{STAT 5214G and Data Analytics 1}
{Final Project}

Marshall Tuck

8/12/2020

Note to Tom: The Data Analytics section of this project begins on page 41 within this same file.

Advanced Methods of Regression

Regression

Model Selection

After wrangling our data, first we fit a full model of all regressors across the selected observations. AIC = 2114.789.

```
#Full Model
housing_fit_full<- lm(sales.price~ . , data=housing %>%
                        dplyr::select(-X1, -Last.Chg.Type, -Status, -'MLS.#', -Type, -Address, -Area, -

#Best subsets
ols_step_both_aic(housing_fit_full)
```

```
##
##
##                                              Stepwise Summary
## ----------------------------------------------------------------------------------------------------
## Variable               Method       AIC          RSS                Sum Sq            R-Sq      Adj. 
## ----------------------------------------------------------------------------------------------------
## List.Price            addition    2138.857    8770339576.452    131795160047.311    0.93761      0.9
## tax.assessed.value    addition    2138.670    8582455477.670         1.31983e+11    0.93894      0.9
## remodeled.kitchen     addition    2138.315    8384680427.554    132180819196.209    0.94035      0.9
## BA                    addition    2138.006    8195129230.249    132370370393.514    0.94170      0.9
## architecture          addition    2137.779    7554051832.283    133011447791.479    0.94626      0.9
## ----------------------------------------------------------------------------------------------------
```

```
#Best Fit
housing_fit_best<- lm(sales.price ~ List.Price + tax.assessed.value+remodeled.kitchen+BA+architecture,
                        data=housing)

summary(housing_fit_best)$coefficients
```

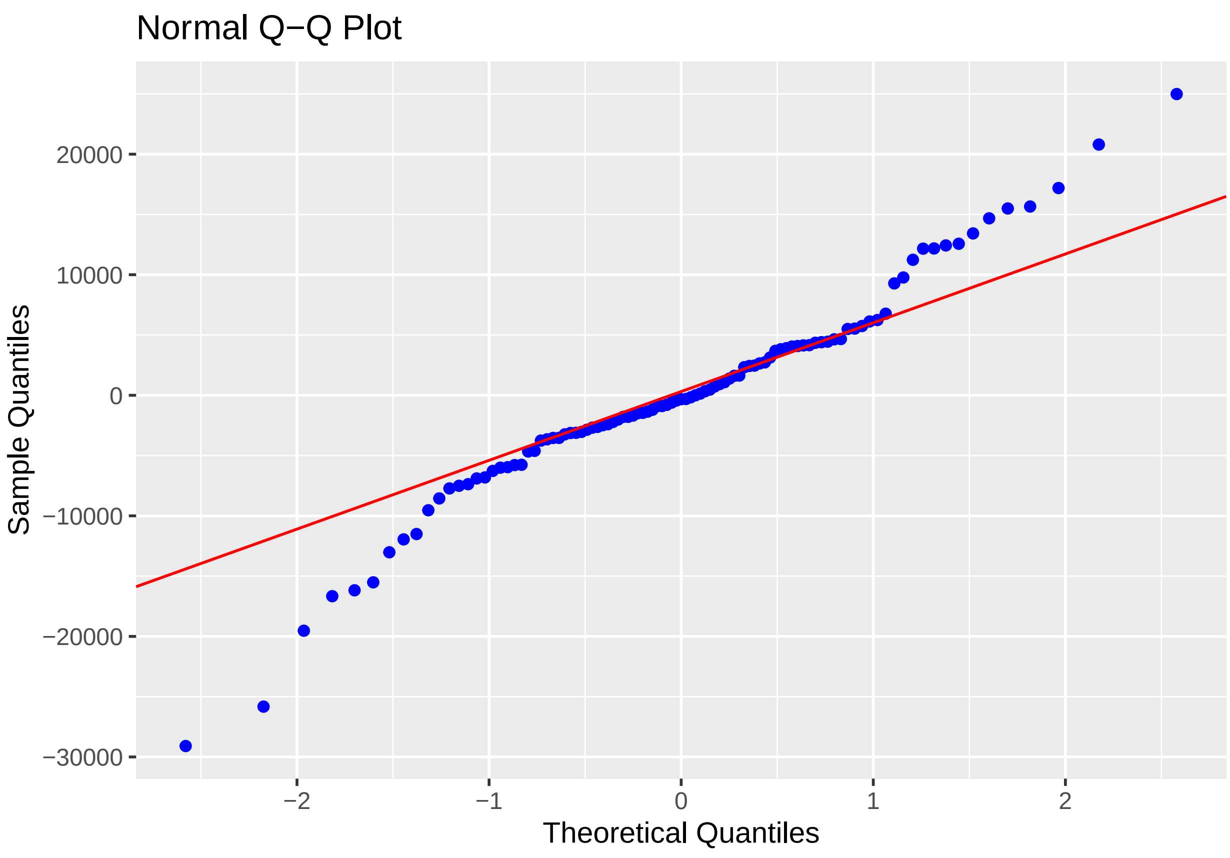
```
##                              Estimate   Std. Error    t value      Pr(>|t|)
## (Intercept)               1.478085e+04 8.250706e+03  1.7914650 7.650684e-02
## List.Price                9.027012e-01 3.772886e-02 23.9260162 2.827060e-41
## tax.assessed.value        8.195410e-02 3.680973e-02  2.2264246 2.842699e-02
## remodeled.kitchenyes      3.583572e+03 1.945078e+03  1.8423794 6.864065e-02
## BA                       -3.301623e+03 2.142963e+03 -1.5406815 1.268270e-01
## architecturecolonial     -1.448688e+03 4.194636e+03 -0.3453669 7.306070e-01
## architectureranch        -5.302695e+03 2.497421e+03 -2.1232681 3.641573e-02
## architecturetrilevel     -3.042207e+03 3.023351e+03 -1.0062368 3.169409e-01
## architecturetwo-story    -1.882333e+04 9.581852e+03 -1.9644771 5.249245e-02
```

Diagnostics

Normality

We see evidence of departure from normality at the extremes.

```
ols_plot_resid_qq(housing_fit_best)
```

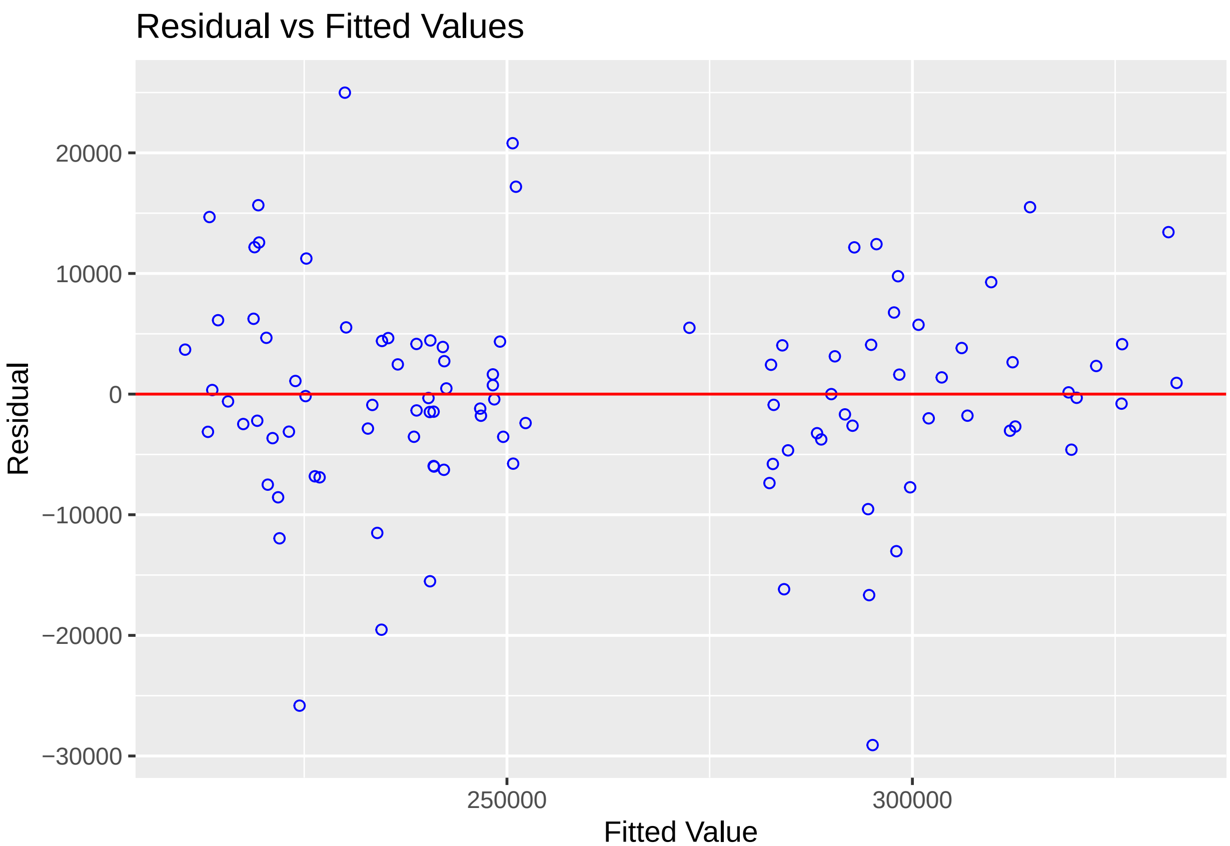



```
#Studentized Residuals plot throws error due to a single Nan value within studres, will be removed in o
#qqPlot(housing_fit_best,ylab="Studentized Residuals",xlab="Theoretical Quantiles")
```

Constant variance

In general we see constant variance with no cause for concern. However there are few observations with fitted values between 250 and 270.

```
#Regular Residuals
ols_plot_resid_fit(housing_fit_best)
```



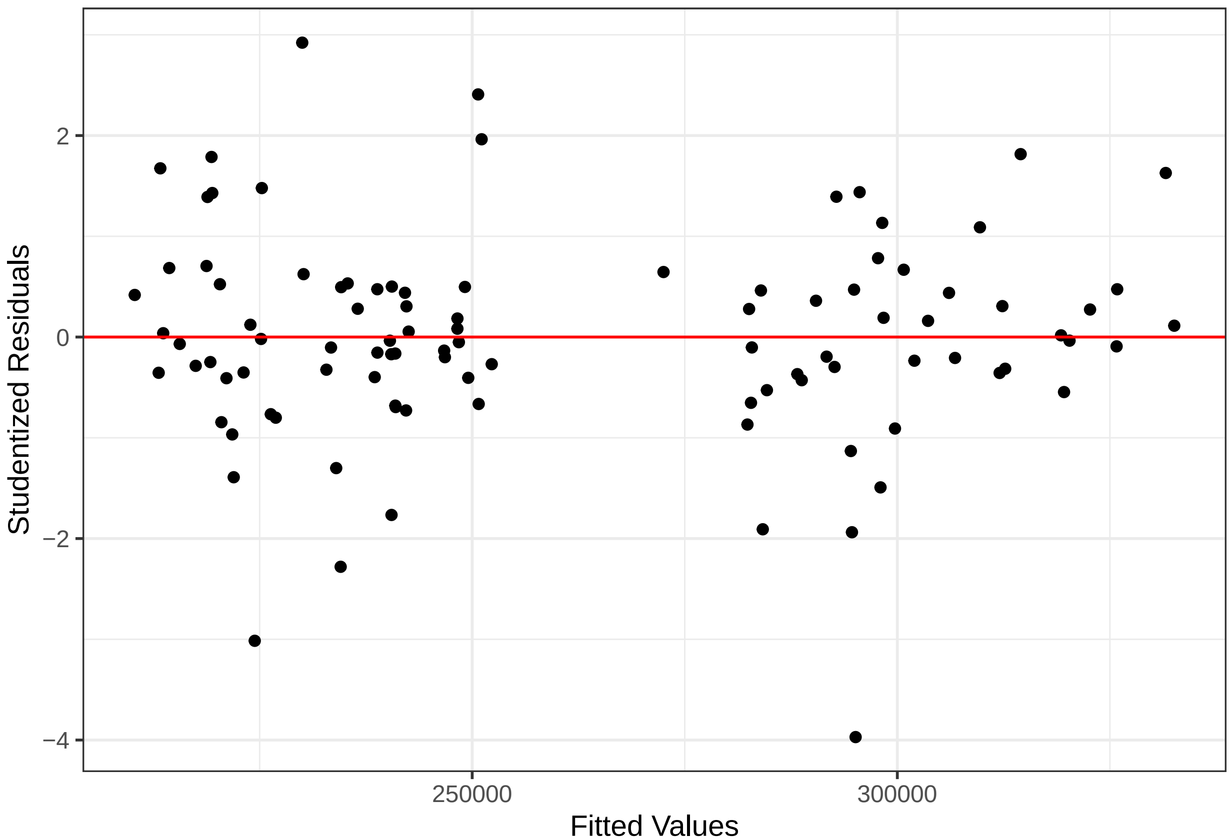

```
#Studentized Residuals
housing$studres<- studres(housing_fit_best)
```

```
## Warning in sqrt((n - p - sr^2)/(n - p - 1)): NaNs produced
```

```
housing$fitted<- housing_fit_best$fitted.values
housing$residuals<- housing_fit_best$residuals

p0<- ggplot(housing, aes(x=fitted, y=studres))+
  geom_point()+
  theme_bw()+
  labs(y="Studentized Residuals", x="Fitted Values")+
  geom_hline(yintercept=0, color="red")

p1<- ggplot(housing, aes(x=fitted, y=studres, color=architecture, shape=architecture))+
  geom_point()+
  theme_bw()+
```

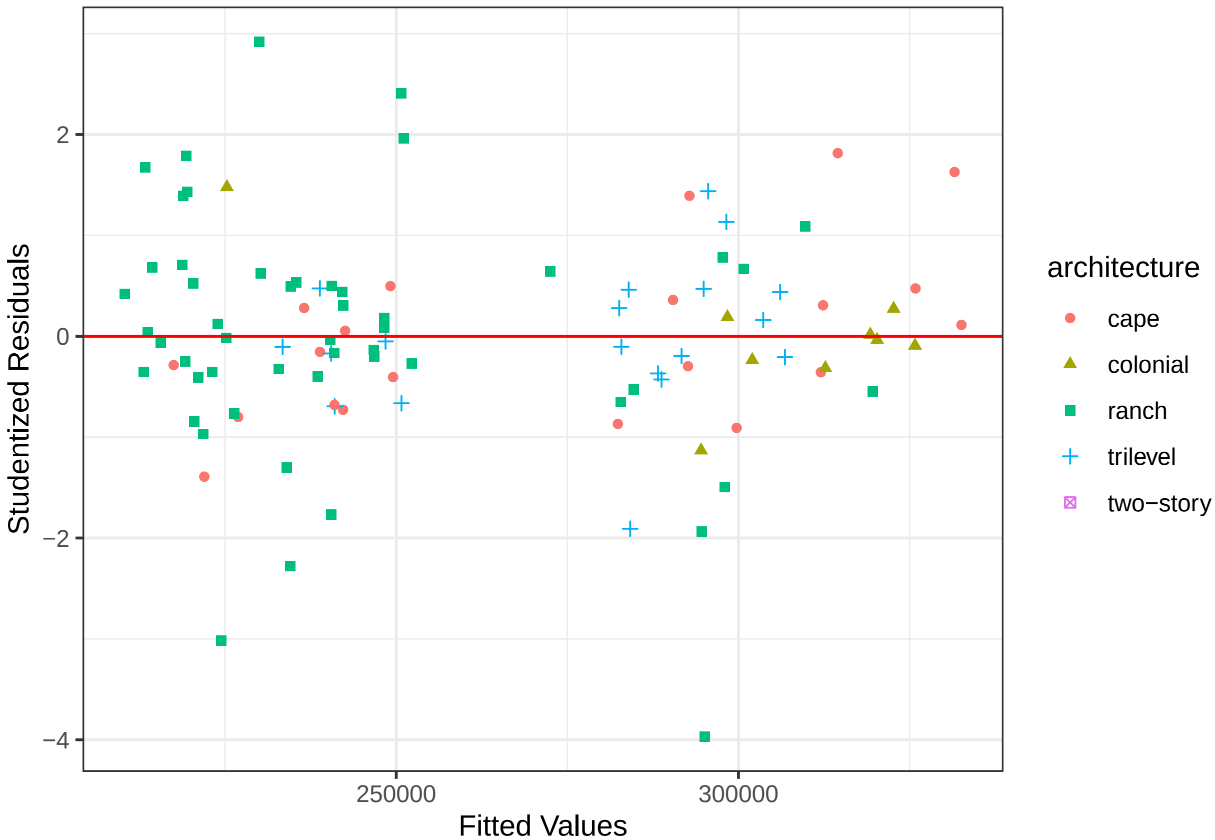
```
  labs(y="Studentized Residuals", x="Fitted Values")+
  geom_hline(yintercept=0, color="red")

p2<- ggplot(housing, aes(x=fitted, y=studres, color=remodeled.kitchen, shape=remodeled.kitchen))+
  geom_point()+
  theme_bw()+
  labs(y="Studentized Residuals", x="Fitted Values")+
  geom_hline(yintercept=0, color="red")


p0
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



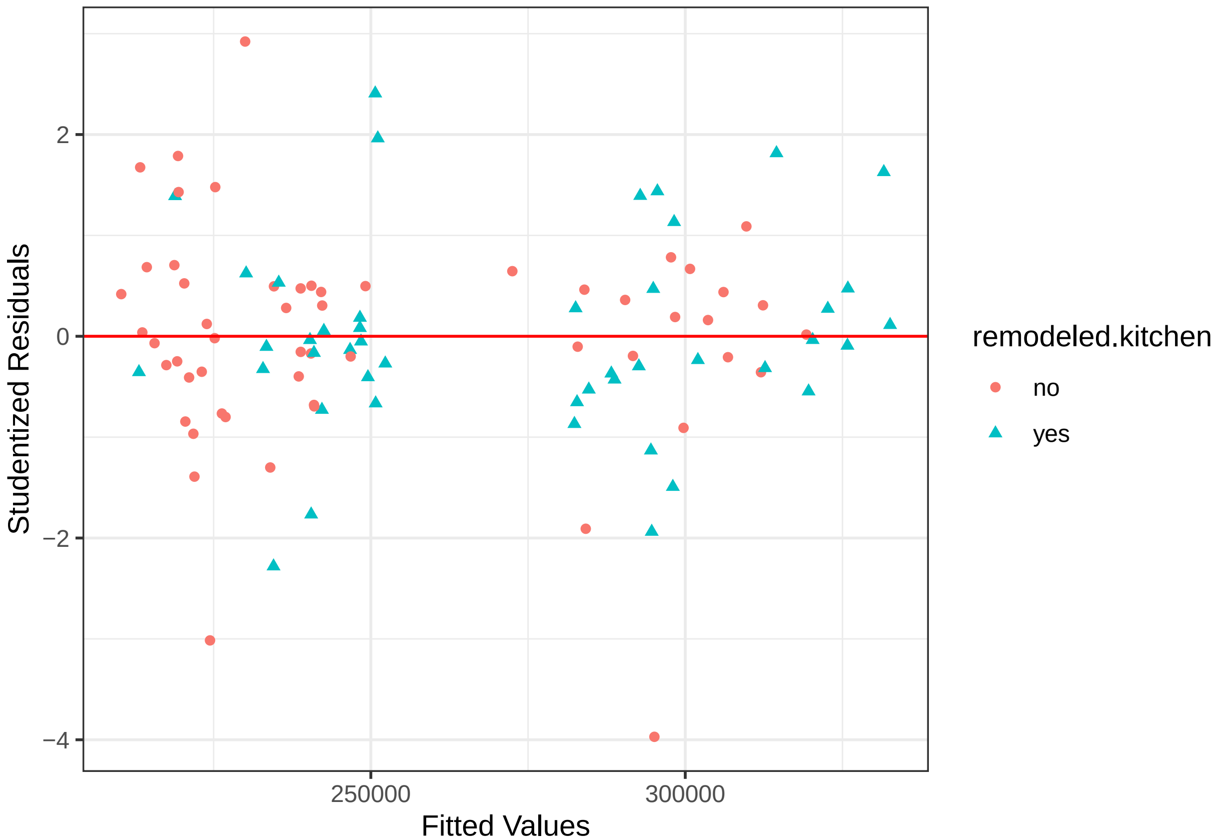

