Final Project: Housing Market of Taiwan 2012-13

Project By: Kristian Abad, Steven Truong, & Keshav Khanna

Abstract

Our group decided to observe a data set relating to real estate valuation in the Sindian District of Taipei City, Taiwan. The data set included attributes such as the transaction date, the house age, the distance to the nearest mass transportation (mrt), the number of convenience stores within the area, and the geographic coordinate such as latitude and longitude. The response of our data set is the house price per unit area. Since a denser population and higher demand of housing in certain areas will cause house prices to increase, we wished to find out whether house buyers in Taiwan in the year of 2012 and 2013 valued being close to mass transportation, being surrounded by many convenience stores, or the age of the house when considered buying a home. In addition, we will observe whether the seasons of Winter, Spring, Summer, and Fall will have any effect on the house price throughout the year.

Problem and Motivation

Buying a home is considered to be a huge milestone into adulthood. It is also a huge commitment as houses are generally expensive and mortgages in Taiwan takes on an average of 25 years to pay off. Especially during the Covid-19 pandemic, the housing market in the United States is thriving. Home buyers are competing and overpaying for houses that do not match their valuations. In the interest of home buyers, we wished to generally determine which aspects of accessibility are most important when considering a real estate true valuation before the pandemic. The main factors that determine values of homes are the historical sale price, the neighborhood, the market, the size and appeal, age and condition, and nearby features. Our data set provided data that relates to the factors of house age and nearby features. We will observe the factor of the nearby features to see if there is any relationship to the price. We did not take into consideration the house age because its coefficient of determination was quite small compared to the distance to the nearest MRT and number of convenience stores. By learning about the effects of these factors that cause housing prices to increase or decrease, home buyers can make the decision of whether their future home matches its valuation according to our data.

Questions of Interest:

1. Do consumers value being closer to MRT, having more convenience stores when considering buying a home?
2. In which seasons are housing prices the lowest and highest?

Data

The dataset comes from the UCI Machine Learning Repository and the variables of interest are X1.transaction.date, X2.house.age, X3.distance.to.the.nearest.MRT.station, X4.number.of.convenience.stores, and Y.house.price.of.unit.area.

Regression Methods:

1. For the first question, a simple linear regression model will be made such that the price per area is regressed by the distance to the nearest MRT station and the number of surrounding convenience stores in two separate models and evaluate how much each predictor explains price. From there, a model including both predictors and submodels will be compared to further determine which one contributes more to price.
2. A simple linear regression will be made such that price per area is regressed by transaction date after encoded into 4 categorical season variables.

Regression Analysis:

Question 1:

Important Details of the Analysis:

Given the linear models:

Price = B0 + B2(Number of convenience store)
Price = B01 + B1(Distance to nearest MRT)

We have the following tests:
H0: B1 = 0, HA: B1 != 0
H0: B2 = 0, HA: B2 != 0

MRT and Convenience store simple linear model and multiple linear regression summaries:

```
Call:
lm(formula = pricePerArea ~ mrt)

Residuals:
    Min      1Q  Median      3Q     Max
-35.396  -6.007  -1.195   4.831  73.483

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 45.8514271  0.6526105   70.26   <2e-16 ***
mrt         -0.0072621  0.0003925  -18.50   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.07 on 412 degrees of freedom
Multiple R-squared:  0.4538,    Adjusted R-squared:  0.4524
F-statistic: 342.2 on 1 and 412 DF,  p-value: < 2.2e-16
```

```
Call:
lm(formula = pricePerArea ~ numConvStores)

Residuals:
    Min      1Q  Median      3Q     Max
-35.407  -7.341  -1.788   5.984  87.681

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    27.1811     0.9419   28.86   <2e-16 ***
numConvStores   2.6377     0.1868   14.12   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.18 on 412 degrees of freedom
Multiple R-squared:  0.326,     Adjusted R-squared:  0.3244
F-statistic: 199.3 on 1 and 412 DF,  p-value: < 2.2e-16
```

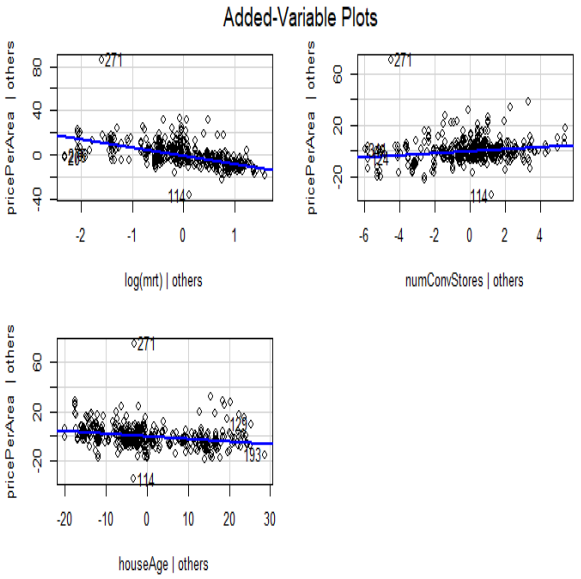
```
Call:
lm(formula = pricePerArea ~ log(mrt) + numConvStores)

Residuals:
    Min      1Q  Median      3Q     Max
-34.783  -5.106  -0.756   3.462  74.582

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    85.8141     4.2006   20.43  < 2e-16 ***
log(mrt)       -7.8611     0.5536  -14.20  < 2e-16 ***
numConvStores   0.5891     0.2104    2.80  0.00536 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.171 on 411 degrees of freedom
Multiple R-squared:  0.5479,    Adjusted R-squared:  0.5457
F-statistic:   249 on 2 and 411 DF,  p-value: < 2.2e-16
```

Based off the fact that the p-values for each model is well below any given threshold, we can reject both of the null hypotheses as well as conclude that each predictor helps explain the variation of price per area. How much each predictor helps explain price per area in their respective models is now in question. It would seem that consumers value distance to MRT more than number of convenience stores due to the fact that MRT explains about 45.4% of the variation of price in a linear relationship compared to 32.6%, their R squared values, but based off these values we see that these models are not a good fit since their R squared values are closer to zero. To get a more wholistic answer, a model with more predictors is needed.

Referring to **D1a**, we know that mrt requires transformation. What about the a model with more predictors? Here We will make a scatterplot matrix to observe the relationship of the following model:

Price = B0 + B1log(Distance to MRT station) + B2(number of Convenience stores)

In the rightmost summary table above, we see that the p value of log(mrt) and houseAge is very small, such that they are significant predictors of the price per area. We see that numConvStores has a larger p value, but it is still relatively small. The number of convenience store would be a significant predictor of price per area at a 0.0005 level of significance.



