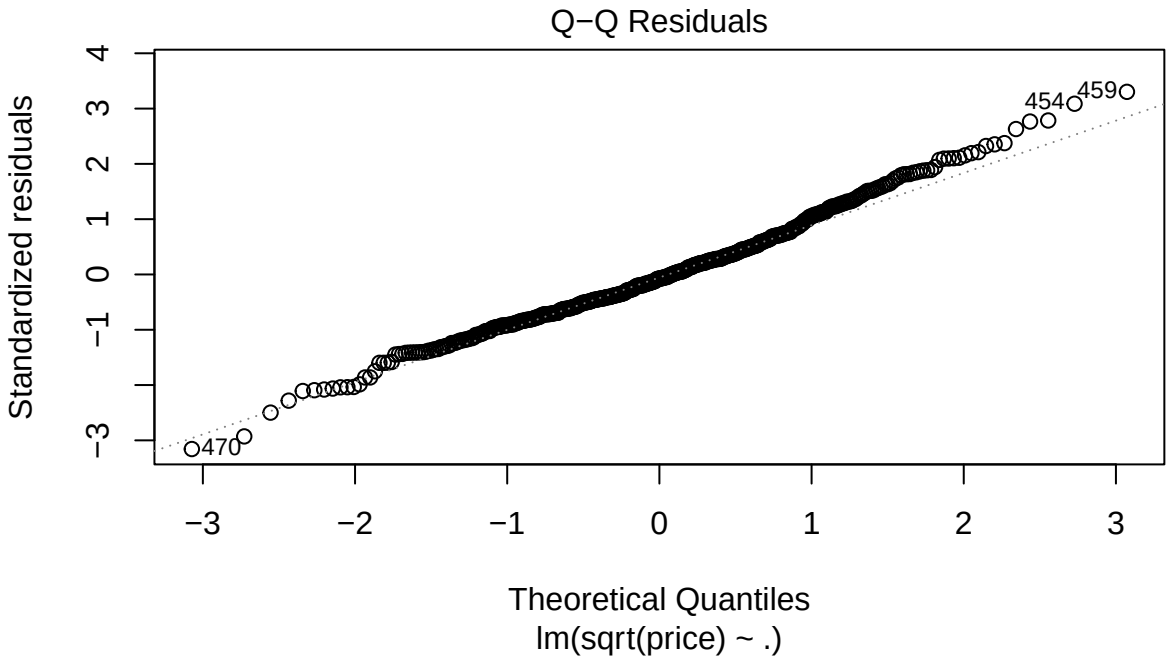
```
##     457     411     389      23     362     201     257     486      65     418
## 0.10379 0.09266 0.05345 0.04314 0.03744 0.03462 0.03007 0.02952 0.02932 0.02277
```