```
p1
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
p2
```

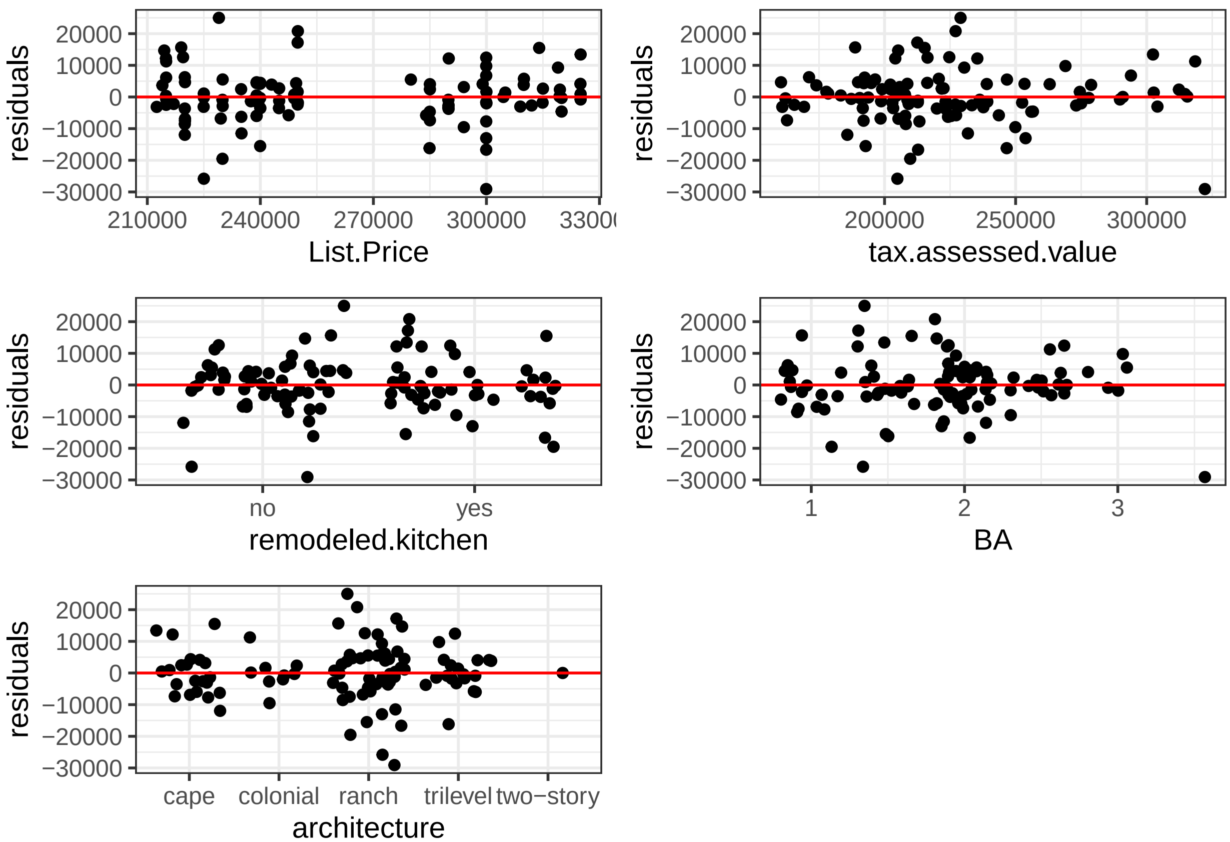
```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Residuals vs. Regresors in Model We see for the most part constant variance per regressor, with some evidence of reverse funneling for tax assessed value which may indicate a transformation is needed.

```r
#Regular Residual Plot
p1<- ggplot(housing, aes(x=List.Price, y=residuals))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p2<- ggplot(housing, aes(x=tax.assessed.value, y=residuals))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p3<- ggplot(housing, aes(x=remodeled.kitchen, y=residuals))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p4<- ggplot(housing, aes(x=BA, y=residuals))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p5<- ggplot(housing, aes(x=architecture, y=residuals))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
```

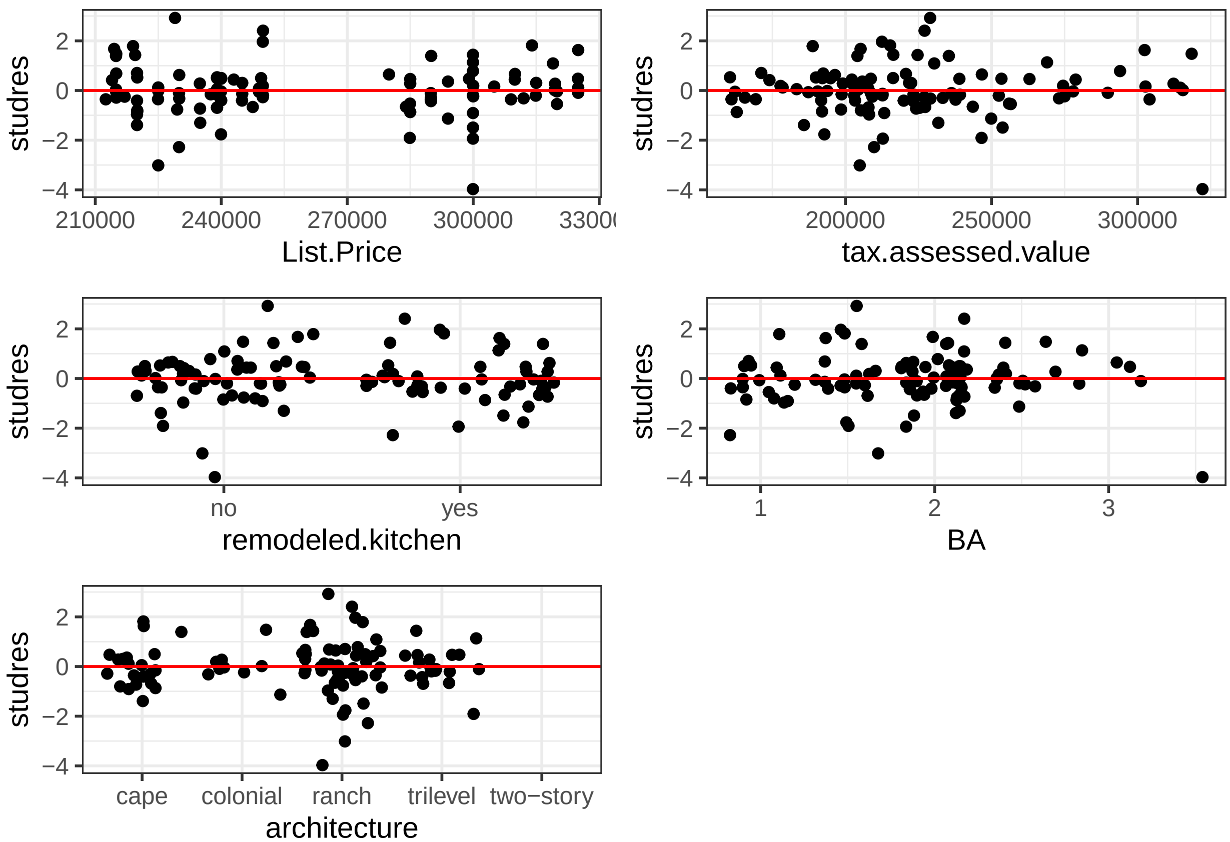
```
  theme_bw()

grid.arrange(p1,p2,p3,p4,p5, nrow=3)
```




```
p1<- ggplot(housing, aes(x=List.Price, y=studres))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p2<- ggplot(housing, aes(x=tax.assessed.value, y=studres))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p3<- ggplot(housing, aes(x=remodeled.kitchen, y=studres))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p4<- ggplot(housing, aes(x=BA, y=studres))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p5<- ggplot(housing, aes(x=architecture, y=studres))+
  geom_jitter()+
```

```r
  geom_hline(yintercept=0, color="red")+
  theme_bw()

grid.arrange(p1,p2,p3,p4,p5, nrow=3)
```

```
## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_point).
```



Multicollinearity

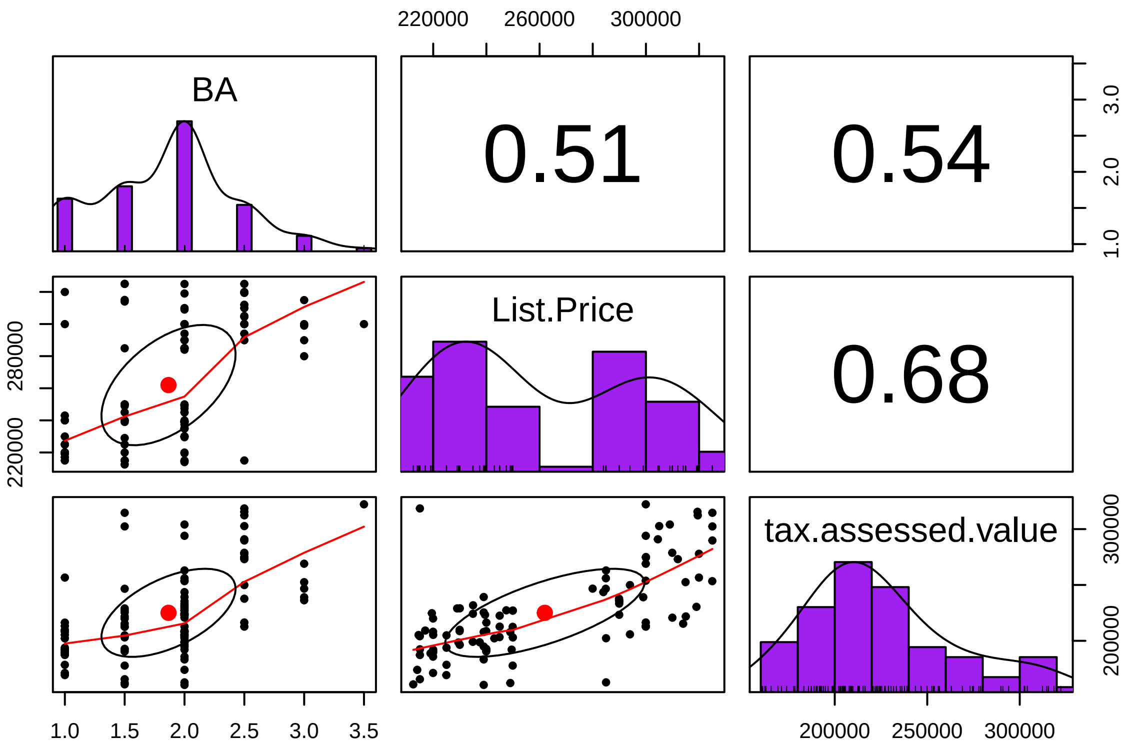
There is no evidence of multicollinearity due to low VIF values and correlation values less than .9 or greater than -.9.

```r
#VIF Values
ols_vif_tol(housing_fit_best)
```

```
##                Variables Tolerance      VIF
## 1             List.Price 0.4105740 2.435614
## 2     tax.assessed.value 0.3892121 2.569293
## 3 remodeled.kitchenyes 0.8664010 1.154200
## 4                     BA 0.5685959 1.758718
```

```
## 5  architecturecolonial 0.5692389 1.756732
## 6     architectureranch 0.5214224 1.917831
## 7  architecturetrilevel 0.5823295 1.717241
## 8 architecturetwo-story 0.9032638 1.107096
```

```r
#Pairs Panels
terms<- names(housing_fit_best$coefficients[-1])
ind3<- which(names(housing)%in%terms)

pairs.panels(housing[ind3],
             method="pearson",
             hist.col = "purple",
             density=TRUE,
             ellipses = TRUE)
```



Transformations Needed: Log Sales Price?

First we test a log transformation on the sales price to see if it will help our error terms appear more normal.

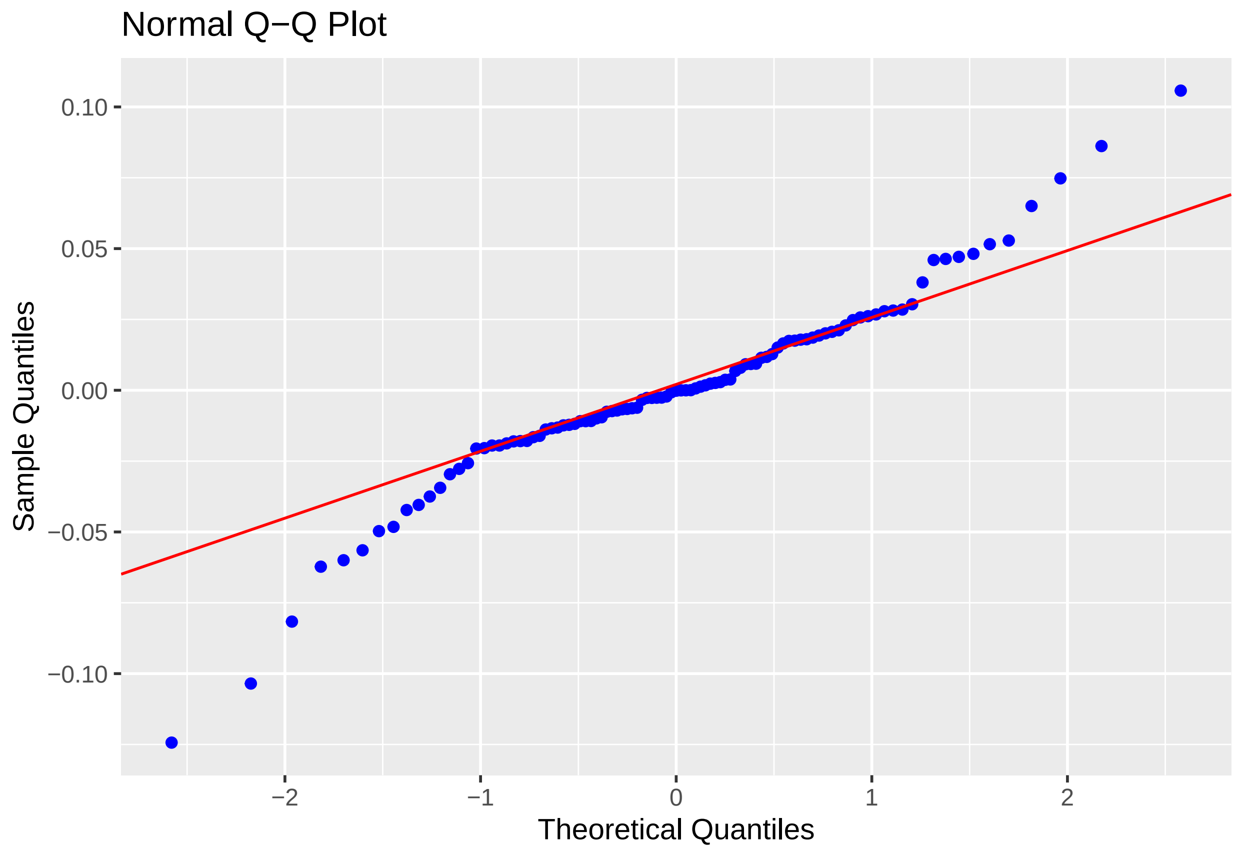
```r
housing$log.sales.price<- log(housing$sales.price)

housing_fit_log<- lm(log.sales.price ~ List.Price + tax.assessed.value+remodeled.kitchen+BA+architectur
                       data=housing%>%
                       dplyr::select(-X1, -Last.Chg.Type, -Status, -'MLS.#', -Type, -Address, -Area, -
```

Diagnostics

Normality

We see a log transformation on Sales Price does not help our error terms appear more normal.

```
ols_plot_resid_qq(housing_fit_log)
```




```
#Studentized Throws error due to a single Nan value for studres, will be removed in outliers
#qqPlot(housing_fit_log,ylab="Studentized Residuals", xlab="Theoretical Quantiles")
```

Transformation Needed: Sqrt Sales Price?

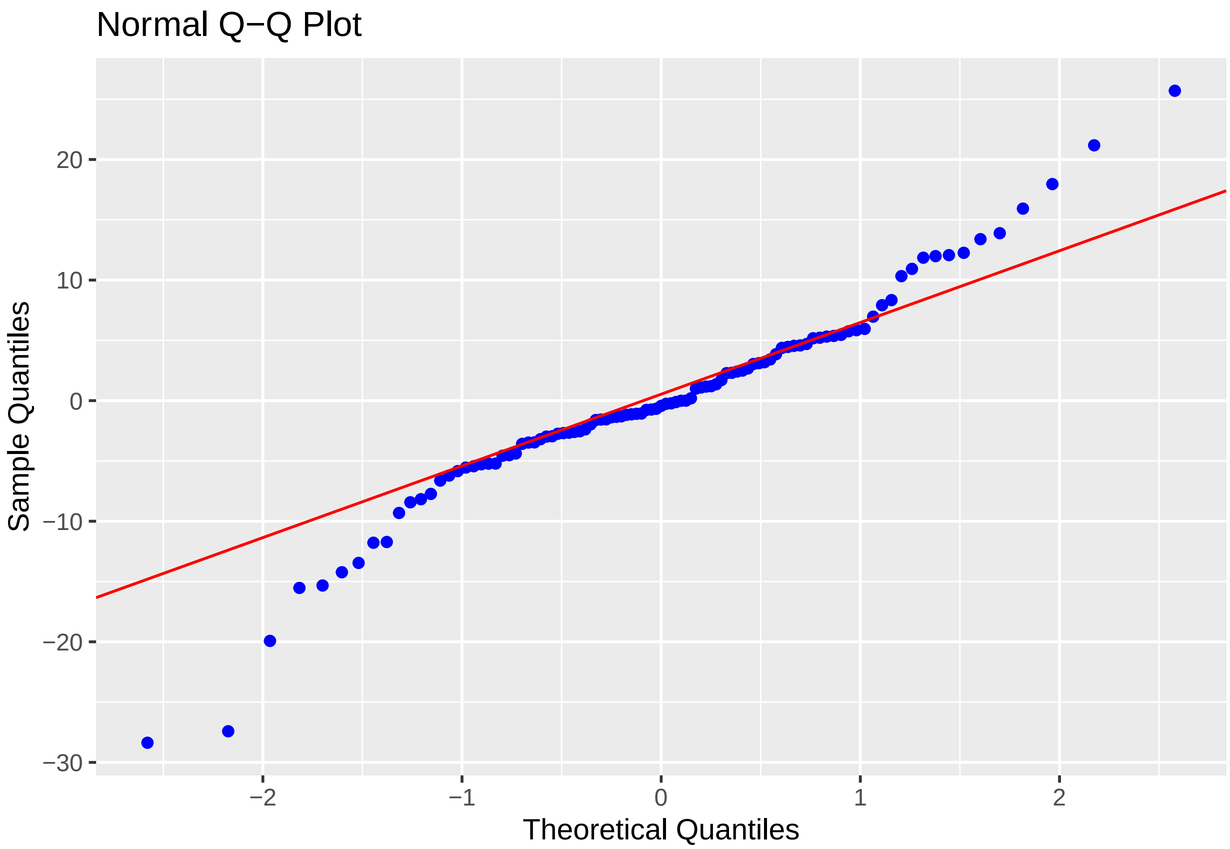
Next we try a square root transformation on Sales price to make the error appear more normal.

```
housing$sqrt.sales.price<- sqrt(housing$sales.price)

housing_fit_sqrt<- lm(sqrt.sales.price ~ List.Price + tax.assessed.value+remodeled.kitchen+BA+architectu
                      data=housing%>%
                      dplyr::select(-X1, -Last.Chg.Type, -Status, -'MLS.#', -Type, -Address, -Area, -
```

Normality

We see it is not helpful.

```
ols_plot_resid_qq(housing_fit_sqrt)
```




```
#Studentized Residuals plot throws error due to Nan stud.residuals which will be addressed in outliers
#qqPlot(housing_fit_sqrt,ylab="Studentized Residuals", xlab="Theoretical Quantiles")
```

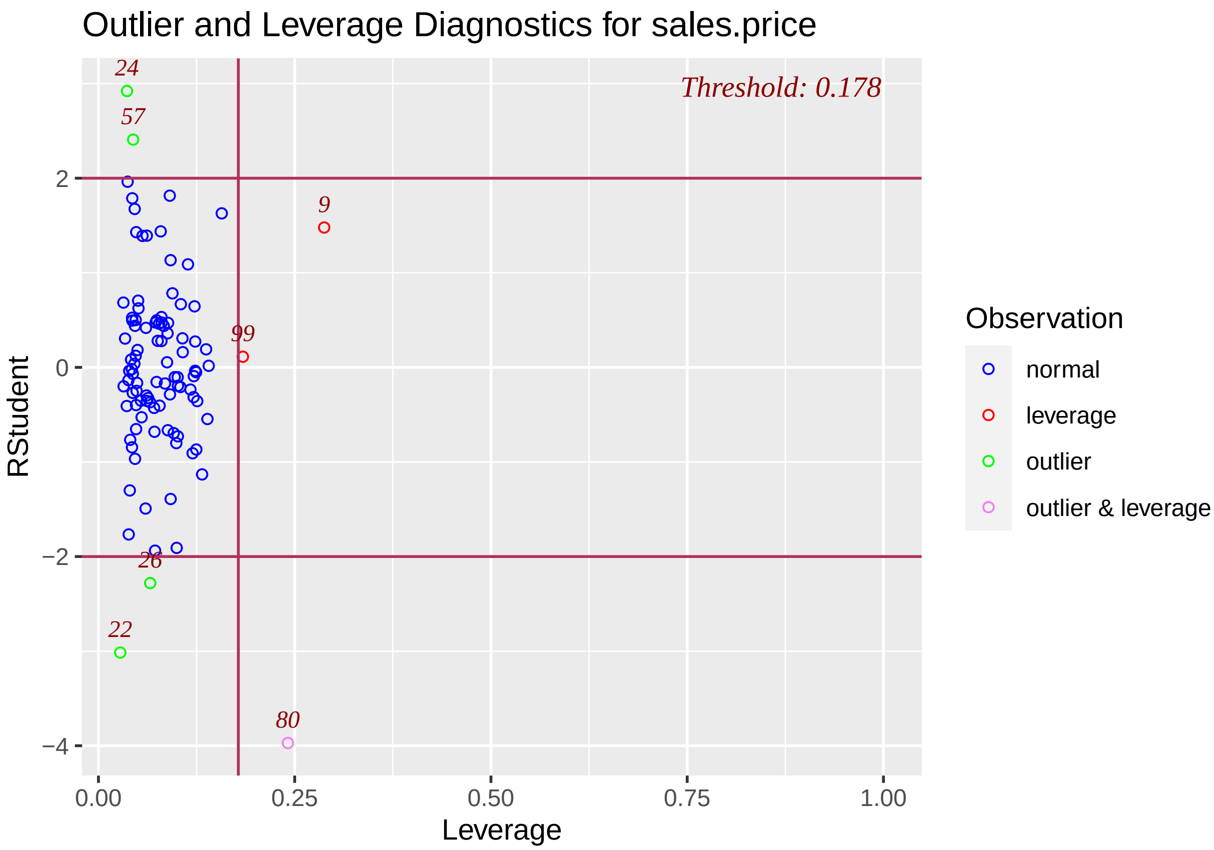
Leverage Points

Now we analyze for leverage points and HIPS to refine our model data.

```
#Leverage Points
leverage<-ols_leverage(housing_fit_best)
cut<-(4*9)/nrow(housing)
which(leverage>=cut)
```

```
## [1] 84
```