The added variable plots also indicate that these predictors explains the change in the response. Now that we've looked at simple linear models, let's take a look at what a model with more predictors looks like using mrt, convenience stores, and house age as predictors.

From the summary table, we can create the regression equation:

Price per Area = 85.8141 - 7.8611log(mrt) + 0.5891(numConvStores)

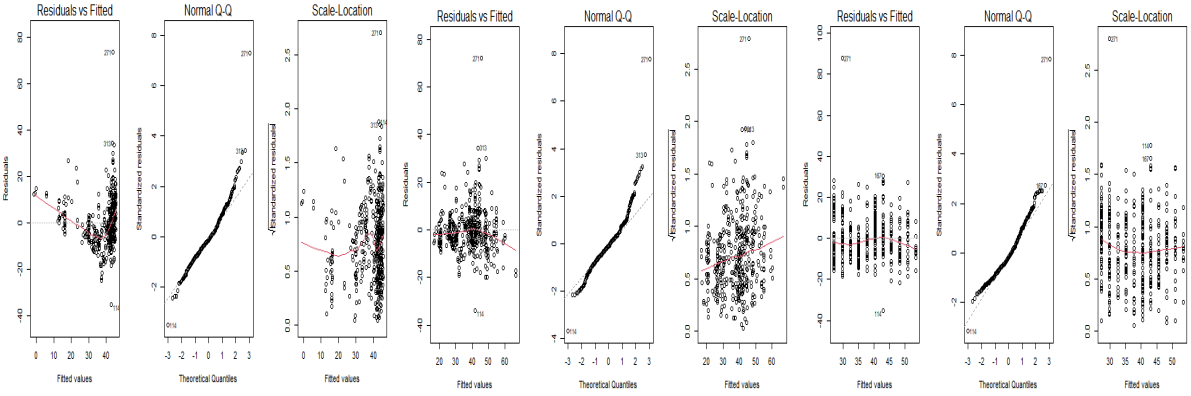
The equation indicates that as the distance to the nearest mrt increase by 1 unit, the price per area decrease by 7.8611

The equation indicates that as the number of convenience store increase by 1 unit, the price per area increase by 0.5891

We see that increasing the distance to the nearest mrt by 1 unit affects the price per area the most and that it also has the lowest p value. Therefore, we conclude that the consumers valued distance of the nearest mrt the most rather than the number of convenience and house when considering buying a house.

Diagnostic Checks:

(D1a)——————————————————————(D1b)——————————————————-(D2)



For the MRT variable, we can see that the normality assumption holds but linearity and the constant variance assumptions do not (**D1a**). However, after the transformation the diagnostics look better in **D1b**.

Based off the diagnostics for convenience stores in **D2**, we concluded that the assumptions of linearity, constant variance, and normality hold true in the model in which the number of convenience stores was the only predictor.

Interpretation:

We can conclude that the distance to the nearest mrt have a greater impact on the housing prices than the number of convenience store. This means house buyers value being closer to a mrt than the number of convenience store.

Question 2:

Important Details of the Analysis:

Given the linear models:

Price = B0 + B1(Spring) + B2(Summer) + B3(Winter)

We have the following tests:

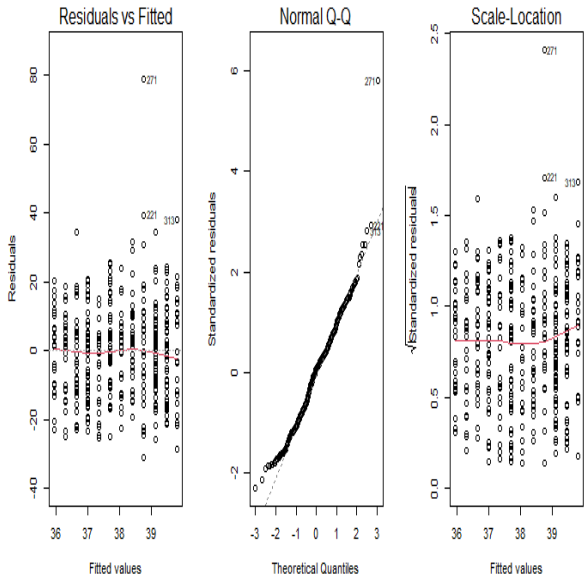
H0: B1 = B2 = B3 = 0, HA: Bi != 0 for i = 1, 2, 3

```
Call:
lm(formula = pricePerArea ~ seasonDate, data = real_estate)

Residuals:
    Min      1Q  Median      3Q     Max
-31.409 -10.103   0.571   8.527  78.491

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       35.684      2.447  14.580   <2e-16 ***
seasonDateSpring   3.325      2.716   1.224    0.222
seasonDateSummer   2.165      2.883   0.751    0.453
seasonDateWinter   1.964      2.663   0.738    0.461
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.63 on 410 degrees of freedom
Multiple R-squared:  0.004255,  Adjusted R-squared:  -0.003031
F-statistic: 0.584 on 3 and 410 DF,  p-value: 0.6258
```

Since p values of all our thresholds are well above the threshold in our parallel model with only the seasons as predictors, we conclude that the purchases of houses in different seasons doesn't affect the price of the house.

Diagnostic Checks:

(D3)



Based off the plots, the assumptions of linearity, normality, and constant variance hold for the linear model with transaction date as a predictor.

Summary of model after encoding season:

```
Call:
lm(formula = pricePerArea ~ seasonDate, data = real_estate)

Residuals:
    Min      1Q  Median      3Q     Max
-31.409 -10.103   0.571   8.527  78.491

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       35.684      2.447  14.580   <2e-16 ***
seasonDateSpring   3.325      2.716   1.224    0.222
seasonDateSummer   2.165      2.883   0.751    0.453
seasonDateWinter   1.964      2.663   0.738    0.461
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.63 on 410 degrees of freedom
Multiple R-squared:  0.004255,  Adjusted R-squared:  -0.003031
F-statistic: 0.584 on 3 and 410 DF,  p-value: 0.6258
```

After encoding seasons and seeing the p-values were well above any threshold we would pick, it was not deemed necessary to do any diagnostic checks on the model with the seasons encoded in addition to numeric variables in one whole model.

Interpretation:

In broader sense, seasons don’t affect the demand of the houses because prices of the houses do not correlate with purchases in different seasons.