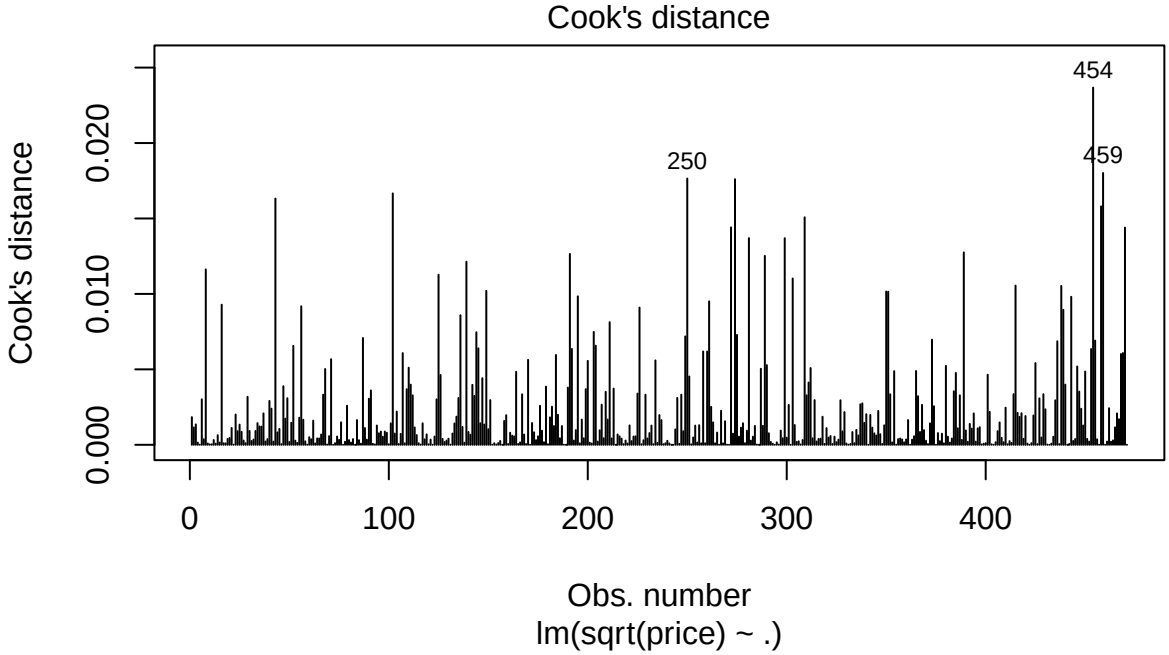
```
diamondSample_clean <- diamondSample[-which(cooks > threshold), ]

mlr.new <- lm(formula=sqrt(price)~., data=diamondSample_clean)
summary(mlr.new)
```

```
##
## Call:
## lm(formula = sqrt(price) ~ ., data = diamondSample_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.2024  -2.4567  -0.2638   2.0639  11.6982
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.73291   11.23428   2.825  0.00494 **
## carat        66.38791    0.49025 135.418  < 2e-16 ***
## colorE       -1.51944    0.65772  -2.310  0.02133 *
## colorF       -0.98998    0.65462  -1.512  0.13115
## colorG       -3.09416    0.64608  -4.789 2.27e-06 ***
## colorH       -5.60847    0.66990  -8.372 7.08e-16 ***
## colorI      -10.14464    0.76162 -13.320  < 2e-16 ***
## colorJ      -15.28029    0.99967 -15.285  < 2e-16 ***
## clarityIF    20.73392    2.69777   7.686 9.49e-14 ***
## claritySI1   12.76267    2.57568   4.955 1.02e-06 ***
## claritySI2    7.23592    2.57997   2.805  0.00525 **
## clarityVS1   18.00834    2.58821   6.958 1.21e-11 ***
## clarityVS2   16.01908    2.57938   6.210 1.19e-09 ***
## clarityVVS1  20.13111    2.64945   7.598 1.73e-13 ***
## clarityVVS2  19.64904    2.62287   7.491 3.59e-13 ***
## depth        -0.45320    0.13951  -3.249  0.00125 **
## table        -0.20561    0.08211  -2.504  0.01263 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.591 on 454 degrees of freedom
## Multiple R-squared:  0.9807, Adjusted R-squared:   0.98
## F-statistic:  1439 on 16 and 454 DF,  p-value: < 2.2e-16
```

```r
plot(mlr.new, c(1, 2, 4))
```







Now that we have cleaned up our sample data a bit, our model looks much better. We can see that the have grouped closer together, indicating a homoscedastic model, and our residuals now follow much closer to the normal distribution that we were looking for.

As well, we can see that our adjusted R^2 value has increased to 0.98, even closer to a perfect fit than when we started, and the standard error of our residuals has significantly decreased from our first model, moving from 988.1, all the way down to 3.591. A clear improvement to our model.

It appears that this model is the best-fitting model for our data that we currently have the tools to find. The model that we found could potentially be improved in the future by running additional diagnostics, but the improvements that we have made are significant and the summary values that have been returned indicate a fairly accurate model.

Final Notes

- Notice that when we run the multiple regression model for our sample, R does a good job at adding dummy variables to incorporate the categorical variables into the model. However, each new variable has a different significance level. So, for each categorical variable, we must analyze the significance of the variable as a whole rather than analyzing the significance of each category individually.
- Although our original regression model ended up being very close to the model that we chose in the end, it was important to continue to check for any issues (multicollinearity, heteroscedasticity, etc.) that may have arisen in the model, inhibiting our ability to predict the response variable.
- The biggest hindrance to our model was the fact that there were a large number of influential points within our sample. This cause our model to overestimate our response variable since some of our observations varied too far from the rest of our data to be of any use. Removing these points allowed us to draw much more accurate conclusions about our response variable.