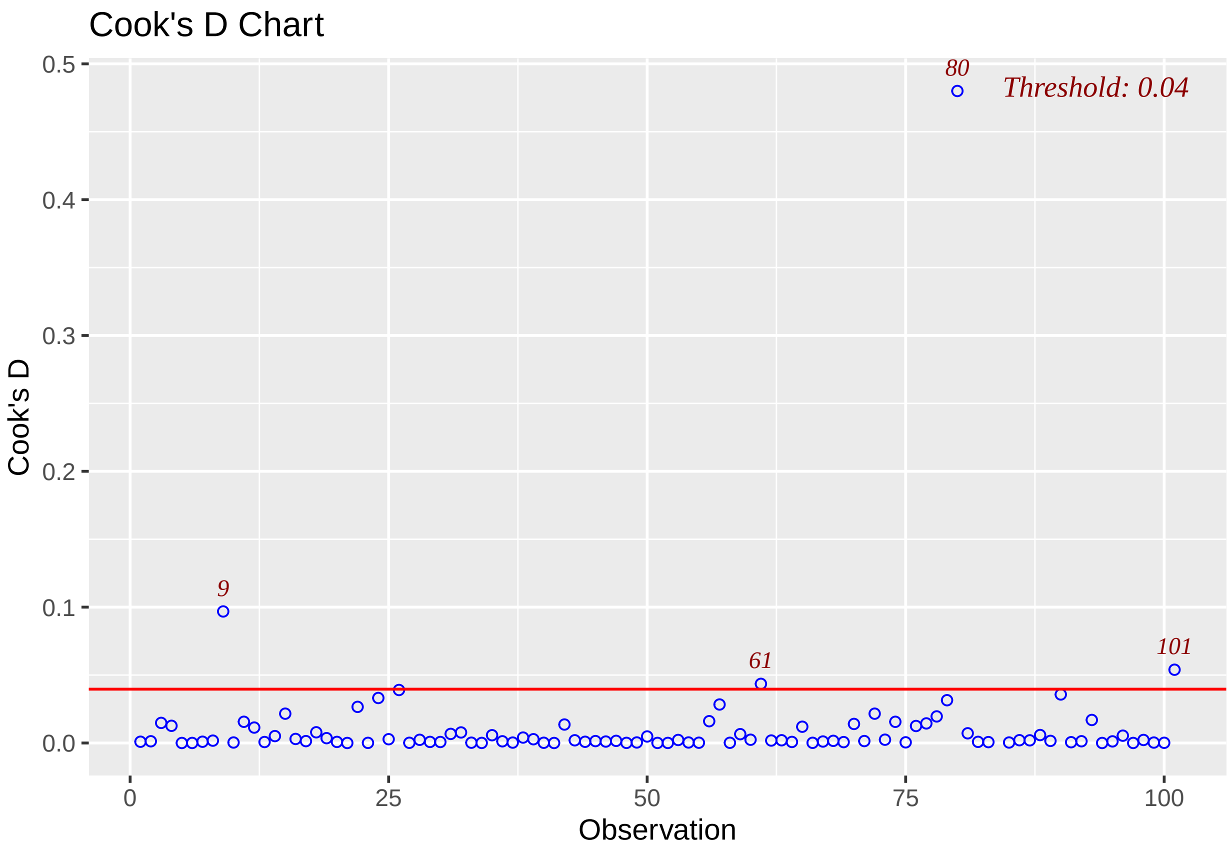
```
ols_plot_resid_lev(housing_fit_best)
```



#

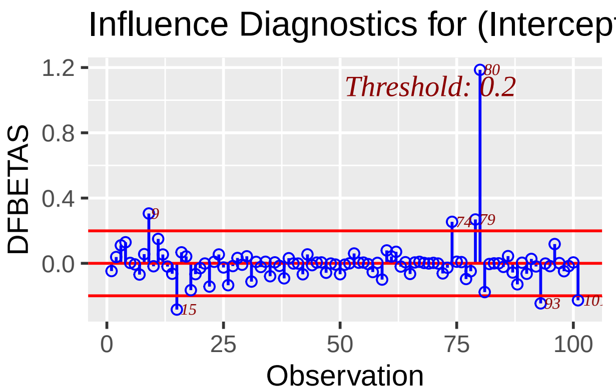
Highly Influential Points

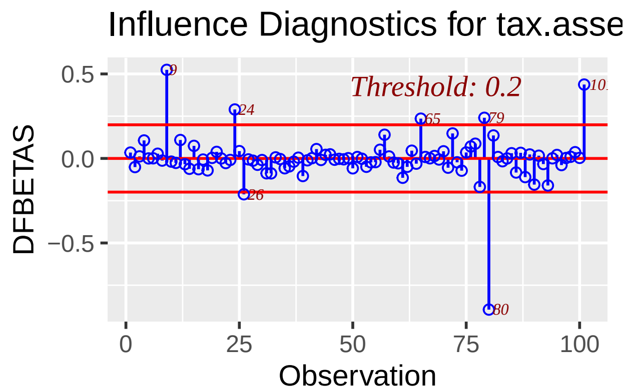
```
ols_plot_cooksd_chart(housing_fit_best)
```

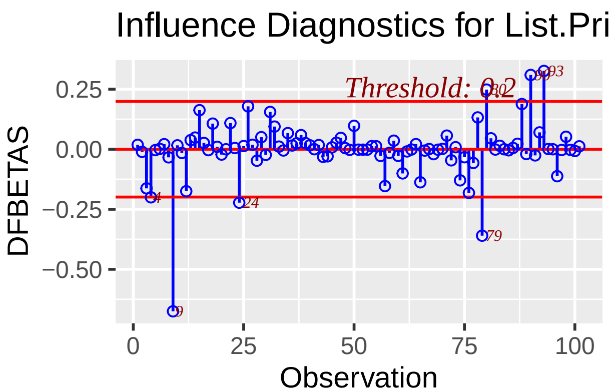


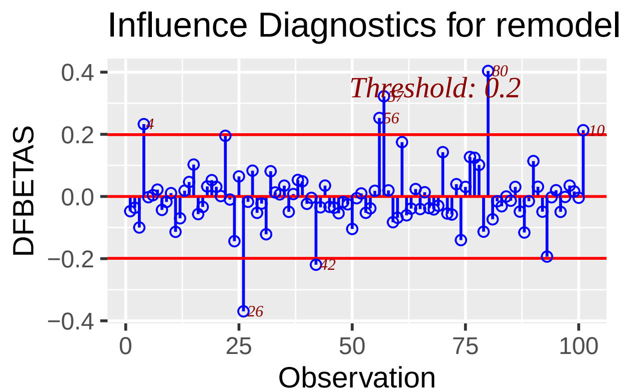

```
ols_plot_dfbetas(housing_fit_best)
```

page 1 of 3

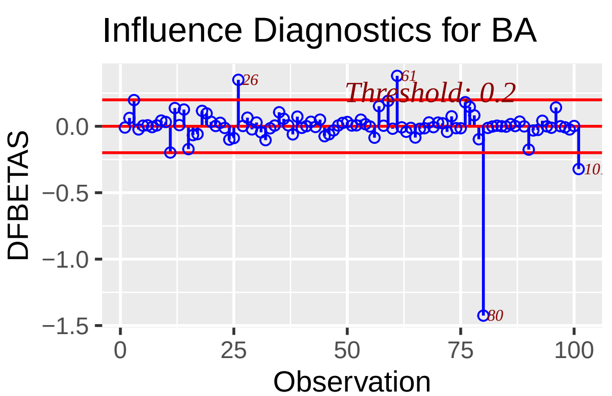


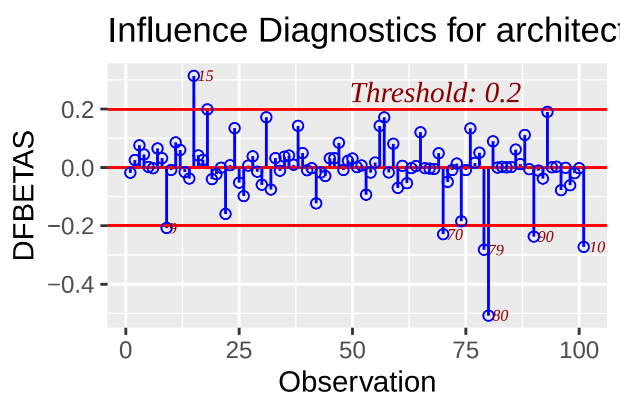


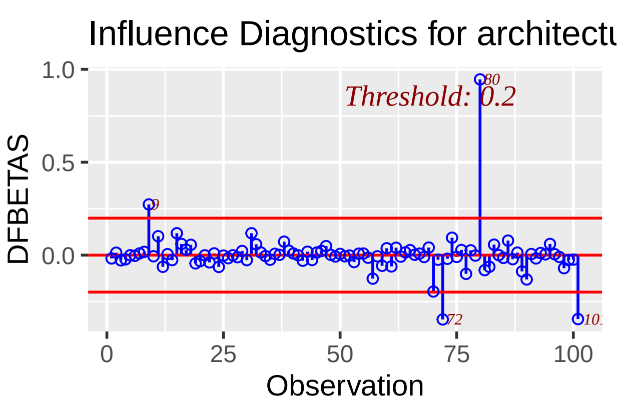


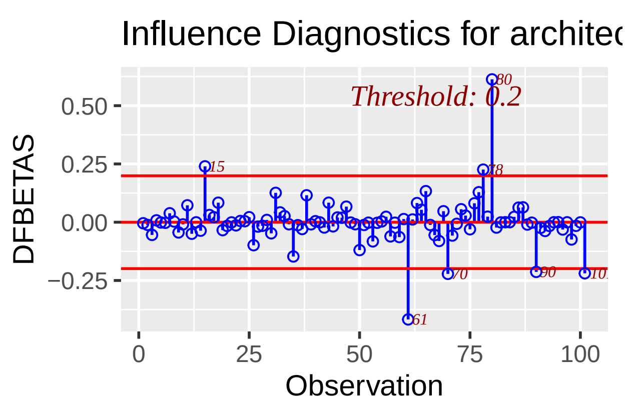


page 2 of 3

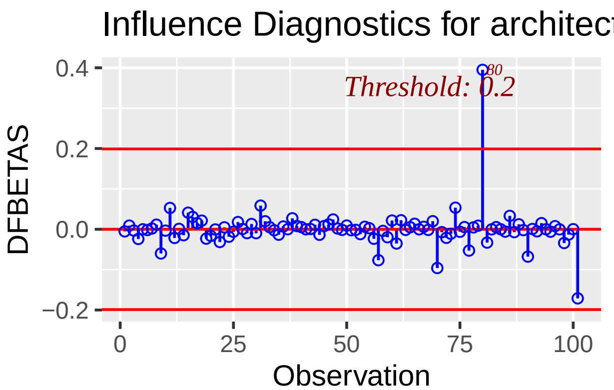




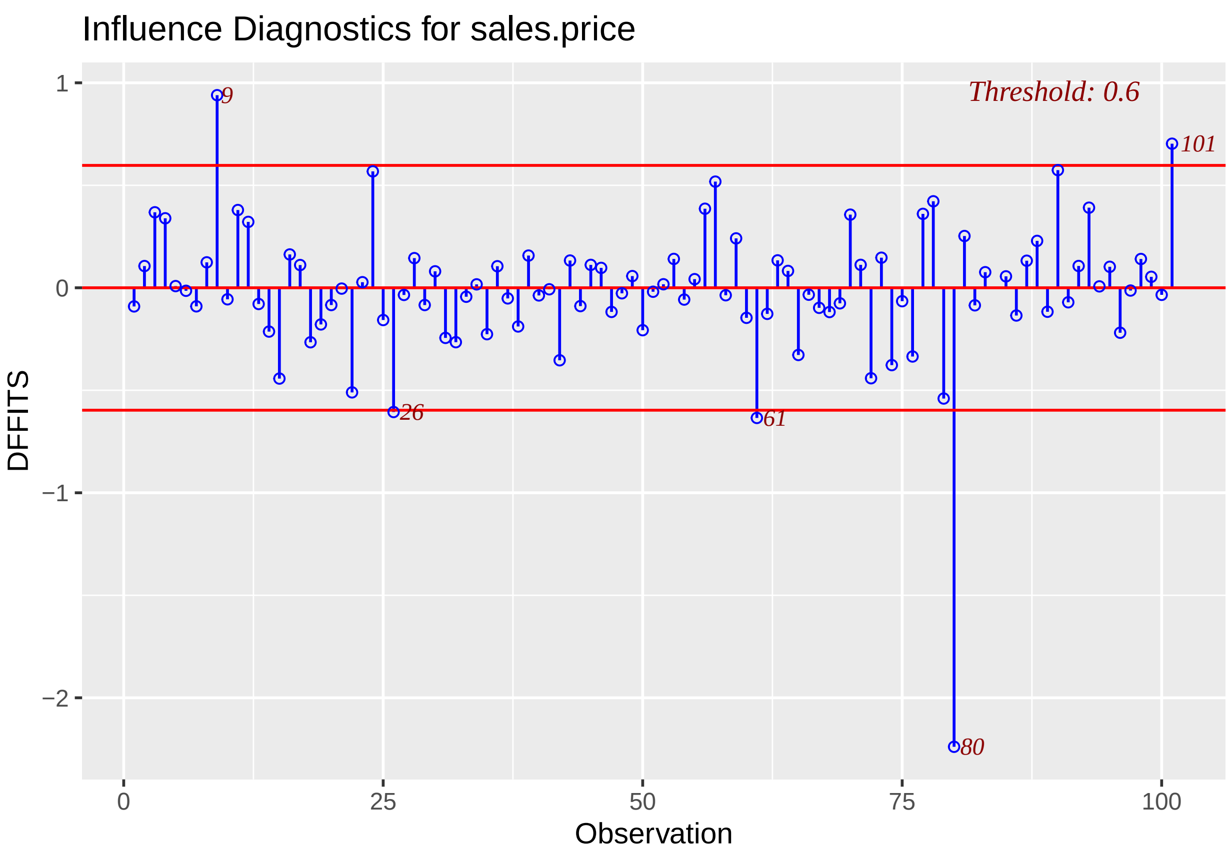




page 3 of 3




```
ols_plot_dffits(housing_fit_best)
```



Removal of Outliers After analysis of outliers, I will remove point 84 from our data, corresponding to X1 value of 295 or Address=9709 Wildbriar LN, as the studentized residuals are Nan - this is considered a leverage point due to it being the only classification of "two-story" home. I will refit my model without this point included in the raw data.

```
#Full Model
#Remove point 84 from data, corresponding to 9709 Wildbriar LN

housing<- housing%>%
  dplyr::filter(Address!="9709 Wildbriar LN")
```

```r
#Fit full model

housing_fit_full_outlier<- lm(sales.price~ . , data=housing %>%
                        dplyr::select(-X1, -Last.Chg.Type, -Status, -'MLS.#', -Type, -Address, -Area, -

#Best subsets
ols_step_both_aic(housing_fit_full_outlier)
```

```
##
##
##                                              Stepwise Summary
## -----------------------------------------------------------------------------------------------------
## Variable               Method       AIC           RSS              Sum Sq          R-Sq      Adj. R
## -----------------------------------------------------------------------------------------------------
## List.Price            addition    2116.804    8602642091.848    131149251172.902    0.93844      0.93
## tax.assessed.value    addition    2115.998    8364571795.264    131387321469.486    0.94015      0.93
## remodeled.kitchen     addition    2115.035    8120400266.185    131631492998.565    0.94189      0.94
## BA                    addition    2114.789    7939996422.084    131811896842.666    0.94319      0.94
## -----------------------------------------------------------------------------------------------------
```

```r
#Best Fit
housing_fit_best_outlier<- lm(sales.price ~ List.Price + tax.assessed.value+remodeled.kitchen+BA,
                        data=housing)

summary(housing_fit_best_outlier)$coefficients
```

```
##                          Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)           5.421202e+03 6.683808e+03  0.8110949 4.193386e-01
## List.Price            9.260175e-01 3.584146e-02 25.8364884 9.441030e-45
## tax.assessed.value    7.842294e-02 3.406837e-02  2.3019280 2.352353e-02
## remodeled.kitchenyes  3.513167e+03 1.944984e+03  1.8062708 7.404248e-02
## BA                   -2.940097e+03 2.001184e+03 -1.4691784 1.450882e-01
```

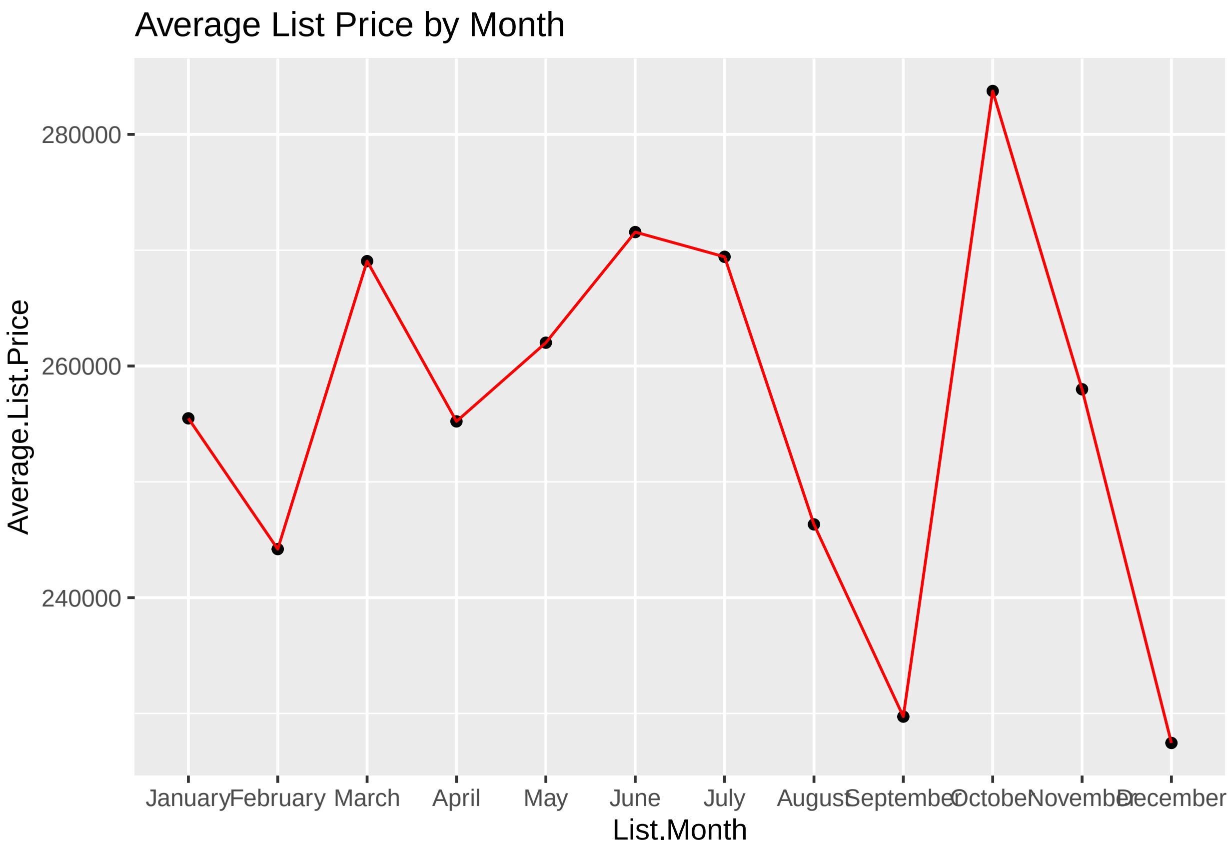
Exploratory Data Analysis

Below is exploratory data analysis of the included regressors in our model after removing outliers.