Conclusion:

We have two final regression models answering both of our questions of interest respectively. Our first question of interest was finding if consumers value being closer to MRT station or having more convenience stores around them. To answer this question, we first showed that both the predictors help explain the variances in the prices. Then, we found that our predictor distance to MRT stores had a much stronger effect on the price than convince stores. We also strengthened this assumption by adding predictors to our simple linear regression models, which showed some relationship with the price, thereby making a multiple linear regression comparison. Additionally, we also transformed our predictor distance to MRT station to match linear regression assumptions. After these adjustments, we concluded that consumers value being closer to MRT station than having more convenience stores around them. Our second question was to find in which seasons the prices are the highest and lowest. To answer it, we included three dummy variables as our predictors for our four categorical variables namely spring, summer, winter, and fall and checked their relation with our response price. We concluded that seasons are not a significant predictors for the prices of house based on our findings. Also, we didn’t include the column values for longitude and latitude coordinates from our original dataset because all their values showed a small degree of variation. On typing the coordinates for the values on Google Maps, we found that the houses weren’t separated from each other noticeably such that we could eliminate factors like weather conditions for different areas in determining the house prices.

The generality of our findings now comes into question. Let’s break down the sampling scenario (concepts from PSTAT 100):

Population: All houses purchased in Taiwan throughout all of time.
Frame: All reported purchased houses in Taiwan throughout all of time.
Sample: Houses purchased from 2012 to 2013 in Sindian Dist., New Taipei City, Taiwan. (Subset of frame)

Overlap: frame can partly overlap population (we can tax evasion through not reporting sale of houses and our sample scenario would be the typical sample one)
Mechanism: nonrandom (convenience) sample

Scope of inference or in this case how general our results are: Can extrapolate to a subpopulation which would be housing prices during the time of 2012 to 2013 but not through all of time or our population.

Regarding transaction dates and seasons, there are some caveats to our analysis:

https://www.bbc.com/news/business-20779609 (https://www.bbc.com/news/business-20779609)
https://www.globalpropertyguide.com/home-price-trends/Taiwan (https://www.globalpropertyguide.com/home-price-trends/Taiwan)

Based off some of these sources we know that generally prices for housing at the time were high. Our model takes only month just because of how in the process of converting the double values of transaction date, month was only indexed. We would think for better accuracy in terms of trying to have season as a factor, more data would be required from previous years leading up to present day or some better metric would be used like climate data over time. Even then we think it's still difficult to try to isolate season in and of itself solely from transaction date without any other unaccounted for factors at any given time.

Appendices

Appendix 1

This code is to import the data set

```
library("readxl")
"Some group members were having issues importing the data so here is a
workaround where they would open the excel sheet select and copy (Ctr+a then ctrl+c)
the data and then run this line to import the data
"
```

```
## [1] "Some group members were having issues importing the data so here is a \nworkaround where
they would open the excel sheet select and copy (Ctr+a then ctrl+c)\nthe data and then run this
line to import the data\n"
```

```
realEstateData <- read.table(file = "clipboard", sep = "\t", header=TRUE)
```

This code rename all of our variables for the code

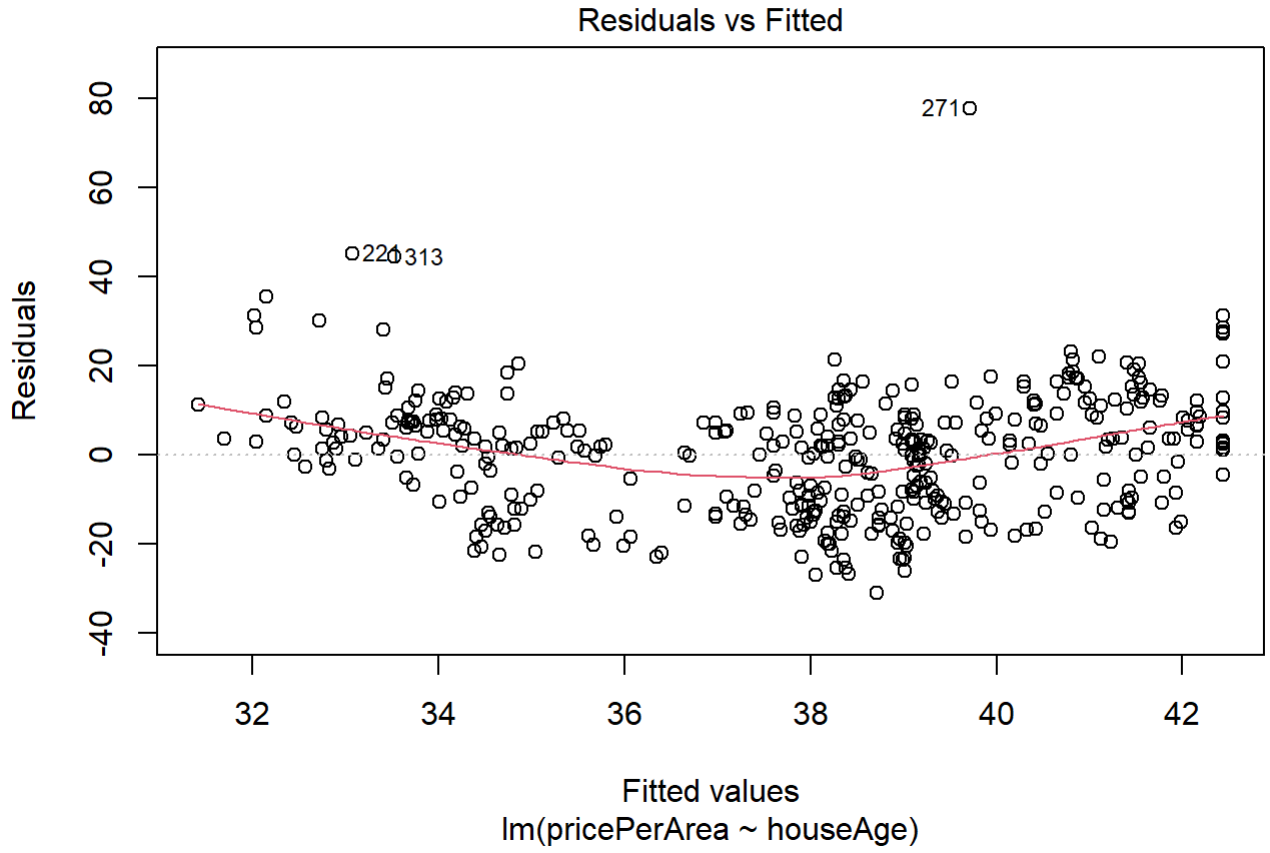
```
houseAge = realEstateData$X2.house.age
numConvStores = realEstateData$X4.number.of.convenience.stores
mrt = realEstateData$X3.distance.to.the.nearest.MRT.station
pricePerArea = realEstateData$Y.house.price.of.unit.area
transDate = realEstateData$X1.transaction.date

houseAgeModel = lm(pricePerArea~houseAge)
csModel = lm(pricePerArea~numConvStores)
mrtModel = lm(pricePerArea~mrt)
dateModel = lm(pricePerArea~transDate)
```

```
summary(houseAgeModel)
```