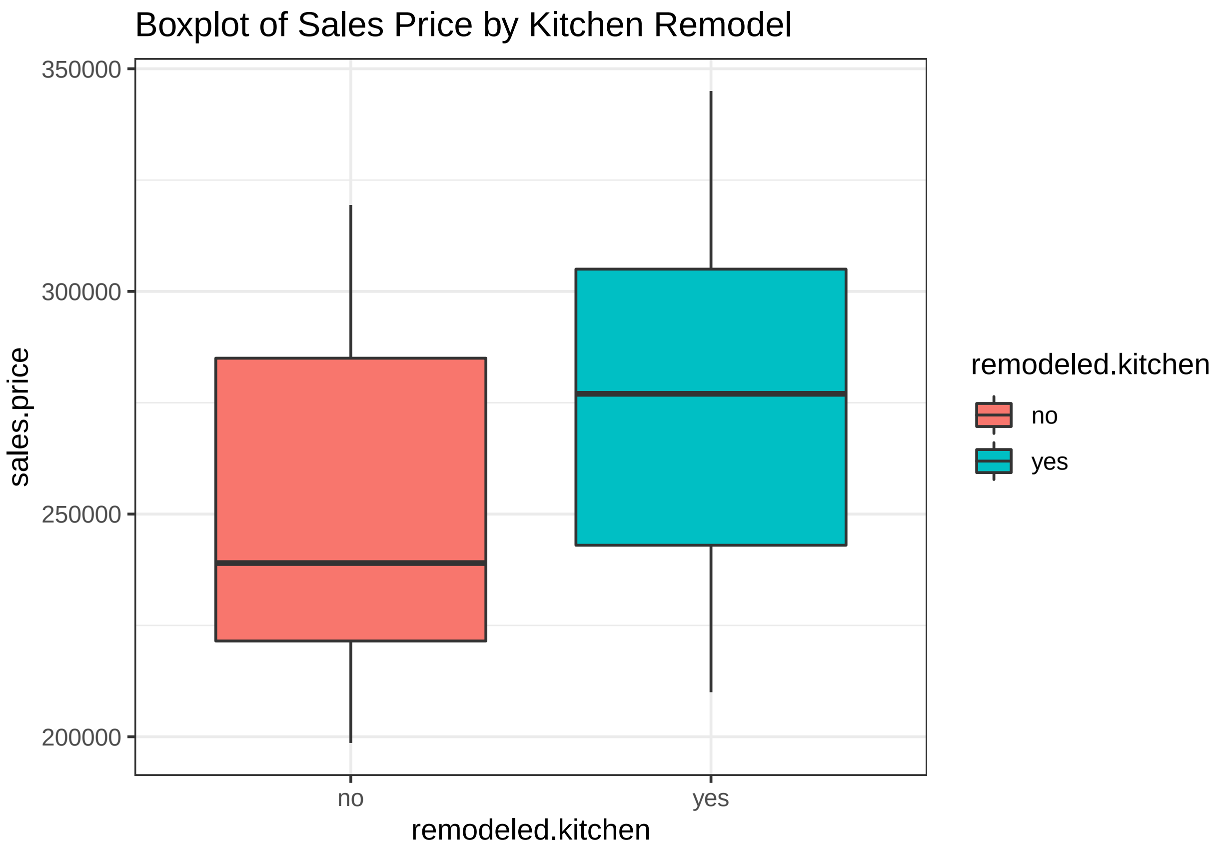
```r
housing$List.Month=factor(housing$List.Month, levels=c("January","February","March","April","May","June

listprice.bymonth<- housing%>%
  group_by(List.Month)%>%
  summarize(mean(List.Price))%>%
  dplyr::rename(Average.List.Price='mean(List.Price)')
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
#Line plot list price by month
ggplot(listprice.bymonth, aes(x=List.Month, y=Average.List.Price, group=List.Month))+
  geom_point()+
  geom_line(group=1, color="red")+
  labs(title="Average List Price by Month")
```

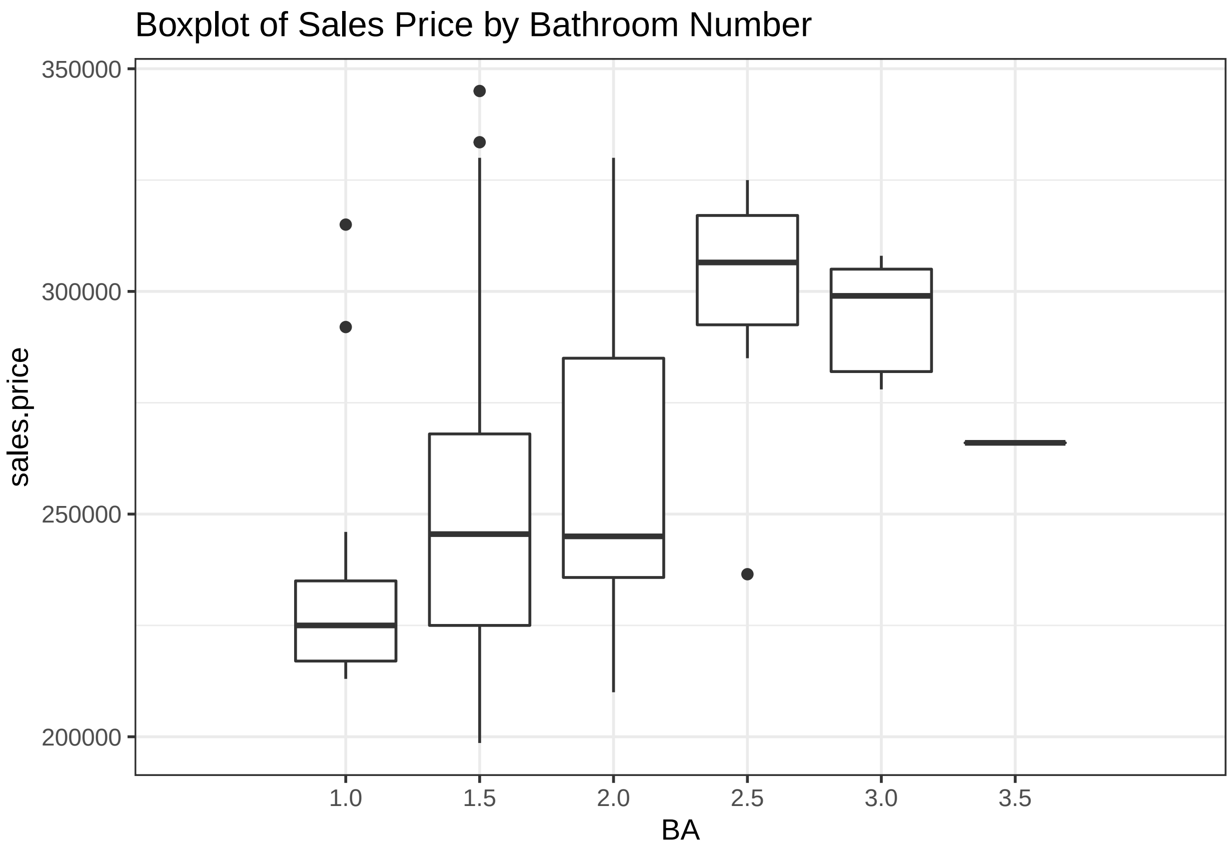



```
#Boxplot sales price remodeled kitchen
housing%>%
  ggplot(aes(x=remodeled.kitchen, y=sales.price, fill=remodeled.kitchen))+
  geom_boxplot()+
  theme_bw()+
  labs(title="Boxplot of Sales Price by Kitchen Remodel")
```

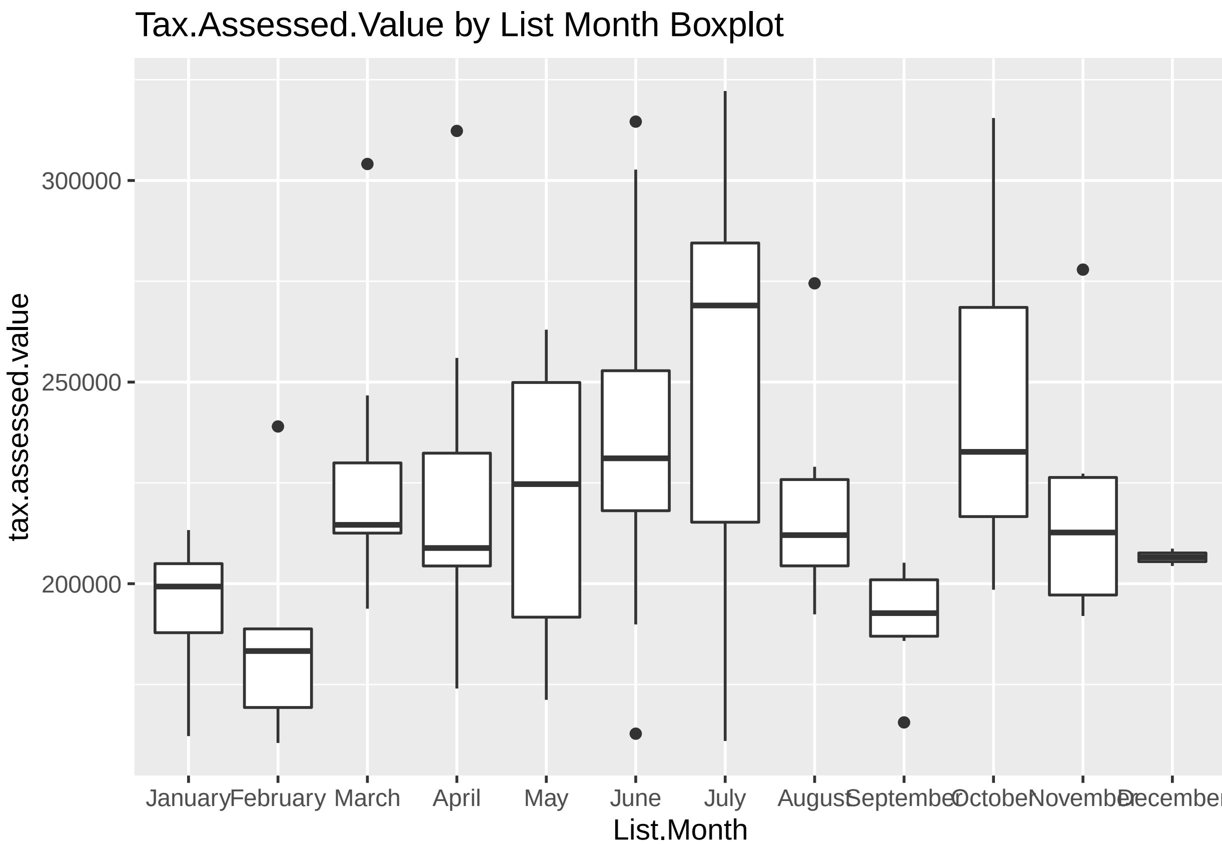



```
#Boxplot Sales Price/Bathrooms
housing%>%
  ggplot(aes(x=BA, y=sales.price, group=BA))+
  geom_boxplot()+
  theme_bw()+
  labs(title="Boxplot of Sales Price by Bathroom Number")+
  scale_x_discrete(limits=c(1,1.5,2,2.5,3,3.5))
```

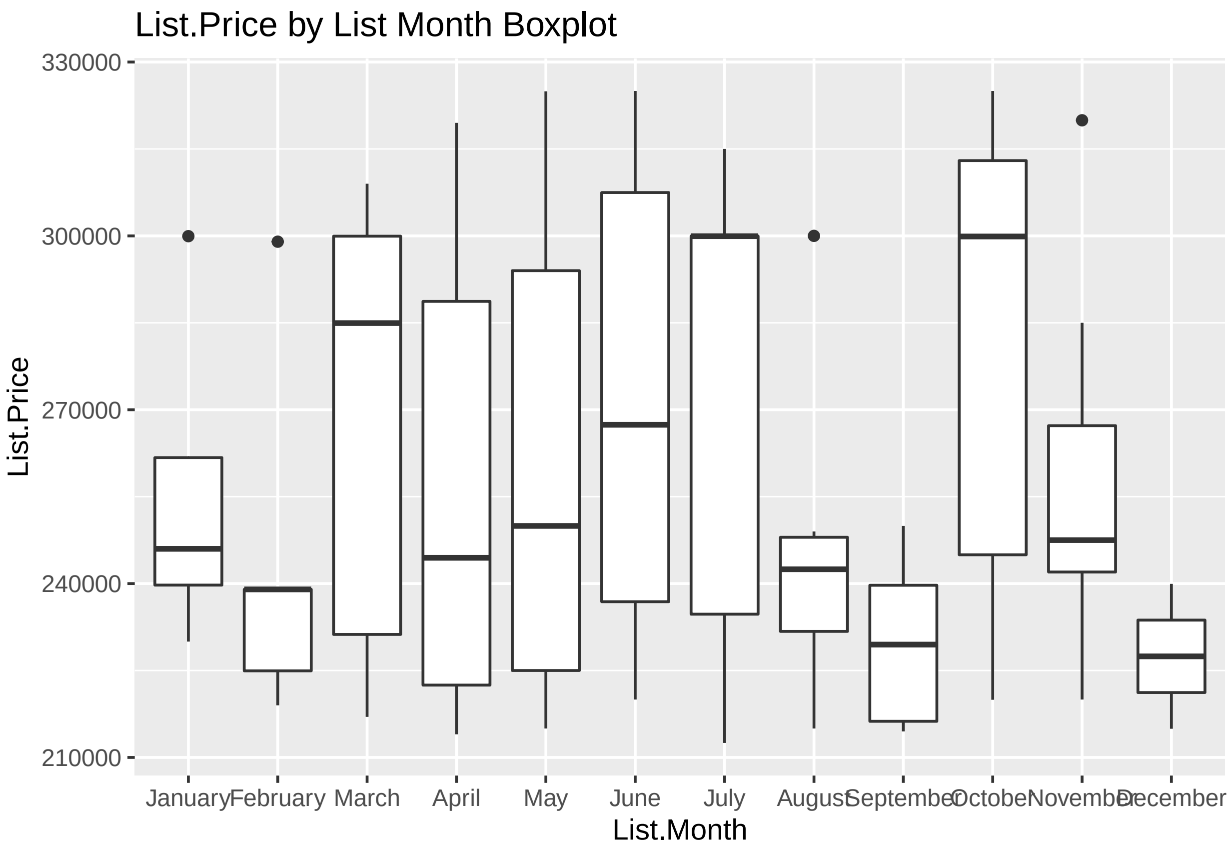
```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```




```
#Boxplot of tax assessed value by month
ggplot(data=housing, aes(y=tax.assessed.value, x=List.Month))+
  geom_boxplot()+
  labs(title="Tax.Assessed.Value by List Month Boxplot")
```



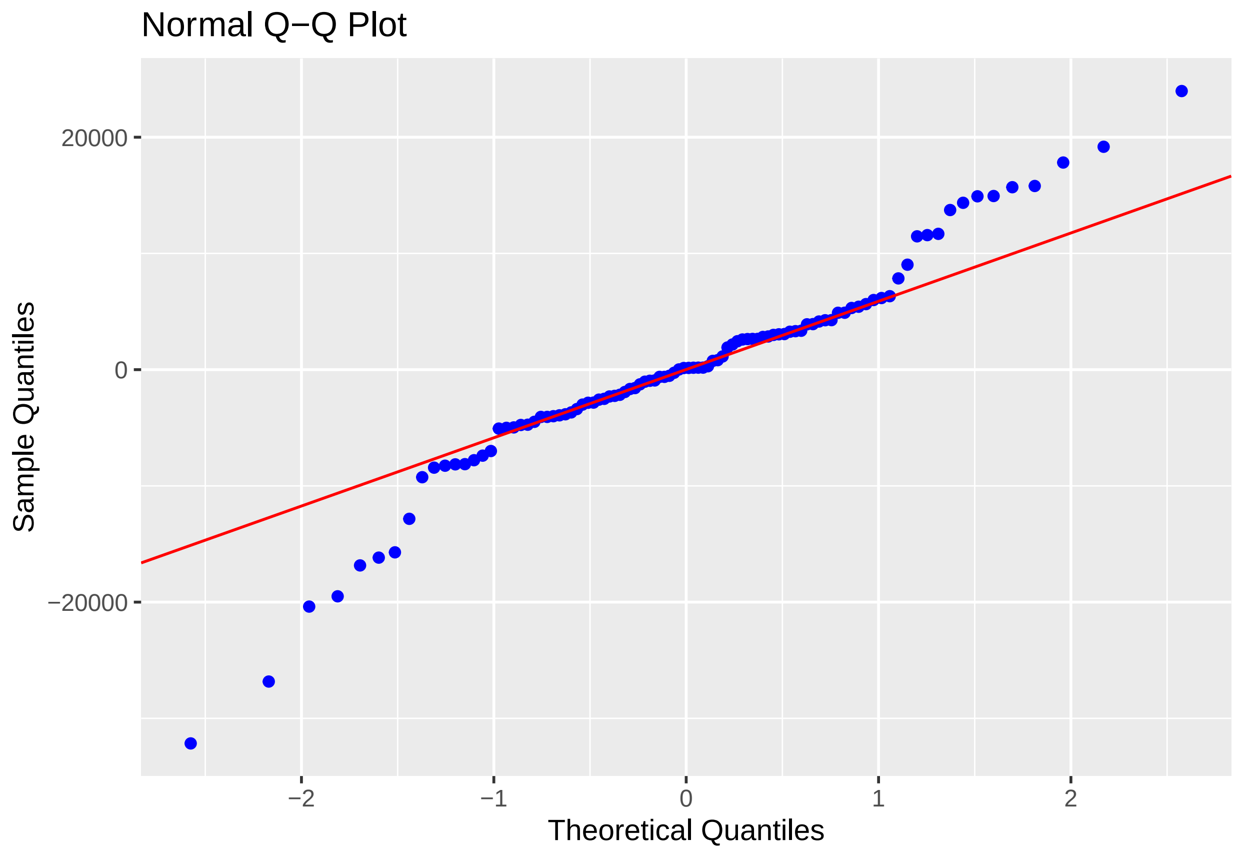

```r
#Boxplot of List Price by Month
ggplot(data=housing, aes(y=List.Price, x=List.Month))+
  geom_boxplot()+
  labs(title="List.Price by List Month Boxplot")
```



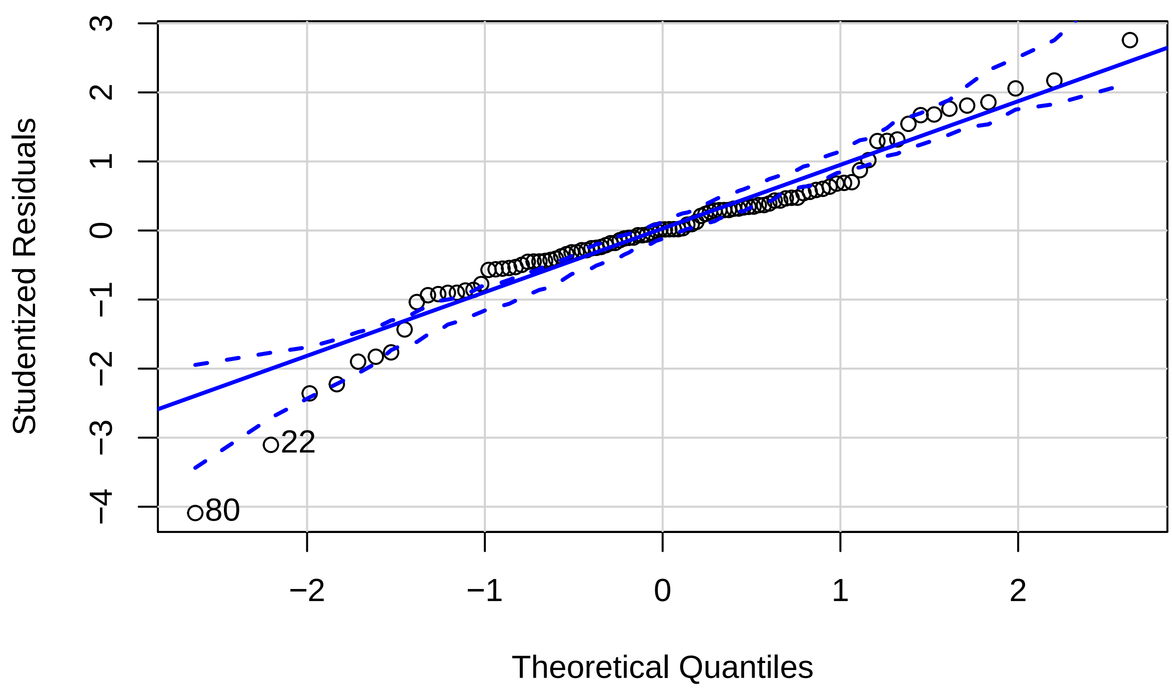
Normality

We still show some evidence of departure from the normal line at the low end of our observed data. This indicates that housing prices have a floor, which in this case may be around $200,000.

```
ols_plot_resid_qq(housing_fit_best_outlier)
```




```
qqPlot(housing_fit_best_outlier,ylab="Studentized Residuals",
xlab="Theoretical Quantiles")
```

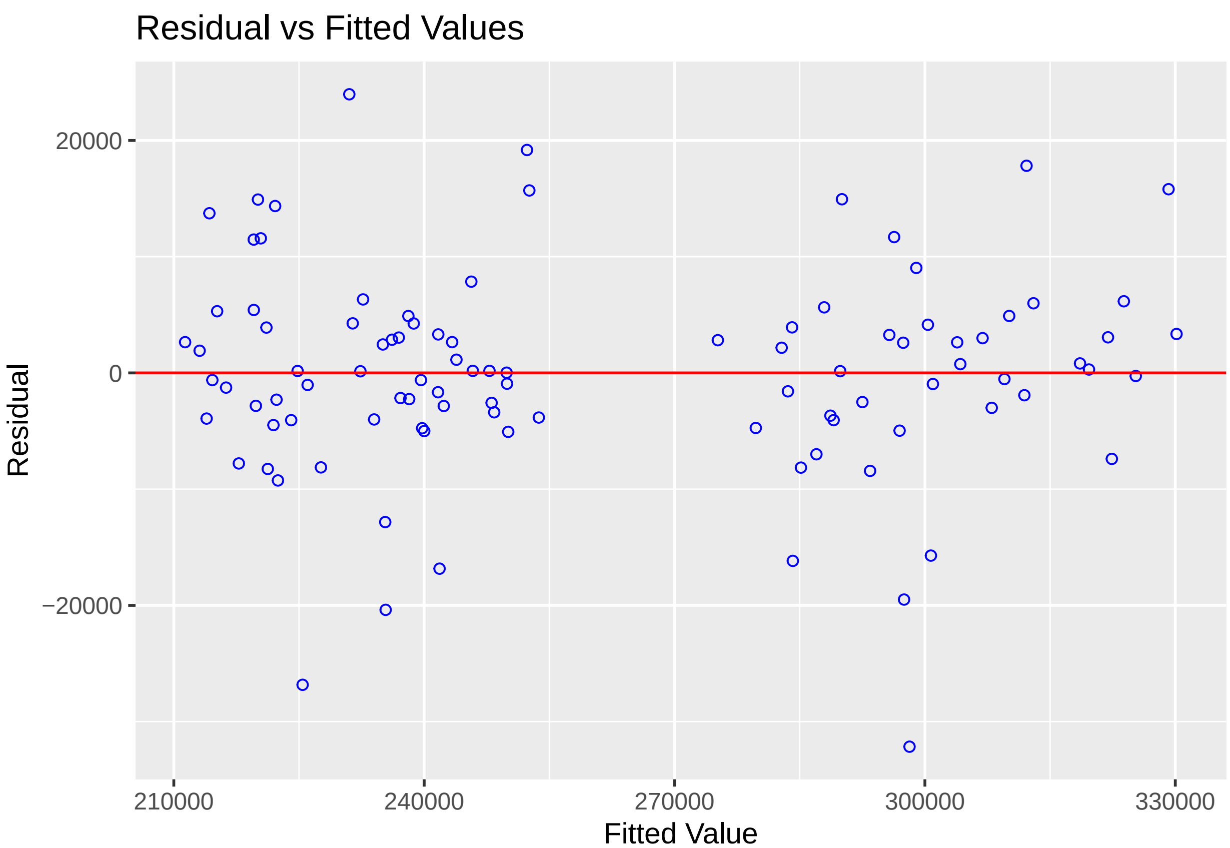



```
## [1] 22 80
```

Constant variance

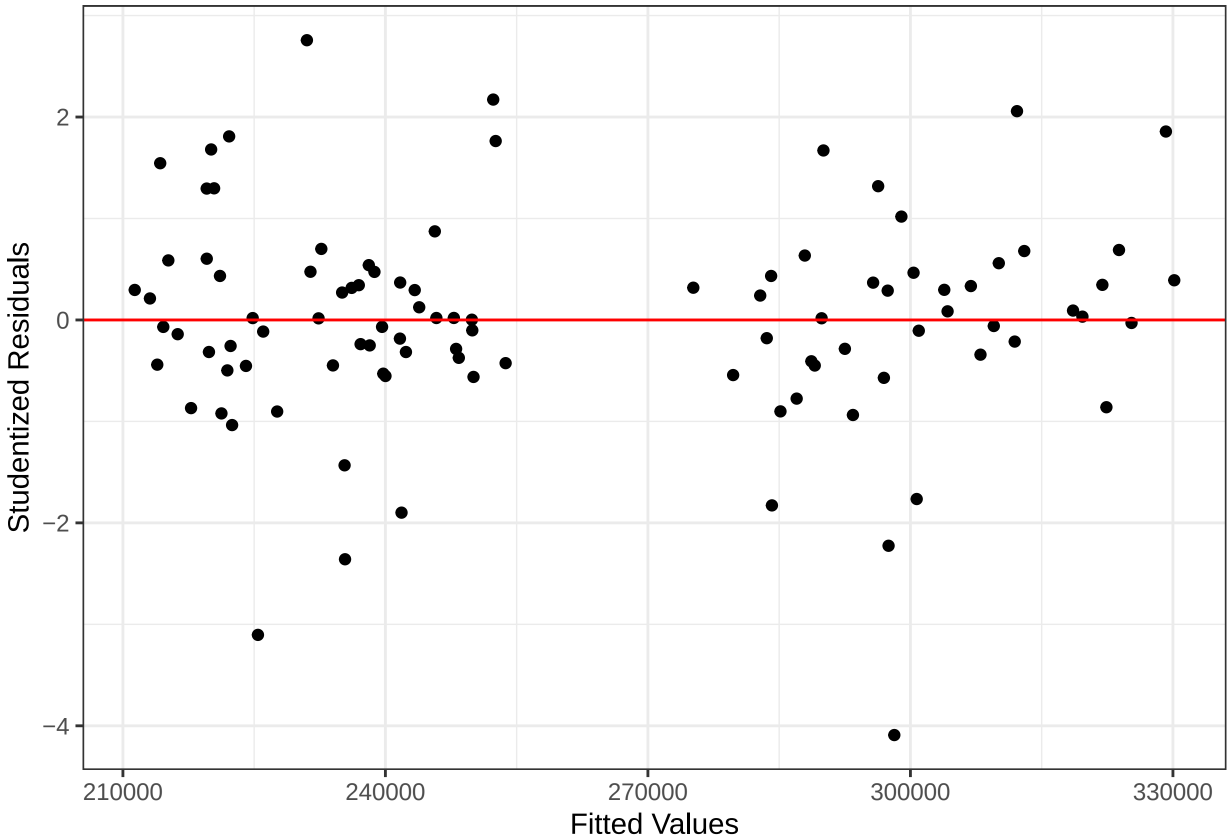
We see constant variance in our residuals vs. fitted values.

```
#Regular Residuals
ols_plot_resid_fit(housing_fit_best_outlier)
```



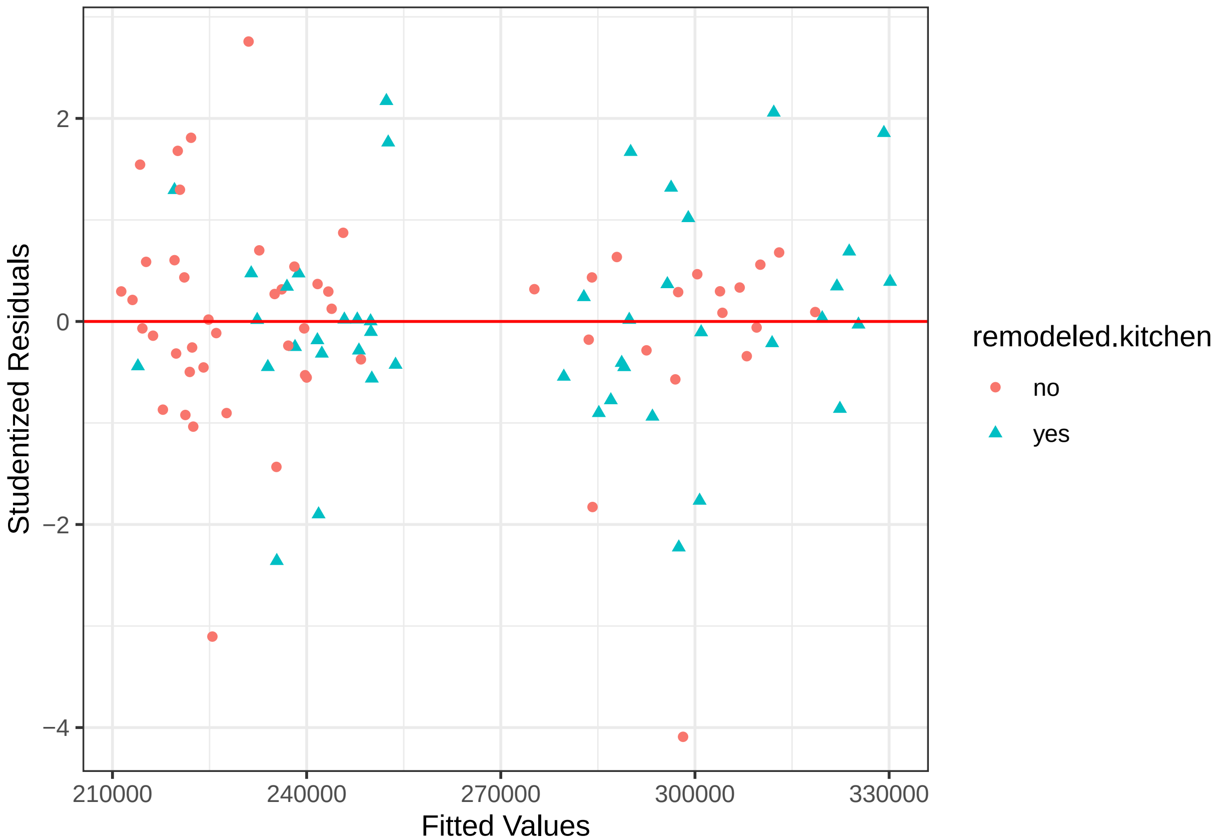

```
#Studentized Residuals
housing$studres<- studres(housing_fit_best_outlier)
housing$fitted<- housing_fit_best_outlier$fitted.values
housing$residuals<- housing_fit_best_outlier$residuals



p0<- ggplot(housing, aes(x=fitted, y=studres))+
  geom_point()+
  theme_bw()+
  labs(y="Studentized Residuals", x="Fitted Values")+
  geom_hline(yintercept=0, color="red")

p1<- ggplot(housing, aes(x=fitted, y=studres, color=remodeled.kitchen, shape=remodeled.kitchen))+
  geom_point()+
  theme_bw()+
  labs(y="Studentized Residuals", x="Fitted Values")+
  geom_hline(yintercept=0, color="red")
```

```
p0
```



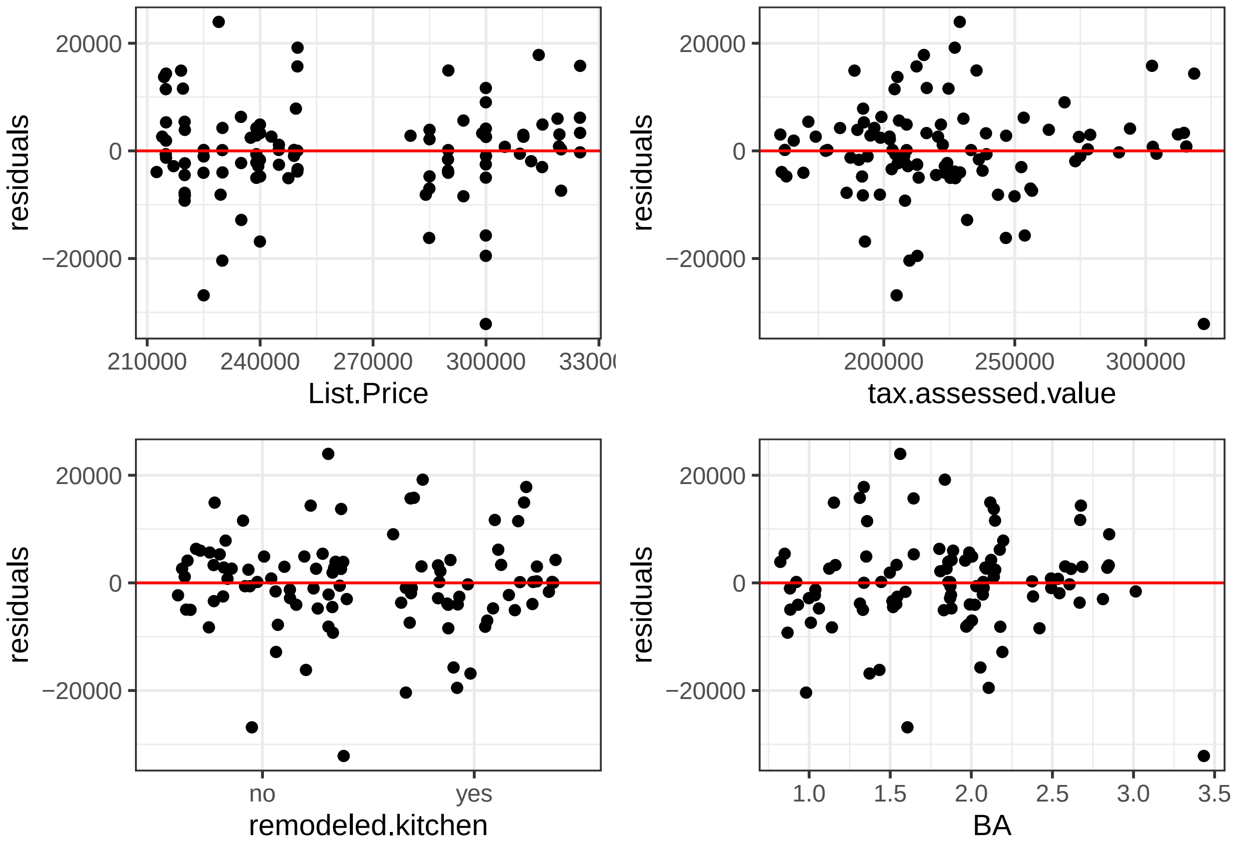
Studentized Residuals
2
0
−2
−4
210000
240000
270000
300000
330000
Fitted Values


```
p1
```

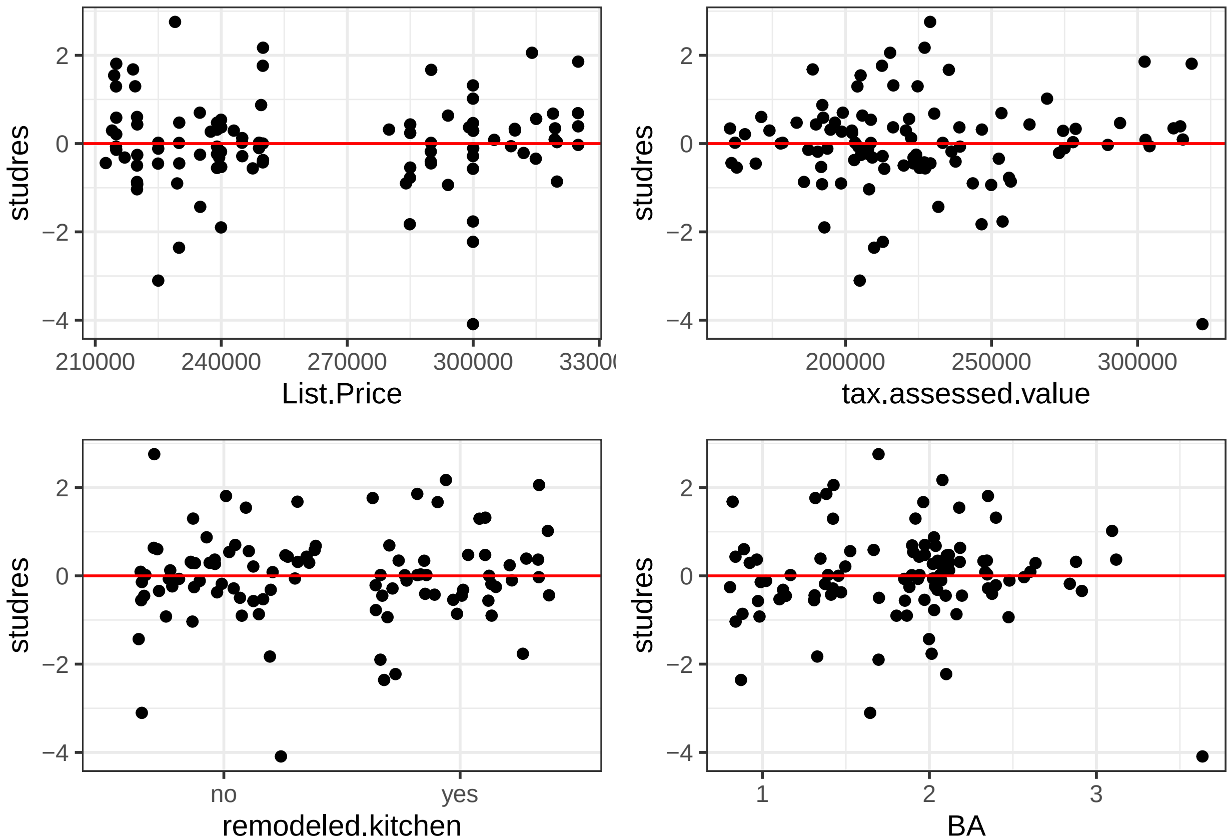


Residuals vs. Regresors in Model Our residuals vs. Regressors in Model may indicate a transformation is needed on list price due to reverse funneling.

```
#Regular Residual Plot
p1<- ggplot(housing, aes(x=List.Price, y=residuals))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p2<- ggplot(housing, aes(x=tax.assessed.value, y=residuals))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p3<- ggplot(housing, aes(x=remodeled.kitchen, y=residuals))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p4<- ggplot(housing, aes(x=BA, y=residuals))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

grid.arrange(p1,p2,p3,p4, ncol=2)
```




```r
p1<- ggplot(housing, aes(x=List.Price, y=studres))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p2<- ggplot(housing, aes(x=tax.assessed.value, y=studres))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p3<- ggplot(housing, aes(x=remodeled.kitchen, y=studres))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p4<- ggplot(housing, aes(x=BA, y=studres))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

grid.arrange(p1,p2,p3,p4, ncol=2)
```



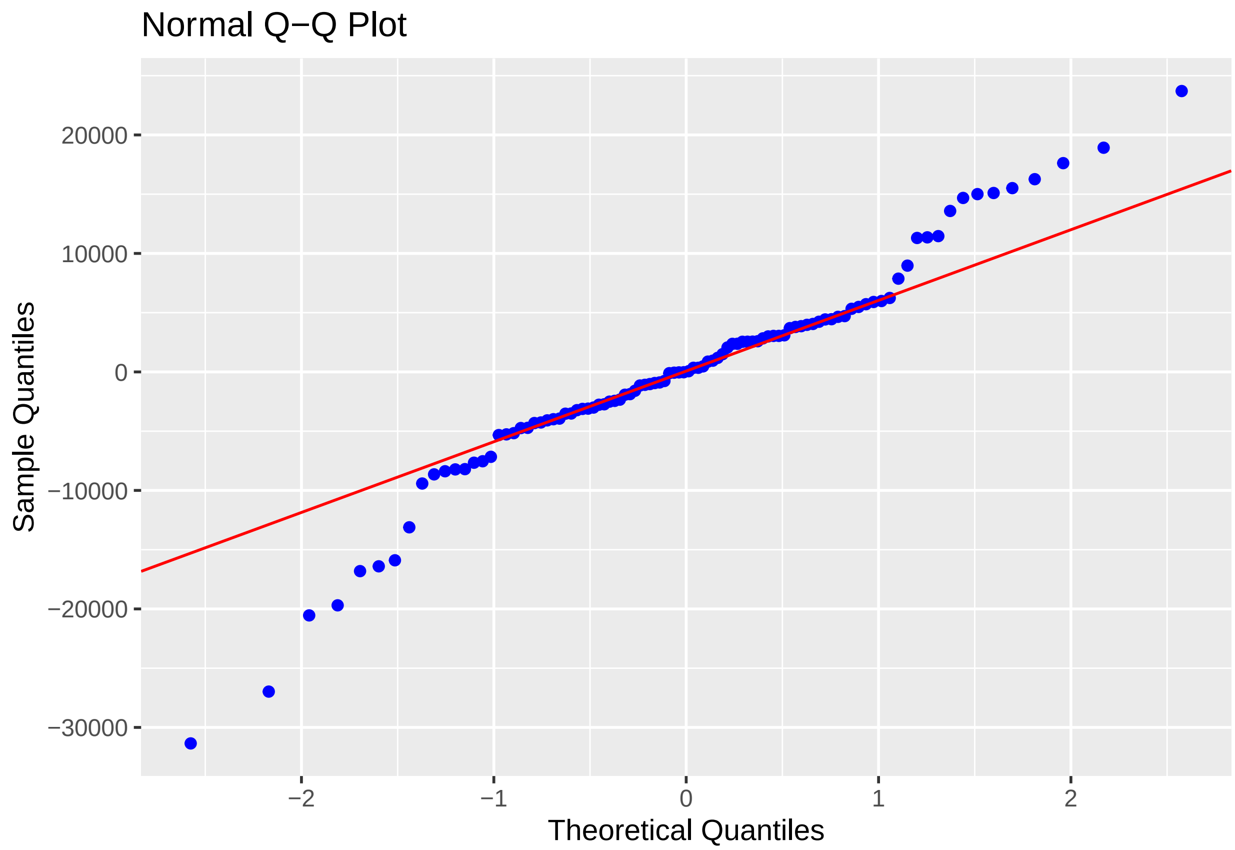
Transformation Needed?: Log tax.assessed value Next we try a log transformation on tax.assessed.value and recomplete our diagnostics.

```
housing$log.tax.assessed.value<-log(housing$tax.assessed.value)

housing_fit_log_tav<- lm(sales.price ~ List.Price + log.tax.assessed.value+remodeled.kitchen+BA,
                        data=housing)
```

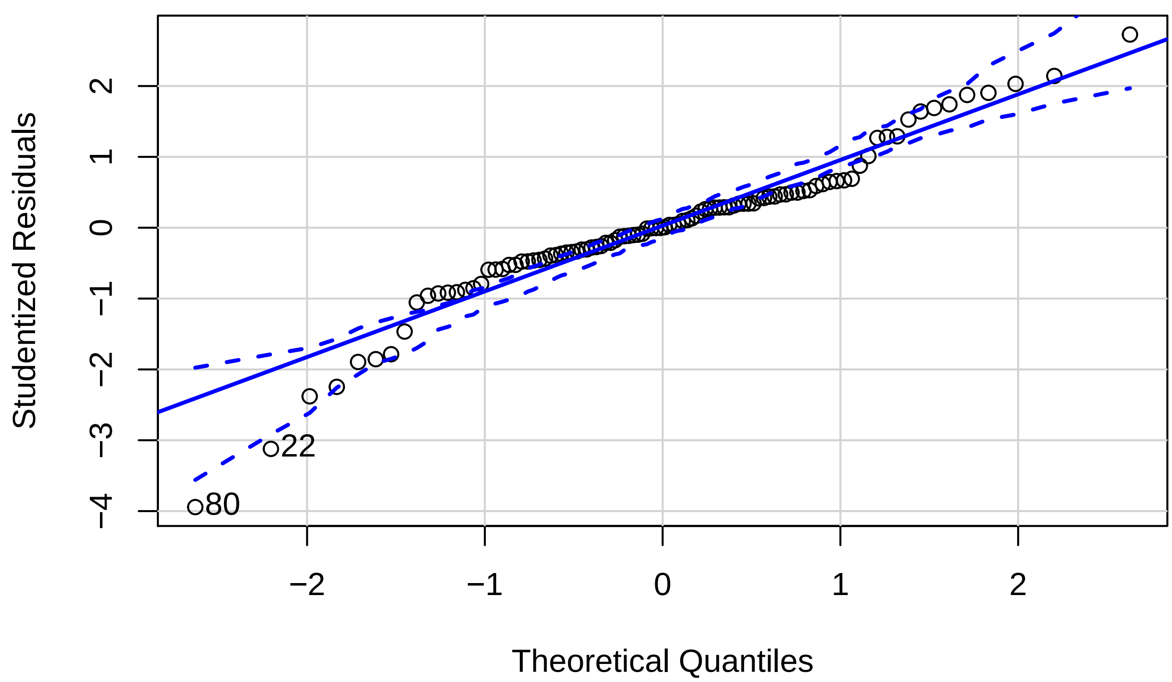
Normality

We see the same departures from the normal plot.