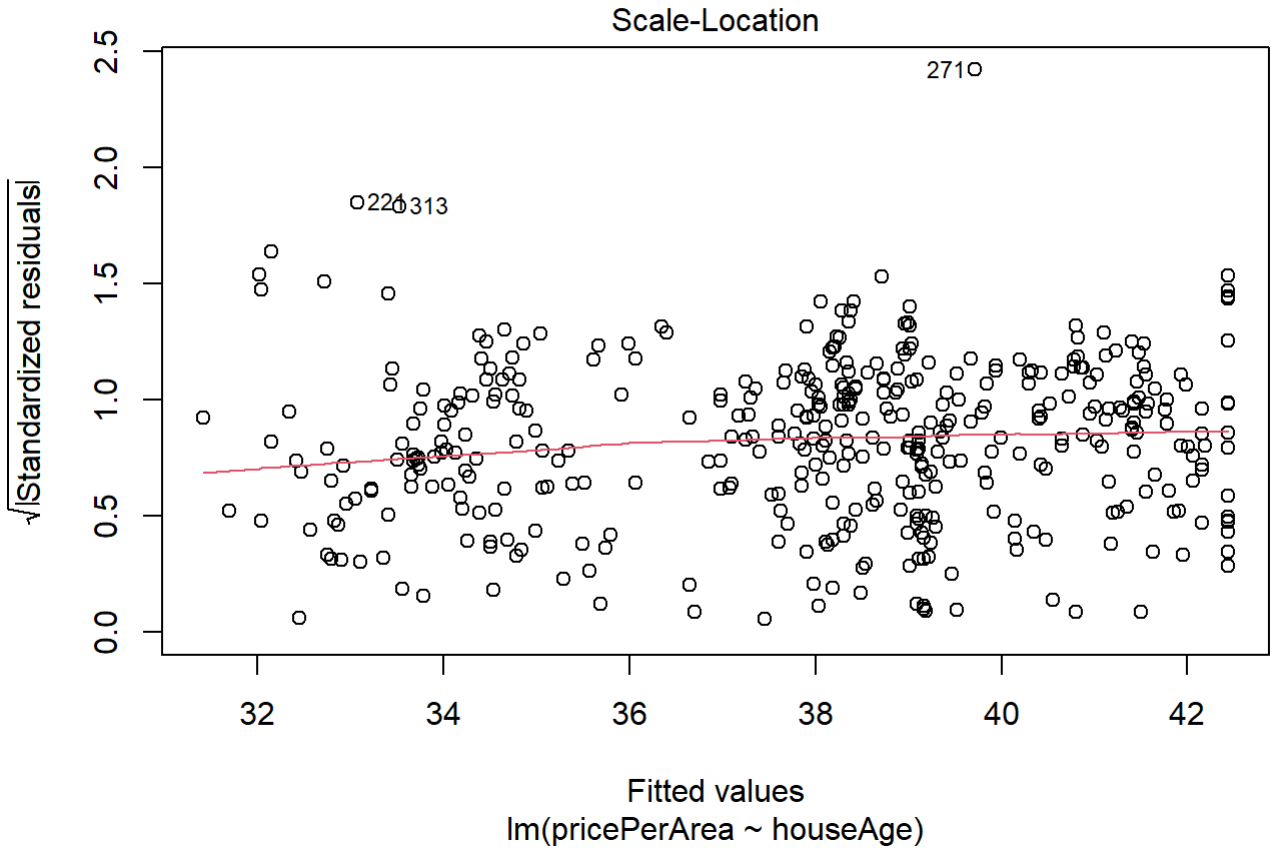
```
##
## Call:
## lm(formula = pricePerArea ~ houseAge)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.113 -10.738   1.626   8.199  77.781
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.43470    1.21098  35.042  < 2e-16 ***
## houseAge    -0.25149    0.05752  -4.372 1.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.32 on 412 degrees of freedom
## Multiple R-squared:  0.04434,    Adjusted R-squared:  0.04202
## F-statistic: 19.11 on 1 and 412 DF,  p-value: 1.56e-05
```

```
plot(pricePerArea~houseAge, xlab = 'Age of House', ylab = 'Price per area')
abline(houseAgeModel, col='red')
```

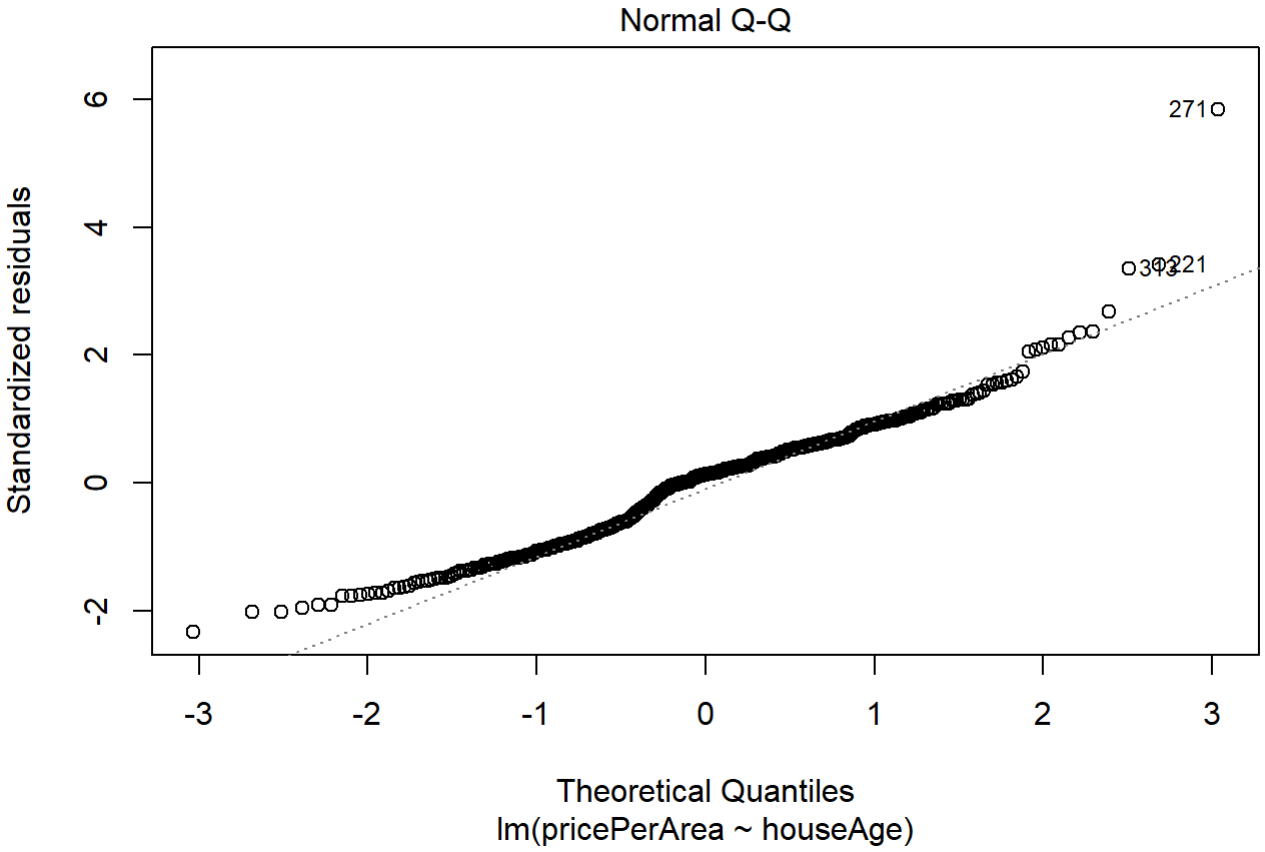
```
plot(houseAgeModel, which = 1)
```



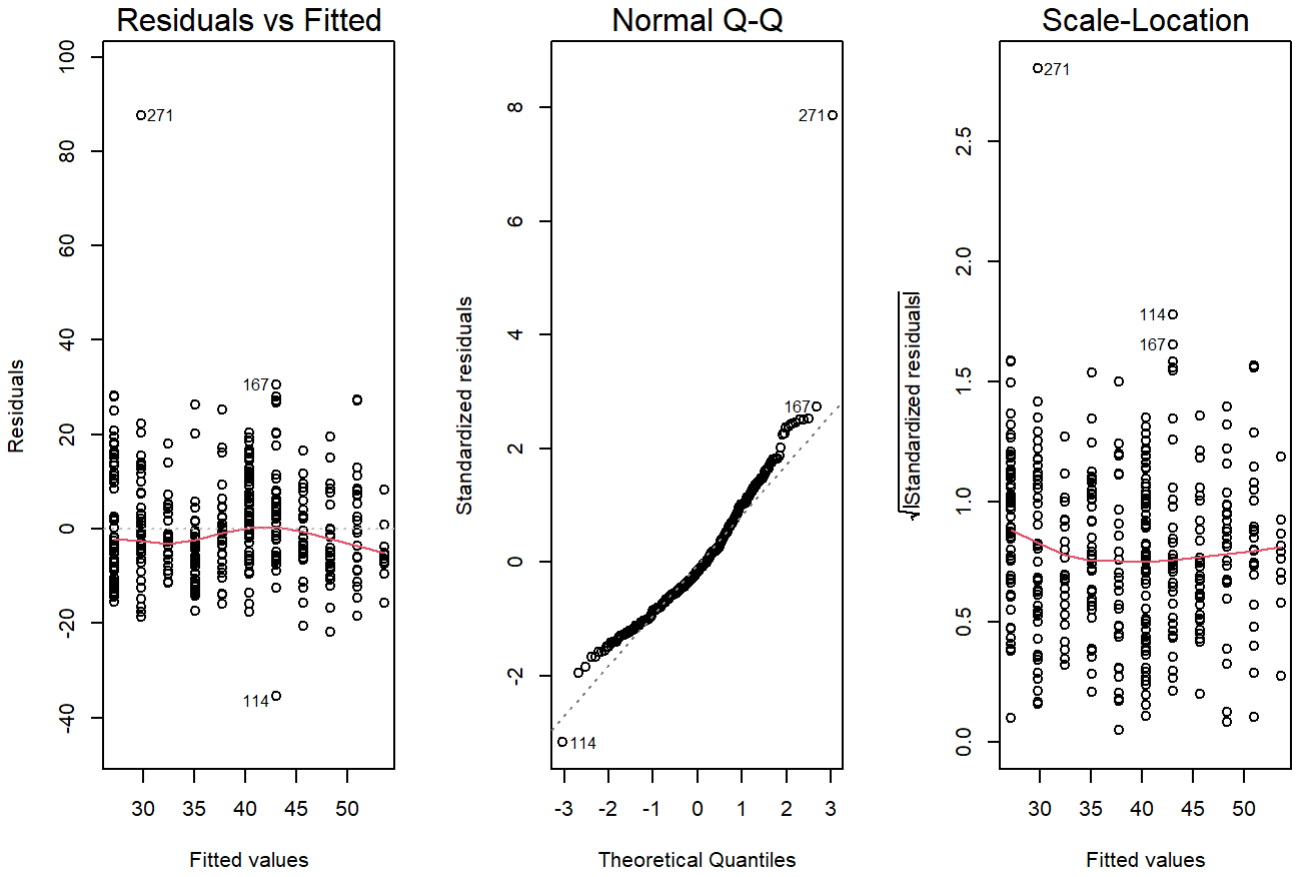

```
plot(houseAgeModel, which = 3)
```




```
plot(houseAgeModel, which = 2)
```



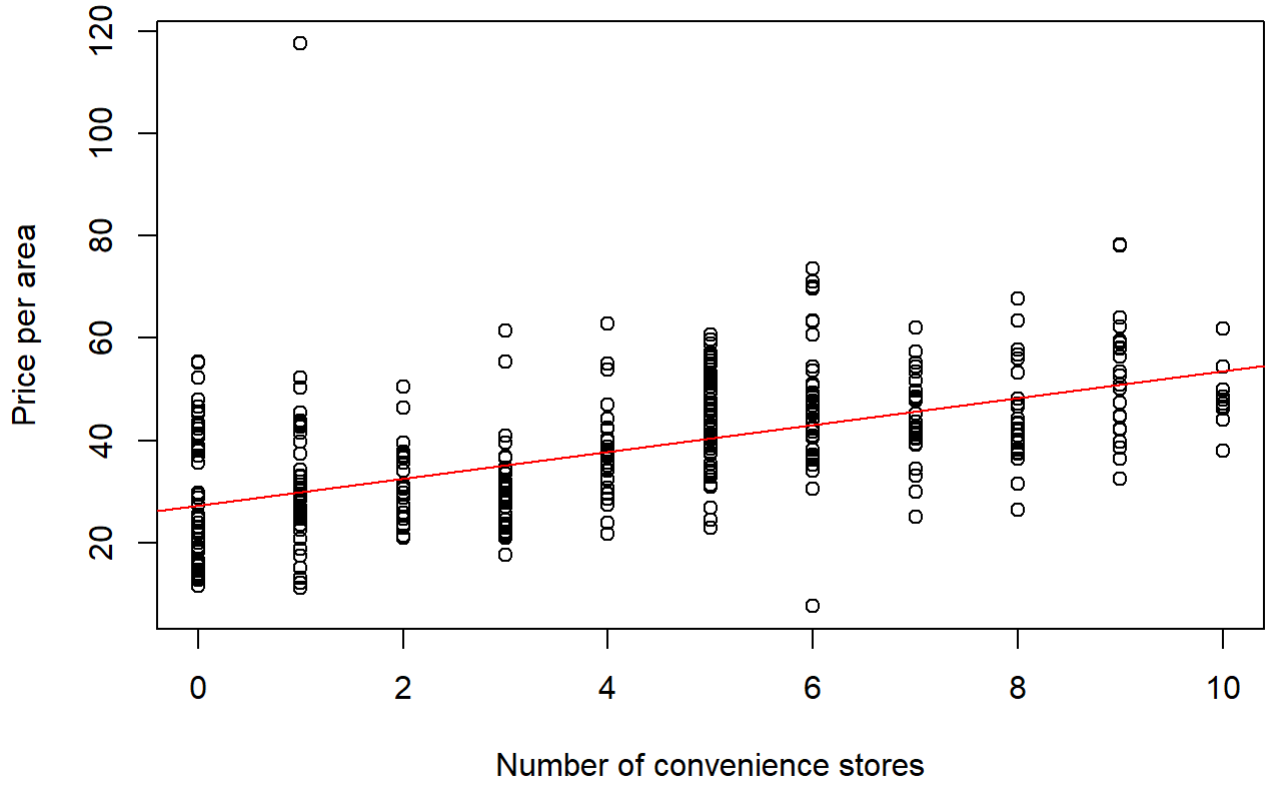

```
par(mfrow = c(1,3))
for(j in 1:3){
  plot(csModel, which = j)
}
```