```
ols_plot_resid_qq(housing_fit_log_tav)
```




```
qqPlot(housing_fit_log_tav,ylab="Studentized Residuals",
xlab="Theoretical Quantiles")
```

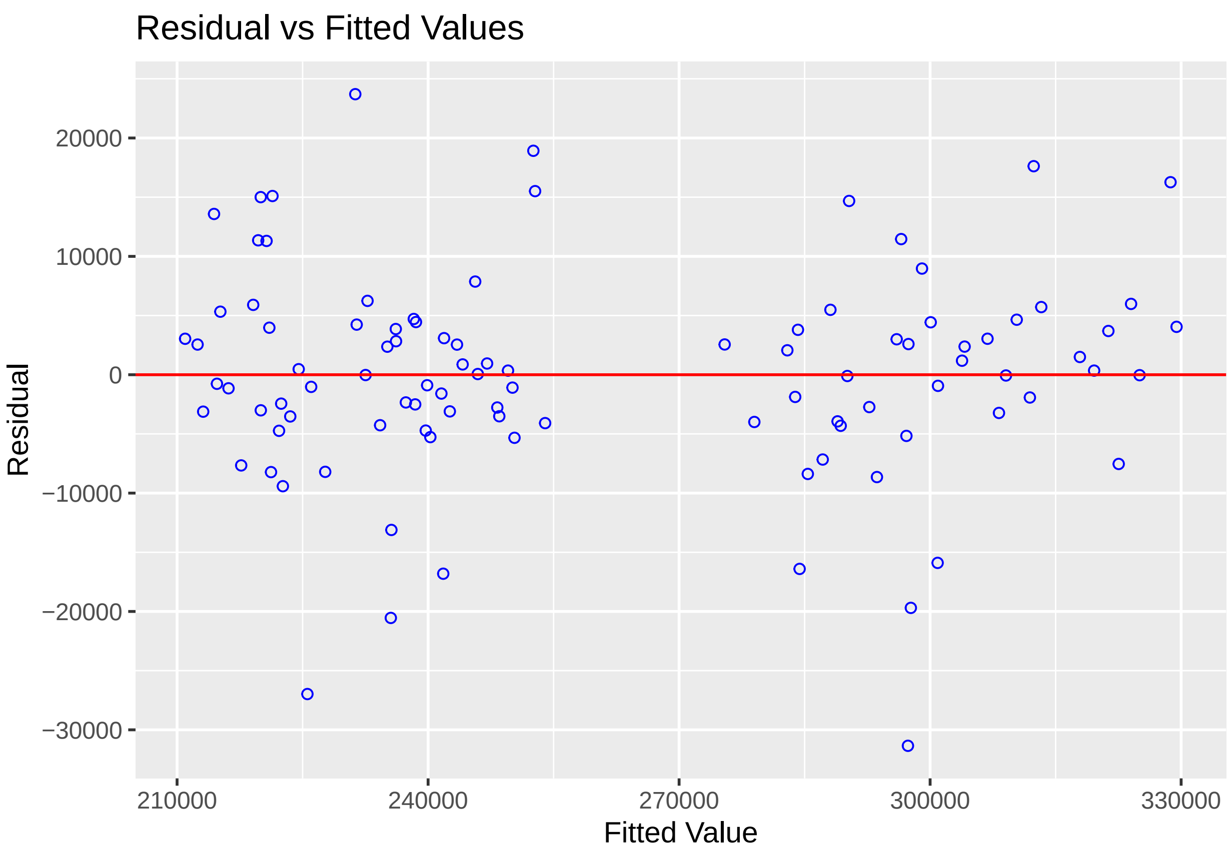



```
## [1] 22 80
```

Constant variance

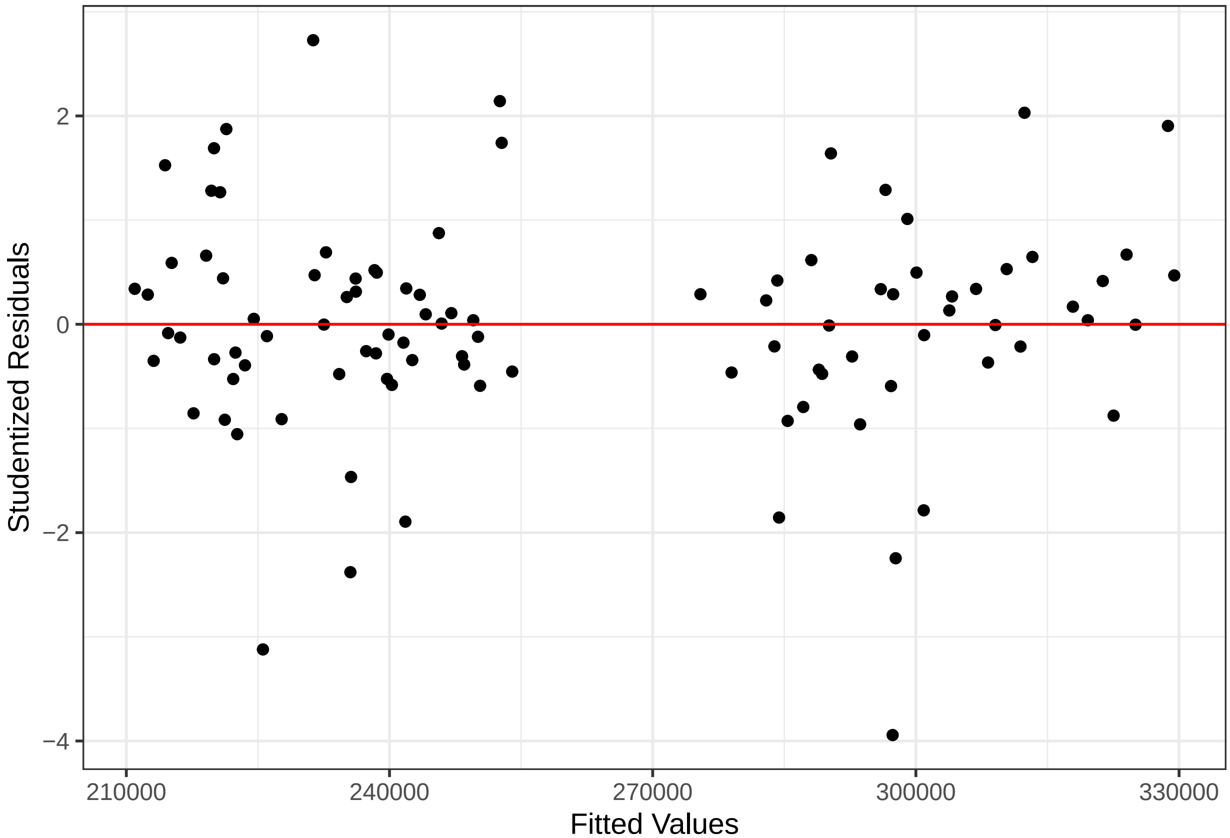
We still see constant variance of residuals vs. fitted.

```r
#Regular Residuals
ols_plot_resid_fit(housing_fit_log_tav)
```



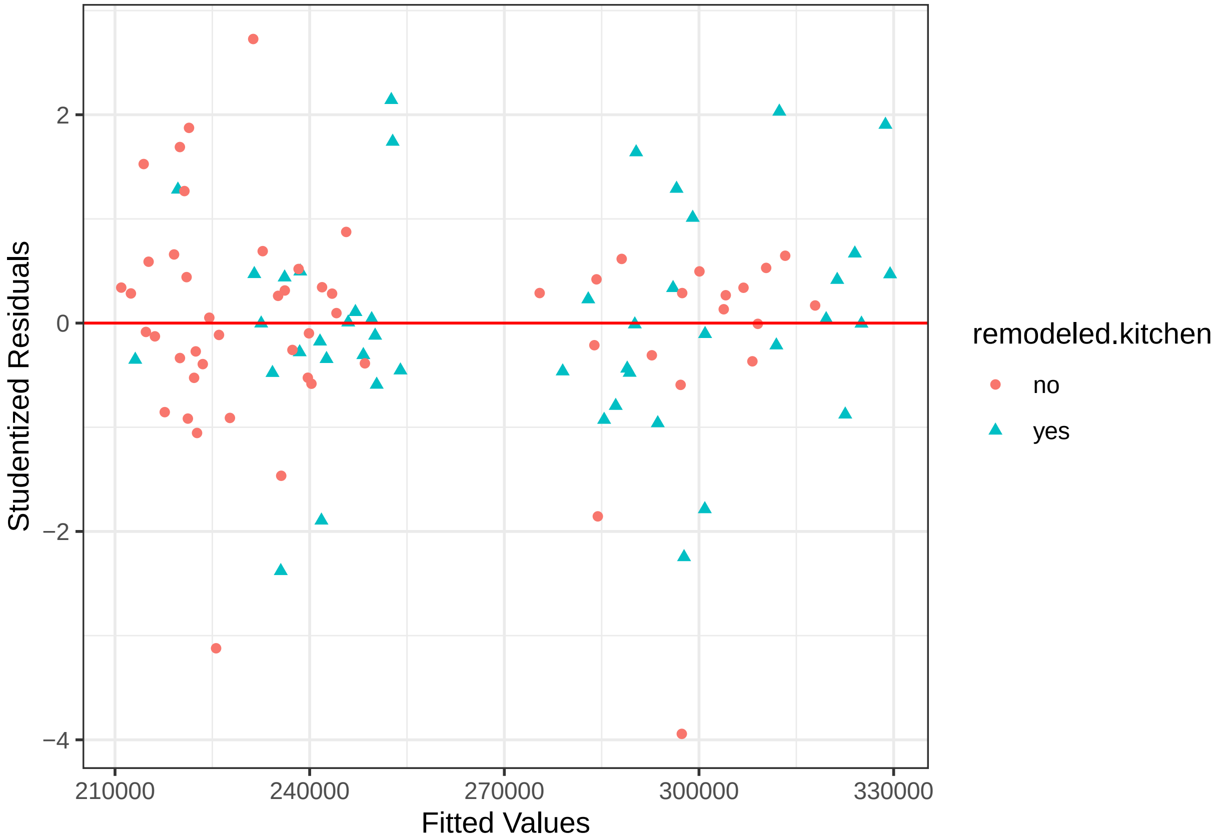

```r
#Studentized Residuals
housing$studres<- studres(housing_fit_log_tav)
housing$fitted<- housing_fit_log_tav$fitted.values
housing$residuals<- housing_fit_log_tav$residuals



p0<- ggplot(housing, aes(x=fitted, y=studres))+
  geom_point()+
  theme_bw()+
  labs(y="Studentized Residuals", x="Fitted Values")+
  geom_hline(yintercept=0, color="red")

p1<- ggplot(housing, aes(x=fitted, y=studres, color=remodeled.kitchen, shape=remodeled.kitchen))+
  geom_point()+
  theme_bw()+
  labs(y="Studentized Residuals", x="Fitted Values")+
  geom_hline(yintercept=0, color="red")
```

```
p0
```



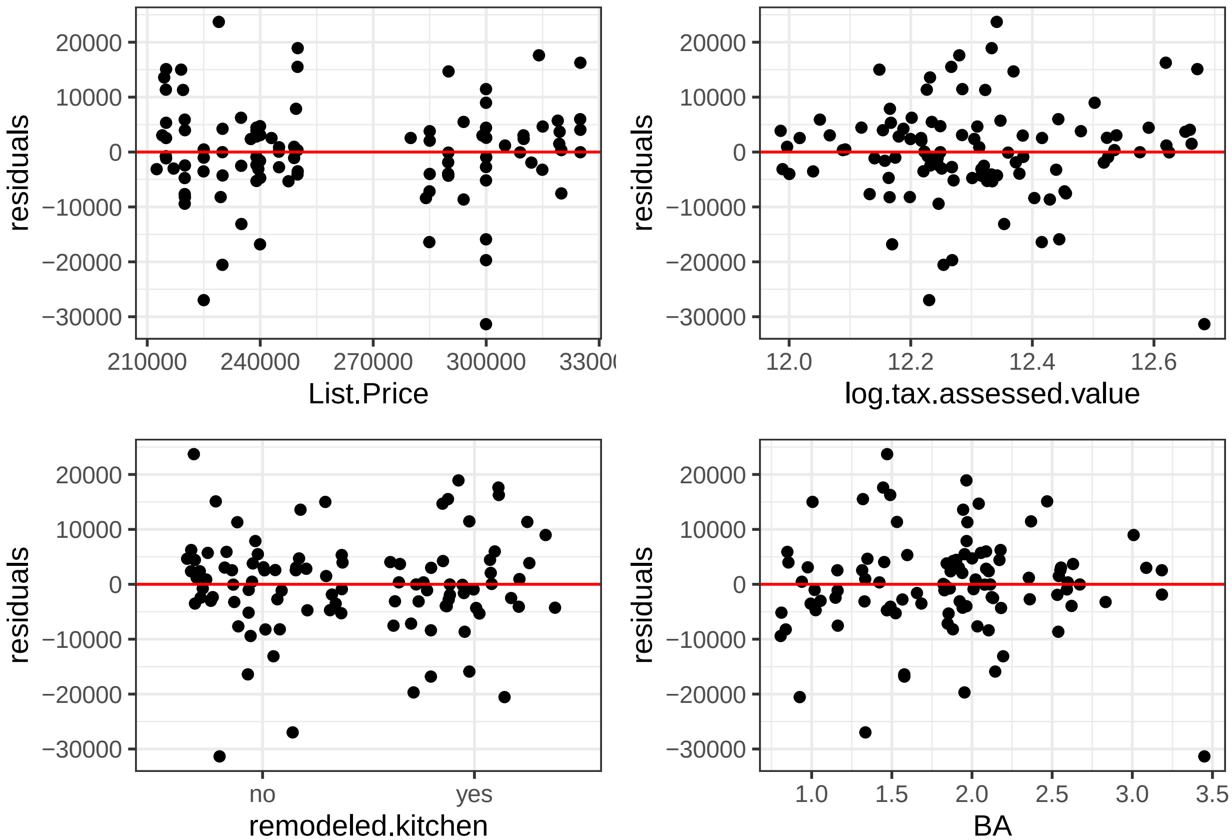
Studentized Residuals
Fitted Values
2
0
−2
−4
210000
240000
270000
300000
330000


```
p1
```

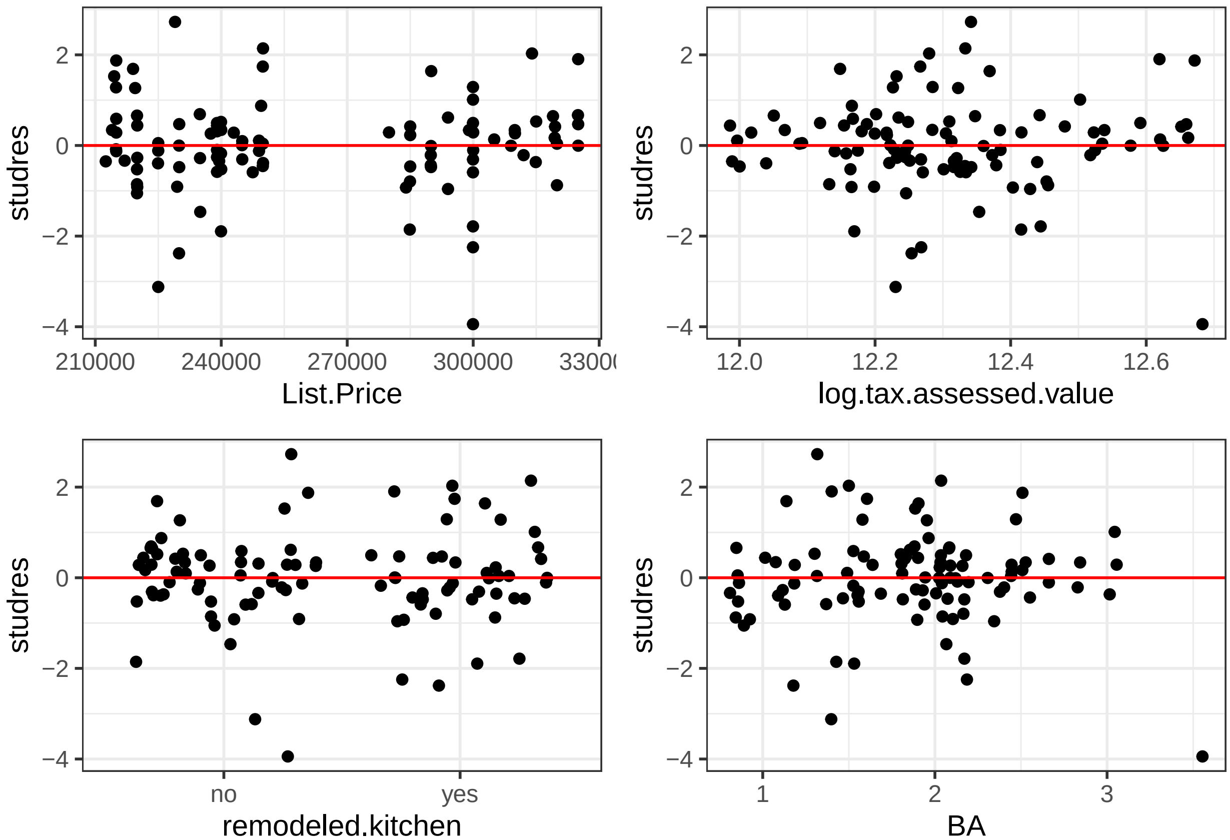


Residuals vs. Regresors in Model Our fitted vs. residuals still shows reverse funneling for tax.assessed.values. This may be due to our data, and few (<10) observed values at the low end of tax.assessed.value. Residuals amounts appear to not fluctuate highly at the low end of the fitted values for tax.assessed.value.

```
#Regular Residual Plot
p1<- ggplot(housing, aes(x=List.Price, y=residuals))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p2<- ggplot(housing, aes(x=log.tax.assessed.value, y=residuals))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p3<- ggplot(housing, aes(x=remodeled.kitchen, y=residuals))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p4<- ggplot(housing, aes(x=BA, y=residuals))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

grid.arrange(p1,p2,p3,p4, ncol=2)
```




```r
#Studentized Residual Plot
p1<- ggplot(housing, aes(x=List.Price, y=studres))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p2<- ggplot(housing, aes(x=log.tax.assessed.value, y=studres))+
  geom_point()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p3<- ggplot(housing, aes(x=remodeled.kitchen, y=studres))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

p4<- ggplot(housing, aes(x=BA, y=studres))+
  geom_jitter()+
  geom_hline(yintercept=0, color="red")+
  theme_bw()

grid.arrange(p1,p2,p3,p4, ncol=2)
```



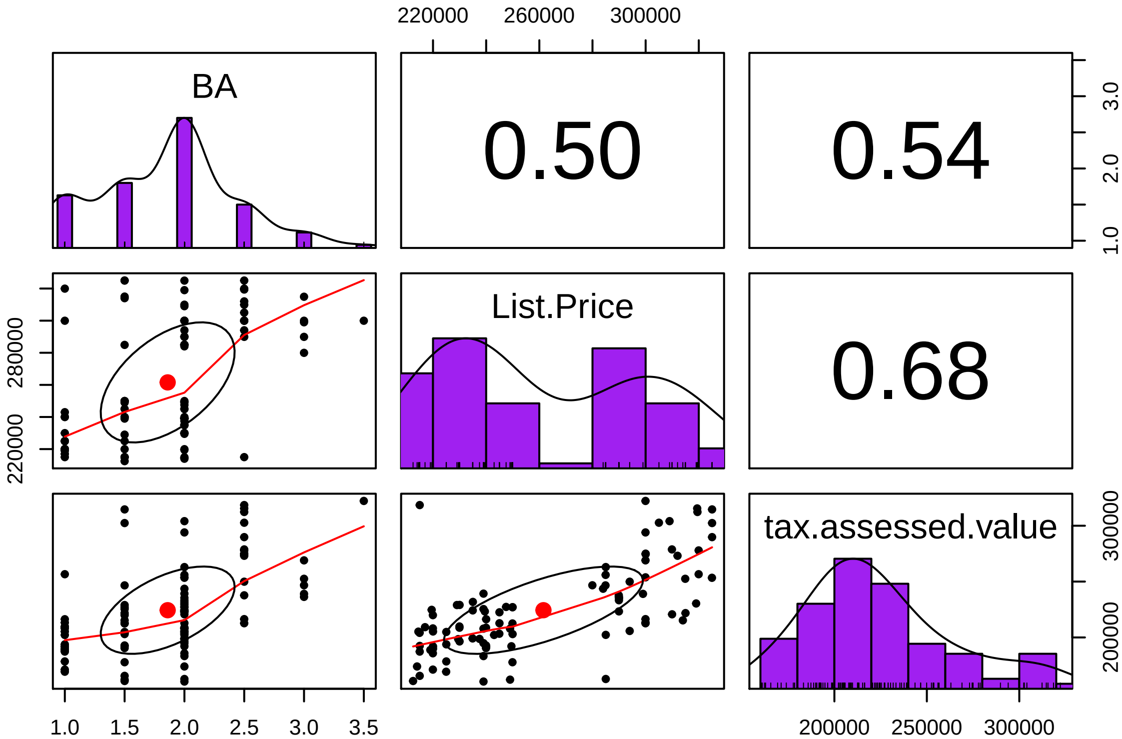
Multicollinearity Recheck

There is no evidence of multicollinearity due to low VIF values and correlation values less than .9 or greater than -.9.

```
#VIF Values
ols_vif_tol(housing_fit_best_outlier)
```

```
##               Variables Tolerance      VIF
## 1            List.Price 0.4691955 2.131308
## 2    tax.assessed.value 0.4758906 2.101323
## 3 remodeled.kitchenyes 0.8926673 1.120238
## 4                    BA 0.6723585 1.487302
```

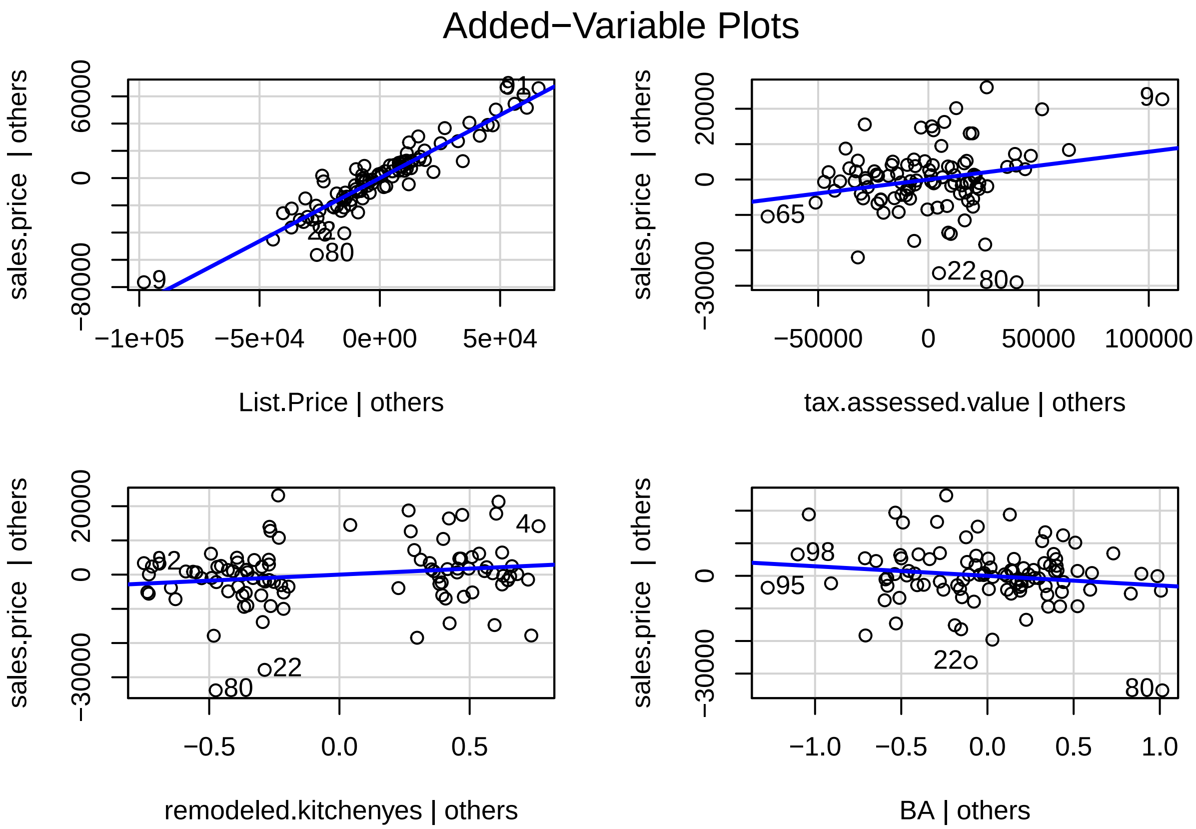
```
#Pairs Panels
terms<- names(housing_fit_best_outlier$coefficients[-1])
ind3<- which(names(housing)%in%terms)

pairs.panels(housing[ind3],
             method="pearson",
             hist.col = "purple",
             density=TRUE,
             ellipses = TRUE)
```



Check for Polynomial Terms by Added Variable Plots

We do not see evidence of a polynomial interaction occurring and therefore do not add a higher order term.

```
avPlots(housing_fit_best_outlier)
```



#
Marginal Plots Below we see marginal plot by each variable, categorized by whether the kitchen is remodeled or not.

```
##########################Marginal Plot of List Value (remodeled = yes)
DataNew=data.frame("(Intercept)"=rep(1,length=100),
"List.Price"=seq(min(housing$List.Price),max(housing$List.Price),length=100),
"tax.assessed.value"=rep(mean(housing$tax.assessed.value),length=100),
"remodeled.kitchen"=rep("yes",length=100),
"BA"=rep(mean(housing$BA),length=100))

MarginalData=data.frame(cbind(predict(housing_fit_best_outlier,newdata = DataNew), DataNew$List.Price))

names(MarginalData)=c("Predicted.Value","List.Price")

#Marginal Plot of List Value (remodeled = no)
DataNew2=data.frame("(Intercept)"=rep(1,length=100),
"List.Price"=seq(min(housing$List.Price),max(housing$List.Price),length=100),
"tax.assessed.value"=rep(mean(housing$tax.assessed.value),length=100),
"remodeled.kitchen"=rep("no",length=100),
"BA"=rep(mean(housing$BA),length=100))

MarginalData2=data.frame(cbind(predict(housing_fit_best_outlier,newdata = DataNew2), DataNew$List.Price

names(MarginalData2)=c("Predicted.Value","List.Price")

p11<- ggplot()+
geom_line(data=MarginalData,aes(x=List.Price,y=Predicted.Value), size=1.5,col='green')+
geom_line(data=MarginalData2, aes(x=List.Price,y=Predicted.Value), size=1.5,col='red')+
```

```r
theme_bw()+
labs(title="Marginal Plot of List Price by remodeled.kitchen")

####################################Tax.assessed.Value
DataNew=data.frame("(Intercept)"=rep(1,length=100),
"List.Price"=rep(mean(housing$List.Price),length=100),
"tax.assessed.value"=seq(min(housing$tax.assessed.value),max(housing$tax.assessed.value),length=100),
"remodeled.kitchen"=rep("yes",length=100),
"BA"=rep(mean(housing$BA),length=100))

MarginalData=data.frame(cbind(predict(housing_fit_best_outlier,newdata = DataNew), DataNew$tax.assessed

names(MarginalData)=c("Predicted.Value","tax.assessed.value")

#Marginal Plot of tax.assessed.Value (remodeled = no)
DataNew2=data.frame("(Intercept)"=rep(1,length=100),
"List.Price"=rep(mean(housing$List.Price),length=100),
"tax.assessed.value"=seq(min(housing$tax.assessed.value), max(housing$tax.assessed.value),length=100),
"remodeled.kitchen"=rep("no",length=100),
"BA"=rep(mean(housing$BA),length=100))

MarginalData2=data.frame(cbind(predict(housing_fit_best_outlier,newdata = DataNew2), DataNew$tax.assess

names(MarginalData2)=c("Predicted.Value","tax.assessed.value")

p12<- ggplot()+
geom_line(data=MarginalData,aes(x=tax.assessed.value,y=Predicted.Value), size=1.5,col='green')+
geom_line(data=MarginalData2, aes(x=tax.assessed.value,y=Predicted.Value), size=1.5,col='red')+
theme_bw()+
labs(title="Marginal Plot of Tax Assessed Value by remodeled.kitchen")


############Marginal Plot of Bathroom number remodeled = yes
DataNew=data.frame("(Intercept)"=rep(1,length=100),
"List.Price"=rep(mean(housing$List.Price),length=100),
"tax.assessed.value"=rep(mean(housing$tax.assessed.value),length=100),
"remodeled.kitchen"=rep("yes",length=100),
"BA"=seq(min(housing$BA),max(housing$BA),length=100))

MarginalData=data.frame(cbind(predict(housing_fit_best_outlier,newdata = DataNew), DataNew$BA))

names(MarginalData)=c("Predicted.Value","BA")

#Marginal Plot of List Value (remodeled = no)
DataNew2=data.frame("(Intercept)"=rep(1,length=100),
"List.Price"=rep(mean(housing$List.Price),length=100),
"tax.assessed.value"=rep(mean(housing$tax.assessed.value),length=100),
"remodeled.kitchen"=rep("no",length=100),
"BA"=seq(min(housing$BA),max(housing$BA),length=100))

MarginalData2=data.frame(cbind(predict(housing_fit_best_outlier,newdata = DataNew2), DataNew$BA))

names(MarginalData2)=c("Predicted.Value","BA")
```

```
p13<- ggplot()+
geom_line(data=MarginalData,aes(x=BA,y=Predicted.Value), size=1.5,col='green')+
geom_line(data=MarginalData2, aes(x=BA,y=Predicted.Value), size=1.5,col='red')+
theme_bw()+
labs(title="Marginal Plot of Bathrooms by remodeled.kitchen")

p11
```


```
p12
```


Marginal Plot of Tax Assessed Value by remodeled.kitchen
270000
265000
260000
255000
Predicted.Value
200000
250000
300000
tax.assessed.value

```
p13
```

Interpretation

```
summary<-summary(housing_fit_best_outlier)
summary$adj.r.squared
```

```
## [1] 0.9407928
```

```
round(summary$coefficients,4)
```

```
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          5421.2022  6683.8076  0.8111   0.4193
## List.Price              0.9260     0.0358 25.8365   0.0000
## tax.assessed.value      0.0784     0.0341  2.3019   0.0235
## remodeled.kitchenyes 3513.1670  1944.9836  1.8063   0.0740
## BA                  -2940.0966  2001.1842 -1.4692   0.1451
```

- Significant F-statistic of 394.3
- Significant p-value of <2.2 e-16
- Adjusted R2 of .941, meaning 94.1% of the variance in sales price is due to our model.

- On average, all else equal, a one unit increase in List Price increases the Sales price by .926
- On average, all else equal, a one dollar increase in tax.assessed.value increases sales price by .78
- On average, a remodeled kitchen yields an added 3513 dollars to the sales price compared to a non-remodeled kitchen.
- On average, all else equal, an increase in 1 full bath leads to a decrease in Sales price by 2940.10
- No evidence of multicollinearity
- One removed leverage point (X1==84), due it is producing NaN fitted value (only two-story home in data)
- Overall this model is robust and a good predictor of future data points.

New Prediction in Range

```
DataNew=data.frame("(Intercept)"=1,
"List.Price"= 264900,
"tax.assessed.value"= 202400,
"remodeled.kitchen"="no",
"BA"=2)

predict(housing_fit_best_outlier, DataNew)
```

```
##        1
## 260715.8
```

New Prediction Out of Range

```
DataNew=data.frame("(Intercept)"=1,
"List.Price"= 349500,
"tax.assessed.value"= 260400,
"remodeled.kitchen"="no",
"BA"=2)

predict(housing_fit_best_outlier, DataNew)
```

```
##        1
## 343605.4
```

Data Analytics

In the section below, I will use the housing dataset to further explore the MSE predictive ability of the MLR model above (sales.price ~ List.Price + Tax.Assessed.Value + Remodeled.Kitchen + Bathrooms) and three alternative models: Principal Components, Lasso, and Ridge. I will provide justification on why the alternative models of interest were chosen.

The primary means for assessing the model's "consistency" will be the use of an iterated 5 fold cross validation procedure. This will split either our training data or test data into 5 folds, using one of the folds as the test data. The other 4 folds will be used to train our model of choice.

Set side training and test data

Our first step below will be to set aside an 80/20 split of training data to test data. This allows us to train our model on a subset of data, and then test that model on unseen data to assess predictive ability.

```
set.seed(42)

#Revert back to baseline housing file - outliers
housing<-housing%>%
  dplyr::select(BR, BA, RM, Fin.SF, Yr.Blt, List.Price, List.Date, List.Month, sales.price, tax.assesse

trainingindex<-sample(1:nrow(housing), .8*nrow(housing))

housing_train<- housing[trainingindex,]
housing_test<- housing[-trainingindex,]
```

Multiple Linear Regression on Training Data

The multiple linear regression model selected above in our previous course assignment was found to be dominated by the variable List.Price. This makes intuitive sense - homes that sell for higher prices will be listed at higher prices on the market. However this has some shortcomings that we will explore with the least squares regression method - namely, that it is inflexible to situations where a house could be listed significantly different than the eventual sales price. in this case, predicting sales price will in most cases only require reference to the list price. This is not always reality.

The data shown below are the first 10 MSE value of 5 fold cross validation on the housing training data set, shown as training MSE and validation MSE. Training MSE is calculated on the 4 training folds, and the validation MSE is calculated on the single holdout fold.

We will compare these metrics further below when evaluated on the full data set.

```
mlr.k.fold.validator <- function(df, K, iter) {

  # this function calculates the errors of a single fold using the fold as the holdout data
  fold.errors <- function(df, holdout.indices) {
    train.data <- df[-holdout.indices, ]
    holdout.data <- df[holdout.indices, ]

    #Ridge Model Fit to training data
    fit <- lm(formula = sales.price ~ List.Price + tax.assessed.value +
            remodeled.kitchen + BA, data = train.data)
```

```r
    #Training Data MSE per fold
    train.predict <- predict(fit, train.data)
    train.error<- mean((train.data$sales.price-train.predict)^2)

    #Holdout Data MSE per fold
    holdout.predict <- predict(fit, holdout.data)
    holdout.error<- mean((holdout.data$sales.price-holdout.predict)^2)

    tibble(mlr.train.error = train.error, mlr.valid.error = holdout.error)

  }
errors<-tibble()
#Add an outer for loop which iterates iter times
  for (j in 1:iter) {
  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))

  folds <- cut(indices, breaks = K, labels = F)
  # set error to 0 to begin accumulation of fold error rates

  # iterate on the number of folds

  for (i in 1:K) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.errors <- fold.errors(df, holdout.indices)
    errors <- errors %>%
      bind_rows(folded.errors)
  }
  }
errors
}


mlr.train<- mlr.k.fold.validator(housing_train, 5, 100)
mlr.train
```

```
## # A tibble: 500 x 2
##    mlr.train.error mlr.valid.error
##              <dbl>           <dbl>
##  1       95172775.       82097985.
##  2       79210908.      150522062.
##  3       91920868.      100209885.
##  4       80783875.      168070100.
##  5       94156414.       83101920.
##  6       92586640.       94584142.
##  7      106325665.       31203450.
##  8       98815926.       61300230.
##  9       70439588.      173393934.
## 10       79752709.      155013061.
## # ... with 490 more rows
```

Ridge Regression on Training Data

The first alternative model chosen was the ridge regression model. Ridge regression shrinks our coefficients to zero using a tuned shrinking parameter, evaluated by the cv.glmnet function. This is the "penalty" metric that determines the "price" we pay for high coefficients. This lambda value was determined to be 337.0816.

This model is of interest due to the singlular large coefficient, kitchen.remodeledyes (3513). This compares to the coefficient for list price (.926), tax.assessed.value (.078), and BA (-.00294). I was interested to see whether having a single large coefficient would have a negative impact on predictive ability.

We see below the output of the first 10 results of 5 fold cross validation of our ridge regression model on the housing_test dataset. This will later be compared to the MSE on our test data set.

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.0-2
```

```
# X model matrix and Y response
train.x<- model.matrix(sales.price ~.,
                        data=housing_train)[,-1]
test.x<- model.matrix(sales.price ~ .,
                        data=housing_test)[,-1]

#Grid of possible lambdas
grid<- 10^seq(10,-2, length=100)

#Fit ridge model
housing_ridge<- glmnet(train.x,housing_train$sales.price,alpha=0, lambda=grid)

#Cross Validation function, output lambda associated with smallest train error
cv.out<- cv.glmnet(train.x,housing_train$sales.price,alpha=0)
bestlam<- cv.out$lambda.min
bestlam
```

```
## [1] 3566.274
```

```
ridge.k.fold.validator <- function(df, K, iter) {

  # this function calculates the errors of a single fold using the fold as the holdout data
  fold.errors <- function(df, holdout.indices) {
    train.data <- df[-holdout.indices, ]
    holdout.data <- df[holdout.indices, ]

    #Set Train and Test Model Matrix for Ridge Regression
```

```
    train.x<- model.matrix(sales.price ~ .,
                          data=train.data)[,-1]
    holdout.x<- model.matrix(sales.price ~ .,
                          data=holdout.data)[,-1]
    #Ridge Model Fit to training data
    fit <- glmnet(train.x, train.data$sales.price, alpha=0, lambda=bestlam)

    #Training Data MSE per fold
    train.predict <- predict(fit, train.x, s=bestlam)
    train.error<- mean((train.data$sales.price-train.predict)^2)

    #Holdout Data MSE per fold
    holdout.predict <- predict(fit, newx=holdout.x, s=bestlam)
    holdout.error<- mean((holdout.data$sales.price-holdout.predict)^2)

    tibble(ridge.train.error = train.error, ridge.valid.error = holdout.error)

  }
errors<-tibble()
#Add an outer for loop which iterates iter times
  for (j in 1:iter) {
  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))

  folds <- cut(indices, breaks = K, labels = F)
  # set error to 0 to begin accumulation of fold error rates

  # iterate on the number of folds

  for (i in 1:K) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.errors <- fold.errors(df, holdout.indices)
    errors <- errors %>%
      bind_rows(folded.errors)
  }
  }
errors
}

ridge.train<- ridge.k.fold.validator(housing_train, 5, 100)
ridge.train
```

```
## # A tibble: 500 x 2
##    ridge.train.error ridge.valid.error
##                <dbl>             <dbl>
##  1         59389871.        249281013.
##  2         71271142.        130425790.
##  3         55277620.        299928197.
##  4         68529660.        124978177.
##  5         44119638.        338867938.
##  6         72496087.        142070677.
##  7         55796943.        308811599.
##  8         70729571.         80076554.
```

```
## 9           41422839.        411861998.
## 10          60479857.        144281153.
## # ... with 490 more rows
```

Lasso Regression on Training Data

Our next model of interest is the lasso regression model. Lasso regression is similar to the ridge regression in that it shrinks our variable coefficient sizes, some all the way to zero. This means that a subset of predictors will be included in our model of the lasso regression. In this way, lasso regression operates as similar to a variable selection procedure and regression model combined.

I was interested in seeing how a lasso model would predict our data, due to the dominance of list.price as a regressor. If list.price were removed or shrinked in any of models, it may alter our predictive ability. Using the cv.glmnet function, we find the tuned lambda shrinkage parameter of 10.72, and use that value of the shrinkage parameter in our evaluation model on the training and validation data sets. We see the first 10 results below. The evaluation of our model on the test data will be included below.

```r
#Fit Lasso model
housing_lasso<- glmnet(train.x, housing_train$sales.price, alpha=1, lambda=grid)
summary_lasso<- summary(housing_lasso)

#Perform cross validation and output lambda that minimizes training error
cv.lasso<- cv.glmnet(train.x,housing_train$sales.price, alpha=1, lambda=grid)
bestlam.lasso<-cv.lasso$lambda.min
bestlam.lasso
```

```
## [1] 2848.036
```

```r
lasso.k.fold.validator <- function(df, K, iter) {

  # this function calculates the errors of a single fold using the fold as the holdout data
  fold.errors <- function(df, holdout.indices) {
    train.data <- df[-holdout.indices, ]
    holdout.data <- df[holdout.indices, ]

    #Set Train and Test Model Matrix for Lasso Regression
    train.x<- model.matrix(sales.price ~ .,
                         data=train.data)[,-1]
    holdout.x<- model.matrix(sales.price ~ .,
                         data=holdout.data)[,-1]
    #Ridge Model Fit to training data
    fit <- glmnet(train.x, train.data$sales.price, alpha=1, lambda=bestlam.lasso)

    #Training Data MSE per fold
    train.predict <- predict(fit, train.x, s=bestlam.lasso)
    train.error<- mean((train.data$sales.price-train.predict)^2)

    #Holdout Data MSE per fold
    holdout.predict <- predict(fit, newx=holdout.x, s=bestlam)
    holdout.error<- mean((holdout.data$sales.price-holdout.predict)^2)

    tibble(lasso.train.error = train.error, lasso.valid.error = holdout.error)
```

```
  }
errors<-tibble()
#Add an outer for loop which iterates iter times
  for (j in 1:iter) {
  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))

  folds <- cut(indices, breaks = K, labels = F)
  # set error to 0 to begin accumulation of fold error rates

  # iterate on the number of folds

  for (i in 1:K) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.errors <- fold.errors(df, holdout.indices)
    errors <- errors %>%
      bind_rows(folded.errors)
  }
  }
errors
}

lasso.train<-lasso.k.fold.validator(housing_train, 5, 100)
lasso.train
```

```
## # A tibble: 500 x 2
##    lasso.train.error lasso.valid.error
##                <dbl>             <dbl>
##  1         97947624.        115824801.
##  2        104237867.        144523057.
##  3        115304375.         67325599.
##  4         95841557.        115992885.
##  5        103083492.        128999283.
##  6         95832598.        148960591.
##  7        112871940.         68205441.
##  8         78774195.        208574391.
##  9        112667571.         71700310.
## 10        119170391.         67479147.
## # ... with 490 more rows
```

PCR Fit on Training Data

Our last alternative model assessed is the Principal Components Regression. PCR is a model in which the variables in our data are not used, instead the partial components of the interaction of our regressors are chosen. These chosen partial components are the components that explain the most variability in our data.

In this case, I wanted to test whether the results of the PCR model would compare to the MLR model, as the principal components may be dominated by List.Price. This would be similar to the MLR model.

Our number of principal components was found to be 2 - this makes intuitive sense due to the dominance of one variable in particular in our MLR model. Below, we see the first 10 results of the PCR model MSE using 5 fold cross validation evaluated on the housing_train subset. The evaluation on the housing_test subset will be explored further below.