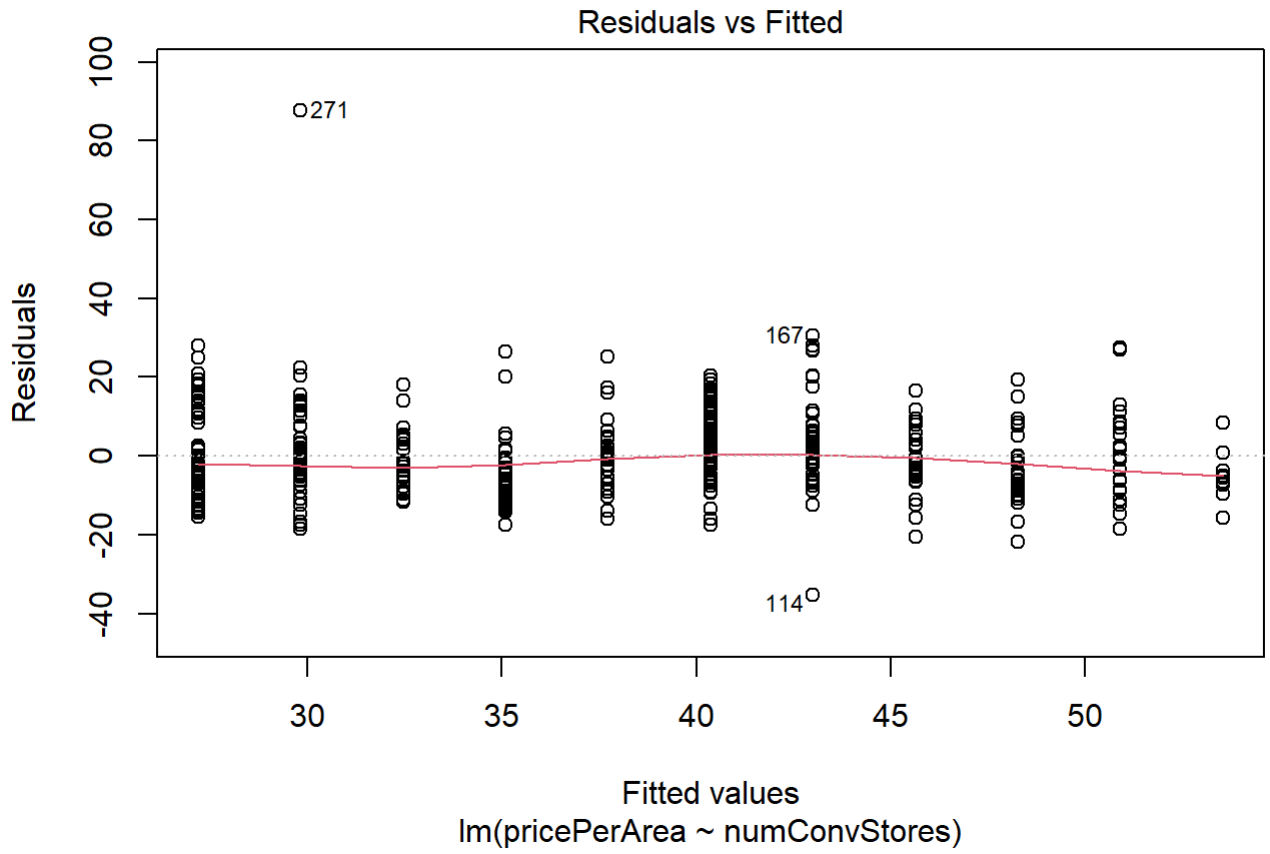

```
summary(csModel)
```

```
##
## Call:
## lm(formula = pricePerArea ~ numConvStores)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.407  -7.341  -1.788   5.984  87.681
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    27.1811     0.9419   28.86   <2e-16 ***
## numConvStores   2.6377     0.1868   14.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.18 on 412 degrees of freedom
## Multiple R-squared:  0.326,  Adjusted R-squared:  0.3244
## F-statistic: 199.3 on 1 and 412 DF,  p-value: < 2.2e-16
```

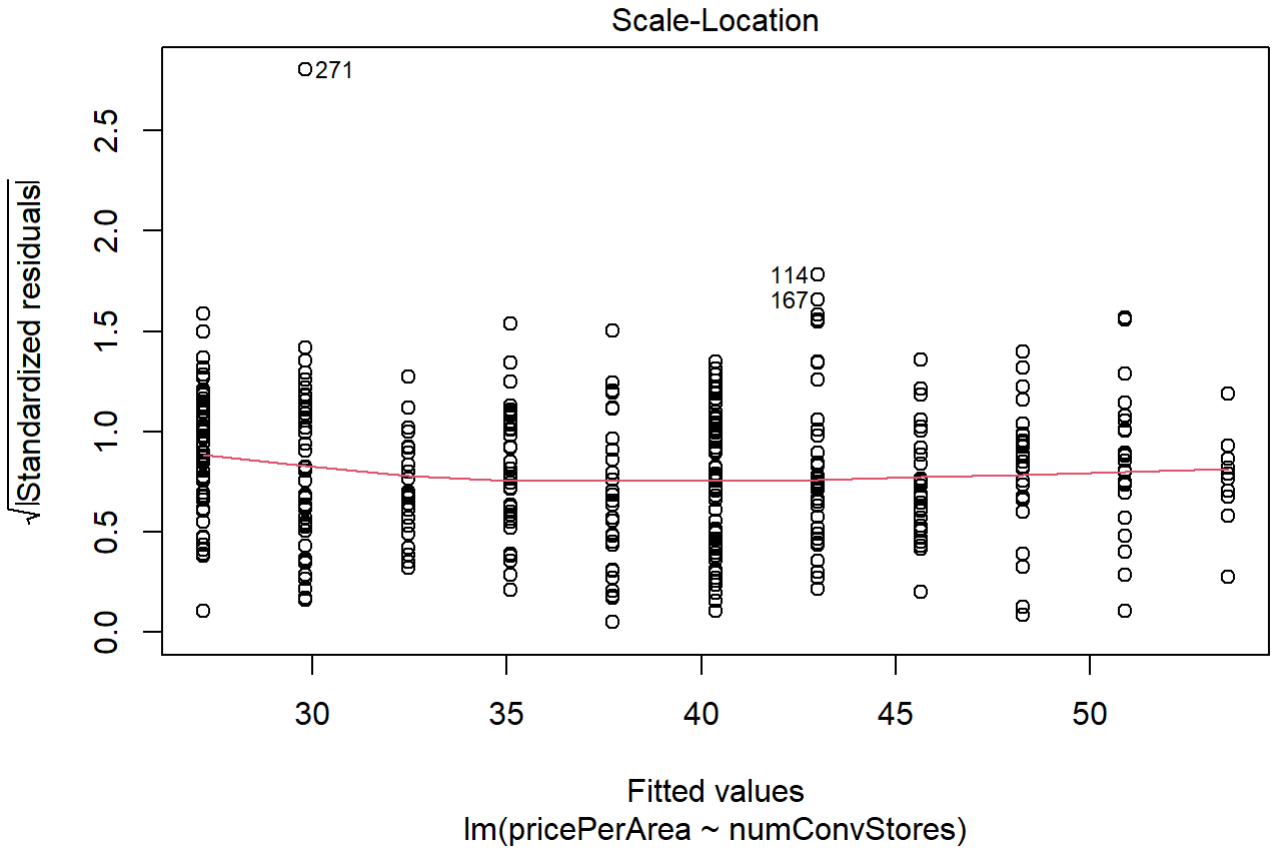
```
plot(pricePerArea~numConvStores, xlab = 'Number of convenience stores',
                                 ylab = 'Price per area')
abline(csModel, col='red')
```