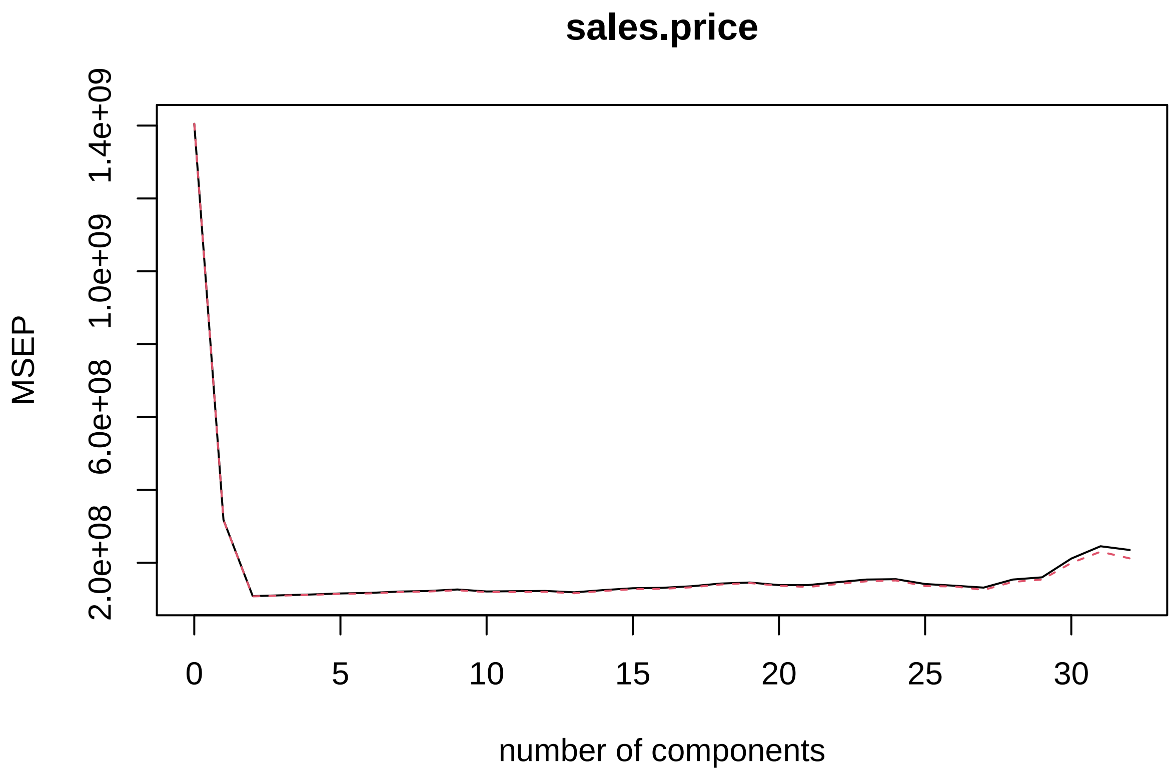
```r
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
housing_pcr<- pcr(sales.price~., data=housing_train, validation="CV")
summary(housing_pcr)
```

```
## Data:    X dimension: 80 32
##  Y dimension: 80 1
## Fit method: svdpc
## Number of components considered: 32
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           37483    17829    10416    10525    10628    10761    10830
## adjCV        37483    17814    10383    10486    10582    10709    10771
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       10999    11072    11258     11010     11044     11076     10903
## adjCV    10932    11005    11167     10932     10935     10973     10789
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        11184     11407     11459     11641     11956     12074     11783
## adjCV     11080     11289     11335     11522     11849     12018     11707
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        11780     12112     12409     12450     11903     11706     11481
## adjCV     11561     11903     12216     12305     11667     11620     11204
##        28 comps  29 comps  30 comps  31 comps  32 comps
## CV        12410     12651     14546     15664     15335
## adjCV     12149     12405     14125     15179     14560
##
## TRAINING: % variance explained
##              1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X              81.36    99.99   100.00   100.00   100.00   100.00   100.00
## sales.price    77.36    92.99    93.02    93.02    93.02    93.11    93.11
##              8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X             100.00   100.00    100.00    100.00    100.00    100.00       100
## sales.price    93.11    93.33     93.44     93.71     93.78     93.89        94
##              15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## X              100.00    100.00    100.00    100.00    100.00    100.00
## sales.price     94.05     94.13     94.14     94.14     94.17     94.39
##              21 comps  22 comps  23 comps  24 comps  25 comps  26 comps
## X              100.00     100.0     100.0    100.00    100.00     100.0
## sales.price     94.76      94.8      94.8     94.82     95.39      95.4
##              27 comps  28 comps  29 comps  30 comps  31 comps  32 comps
## X              100.00    100.00    100.00    100.00    100.00    100.00
## sales.price     95.86     95.86     95.94     96.39     96.48     96.62
```

```r
validationplot(housing_pcr, val.type="MSEP")
```




```r
#Optimal Number of Components found to be 2
```

```r
pcr.k.fold.validator <- function(df, K, iter) {

  # this function calculates the errors of a single fold using the fold as the holdout data
  fold.errors <- function(df, holdout.indices) {
    train.data <- df[-holdout.indices, ]
    holdout.data <- df[holdout.indices, ]

    #PCR Model Fit to training data
    fit<- pcr(sales.price~., data=train.data)

    #Training Data MSE per fold
    train.predict <- predict(fit, train.data, ncomp=2)
    train.error<- mean((train.data$sales.price-train.predict)^2)

    #Holdout Data MSE per fold
    holdout.predict <- predict(fit, newdata=holdout.data, ncomp=2)
    holdout.error<- mean((holdout.data$sales.price-holdout.predict)^2)

    tibble(pcr.train.error = train.error, pcr.valid.error = holdout.error)

  }
errors<-tibble()
#Add an outer for loop which iterates iter times
  for (j in 1:iter) {
```

```r
  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))

  folds <- cut(indices, breaks = K, labels = F)
  # set error to 0 to begin accumulation of fold error rates

  # iterate on the number of folds

  for (i in 1:K) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.errors <- fold.errors(df, holdout.indices)
    errors <- errors %>%
      bind_rows(folded.errors)
  }
  }
errors
}

pcr.train<-pcr.k.fold.validator(housing_train, 5, 100)
pcr.train
```

```
## # A tibble: 500 x 2
##    pcr.train.error pcr.valid.error
##              <dbl>           <dbl>
##  1       83629681.      165634429.
##  2       96465949.       98153833.
##  3      102560787.       75429655.
##  4       97303732.      112481522.
##  5       92072581.      112991385.
##  6       64360408.      227354644.
##  7       97670978.       92663246.
##  8      103628552.       66490818.
##  9      103437875.       67142312.
## 10      109093070.       46472041.
## # ... with 490 more rows
```

MLR, Ridge, Lasso, and PCR Regression Fit on Training Data, Tested on test data

The boxplots below explore the MSEs of our model when used to predict previously unseen test data. My analysis is below as well.

```r
mlr.test<- mlr.k.fold.validator(housing,5,100)%>%
  dplyr::rename(mlr.test.error = mlr.valid.error)

ridge.test<- ridge.k.fold.validator(housing,5,100)%>%
  dplyr::rename(ridge.test.error = ridge.valid.error)

lasso.test<- lasso.k.fold.validator(housing,5,100)%>%
  dplyr::rename(lasso.test.error = lasso.valid.error)
```

```
pcr.test<- pcr.k.fold.validator(housing,5,100)%>%
  dplyr::rename(pcr.test.error = pcr.valid.error)
```

Boxplots

```
p1<- mlr.train%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method,MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
  labs(x="Value", y="MSE", title = "Linear Regression Errors Train")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))

p2<- mlr.test%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method, MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
  labs(x="Value", y="MSE", title = "Linear Regression Errors Test")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))


p3<- ridge.train%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method,MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
  labs(x="Value", y="MSE", title = "Ridge Regression Errors Train")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))

p4<- ridge.test%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method,MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
  labs(x="Value", y="MSE", title = "Ridge Regression Errors Test")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))

p5<- lasso.train%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method,MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
```

```
  labs(x="Value", y="MSE", title = "Lasso Regression Errors Train")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))

p6<- lasso.test%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method,MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
  labs(x="Value", y="MSE", title = "Lasso Regression Errors Test")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))

p7<- pcr.train%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method,MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
  labs(x="Value", y="MSE", title = "PCR Regression Errors Train")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))

p8<- pcr.test%>%
  gather(Method, MSE)%>%
  ggplot()+
  geom_boxplot(aes(x=reorder(Method,MSE, FUN=median), y=MSE, fill= reorder(Method, MSE, FUN=median)))+
  theme_bw()+
  labs(x="Value", y="MSE", title = "PCR Regression Errors Test")+
  theme(legend.title = element_blank())+
  scale_y_continuous(breaks=c(1e8,2e8,3e8))+
  coord_cartesian(ylim=c(0,3e8))
```

Our first set of side-by-side box plots explores the predictive ability of our initial Linear Regression model. We see a baseline value for the median MSE, and a baseline value for the variation, to compare other models against.

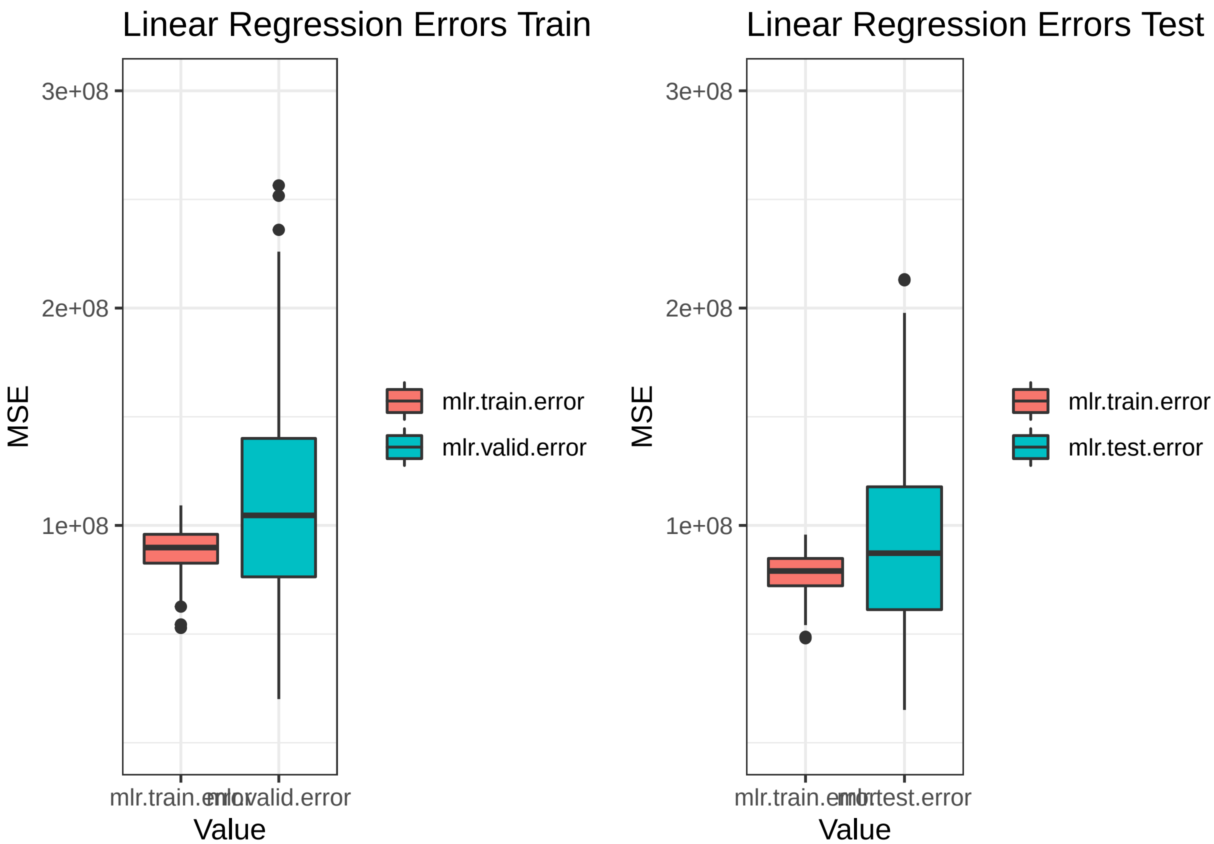
Our second set of boxplots is for the Ridge Regression model. Using ridge regression, we see an overfitting situation, where our median MSE and variation is low on our training data, but extremely high on our test data. I prefer the MLR model to the ridge regression model.

The next set of boxplots to explore is for the Lasso Regression. We see our median and variation within MSE is very similar to that of our MLR model. This indicates that the variables included in our lasso model are likely to be similar to the 4 variables within our MLR model (list.price, tax.assessed.value, remodeled.kitchen, and BA). Both the lasso model and the MLR model are near-equal models.

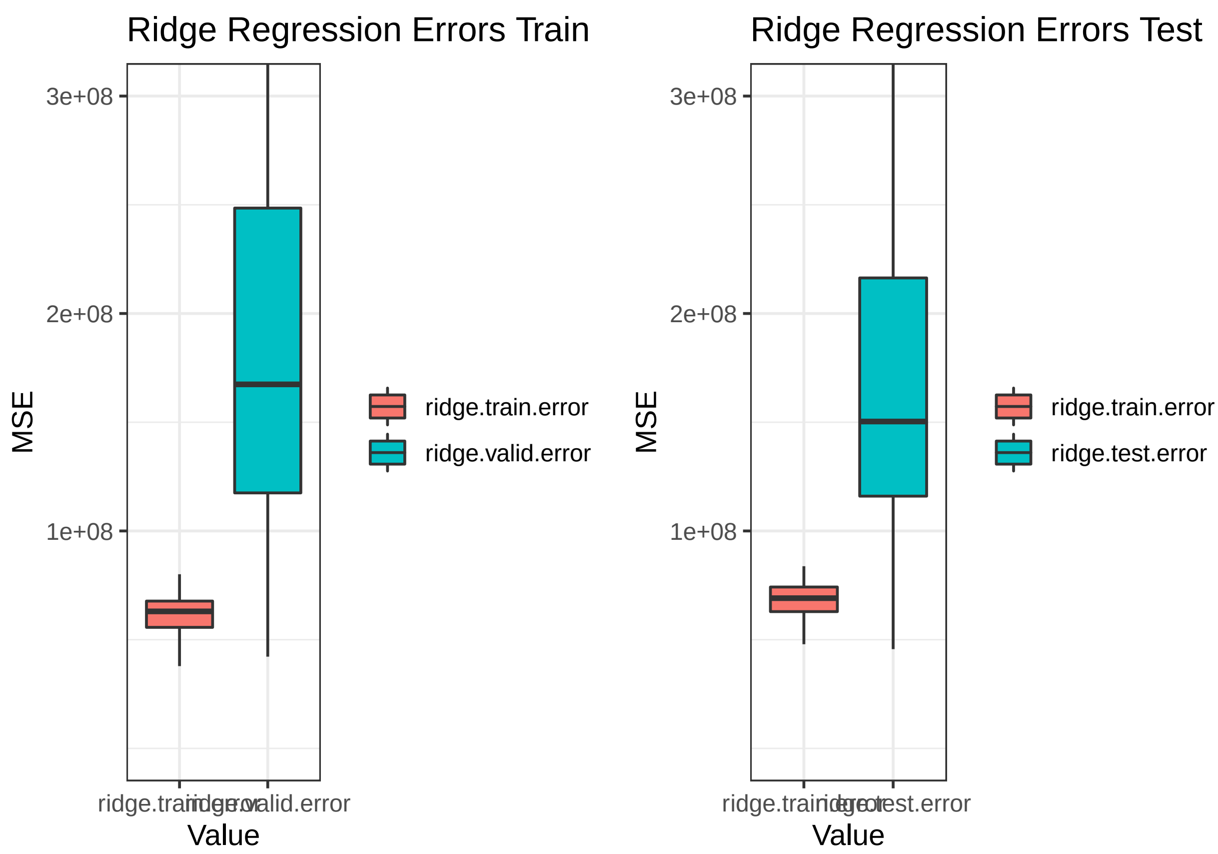
The last model to explore is Principal Components Regression model. We find similar mean and deviation to our MLR and Lasso regression models.

In summary, given a choice, I would exclude the ridge regression model and proceed with either a linear model, a lasso model, or a PCR model given the data set.

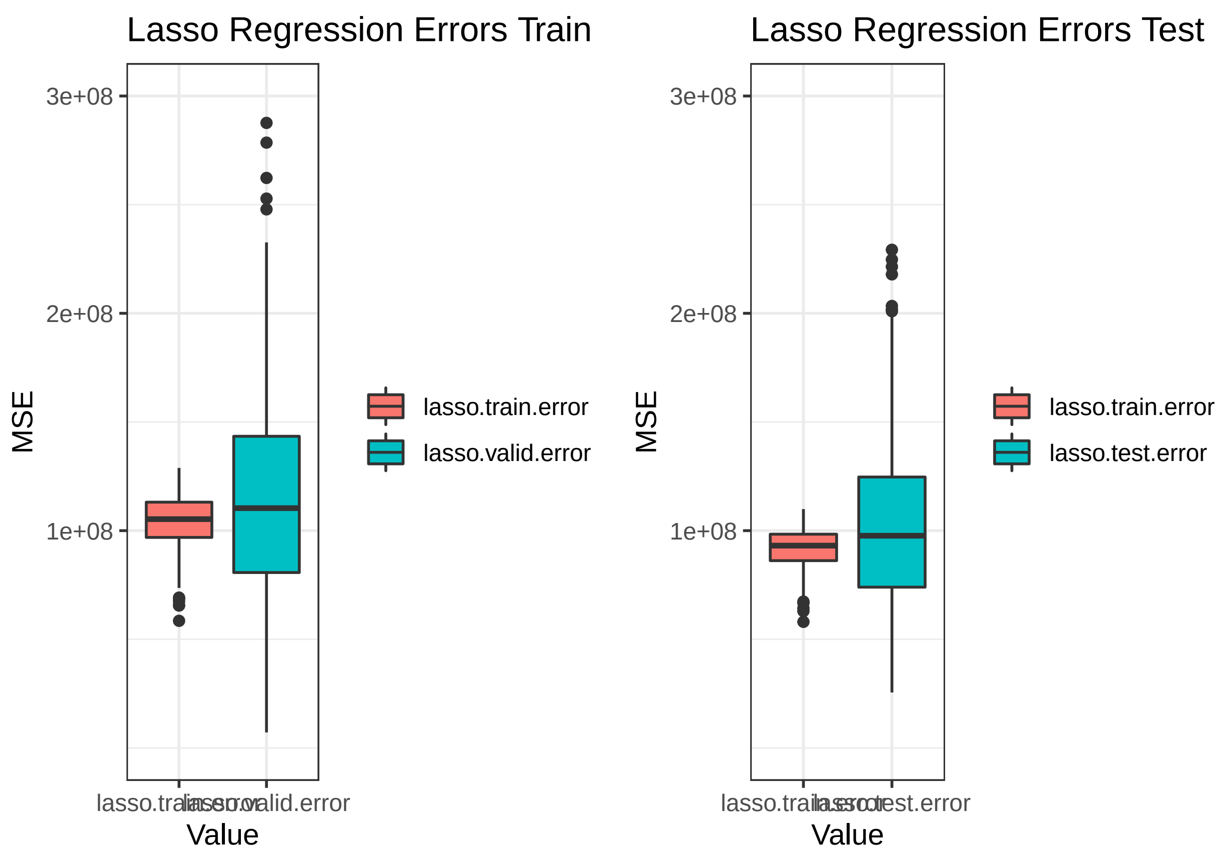
```r
grid.arrange(p1,p2,nrow=1)
```



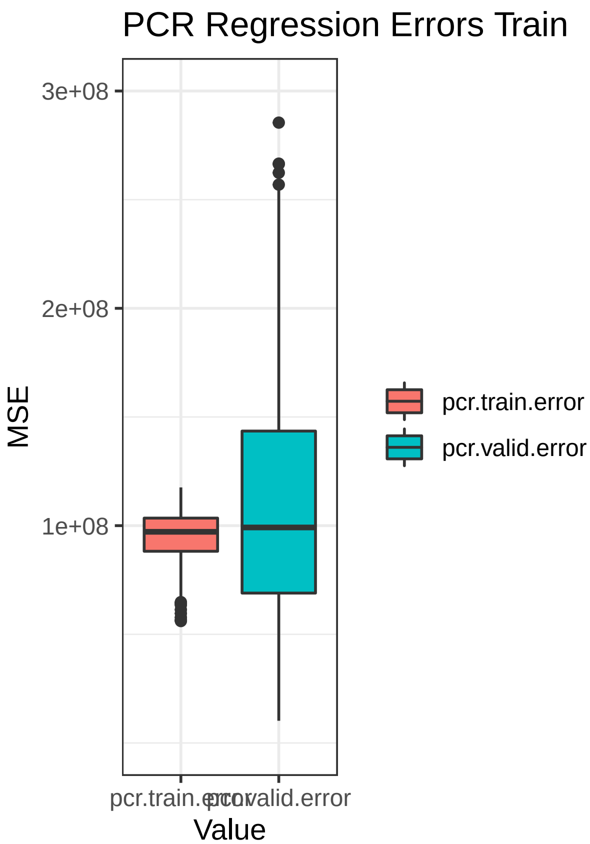

```r
grid.arrange(p3,p4,nrow=1)
```

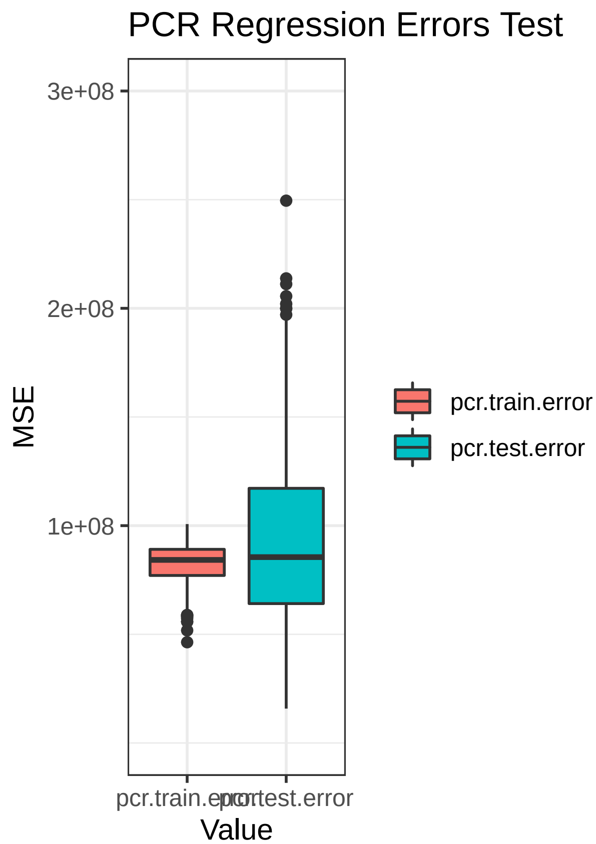



```
grid.arrange(p5,p6,nrow=1)
```




```
grid.arrange(p7,p8,nrow=1)
```





Summary

Compared to the Advanced Methods of Regression class, this class has focused on more brute force modeling approaches, where all of our data can be included in our potential model, and we rely on our computing power to find the right solution. In the Regression class, much of the class is spent on refining and evaluating how a model fits data, to test underlying assumptions. While underlying assumptions are within this class, the procedures for testing are more focused n predictive ability and not assumption validation.

We've found in our analysis above that median and variance is not just a factor in our data set, but it is also a metric to be used in evaluating the predictive ability of our model on our data set. In other words, randomness exists in the MSE evaluation as we test predictive ability.

Also in this class, we constantly are sampling and subsampling our data to create data sets to evaluate - and then shuffling those data sets to ensure that we evaluate our models from every possible angle using all available data. This is the 5 fold cross validation technique. In the other class, our results from our model are what they are - there is no constant sampling and subsampling.

Lastly, least squares regression we've learned is a great technique, however it requires adherence to normality and variance assumptions. In this class, predictive ability is our goal.