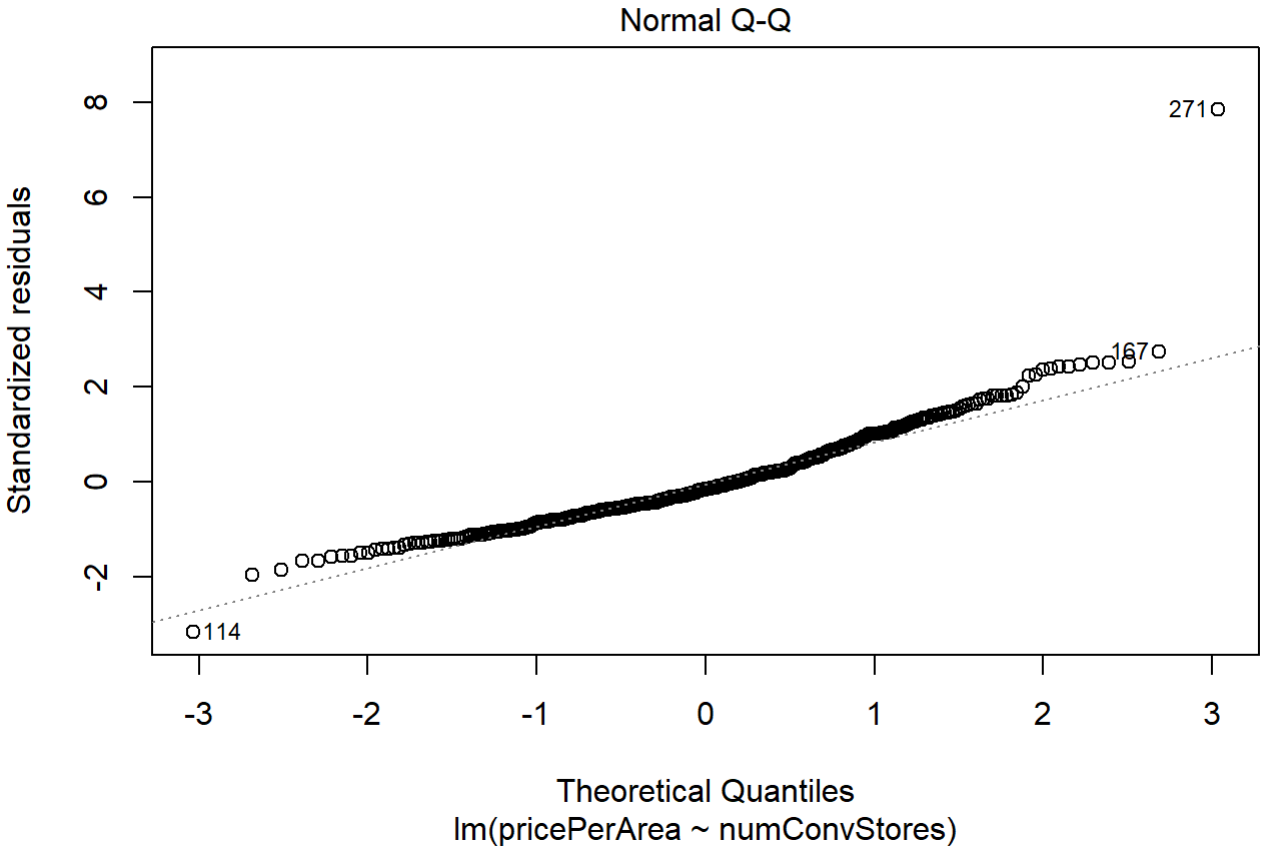

```
plot(csModel, which = 1)
```



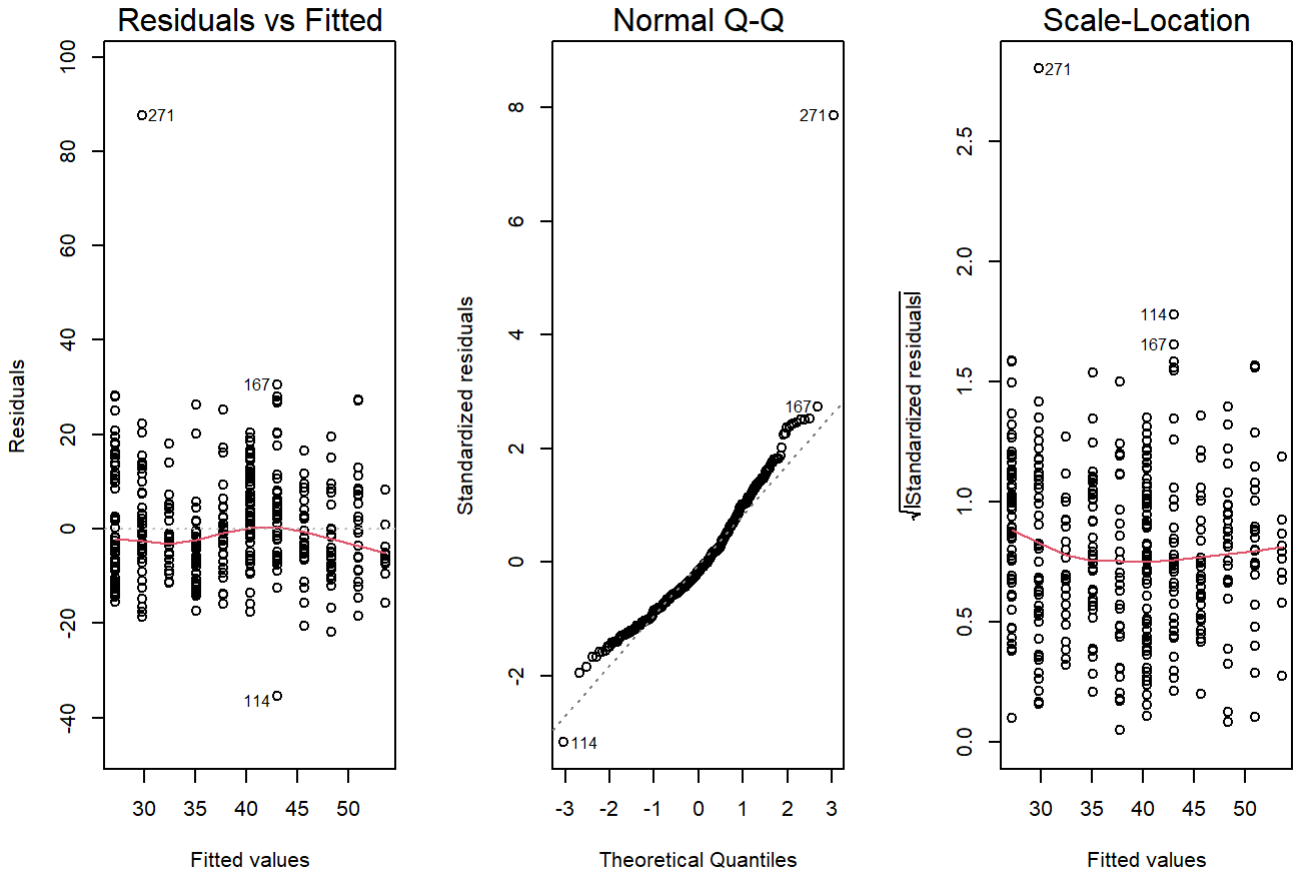

```
plot(csModel, which = 3)
```




```
plot(csModel, which = 2)
```



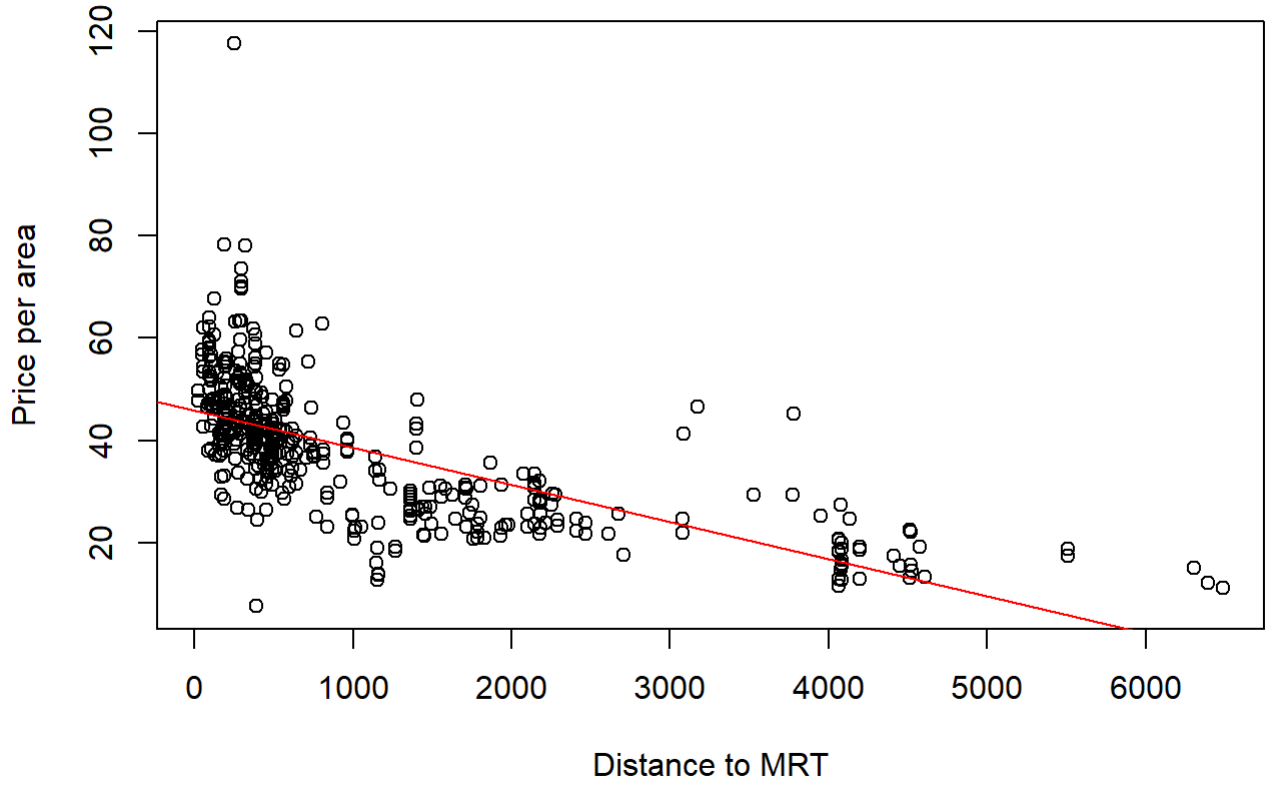

```
par(mfrow = c(1,3))
for(j in 1:3){
  plot(csModel, which = j)
}
```



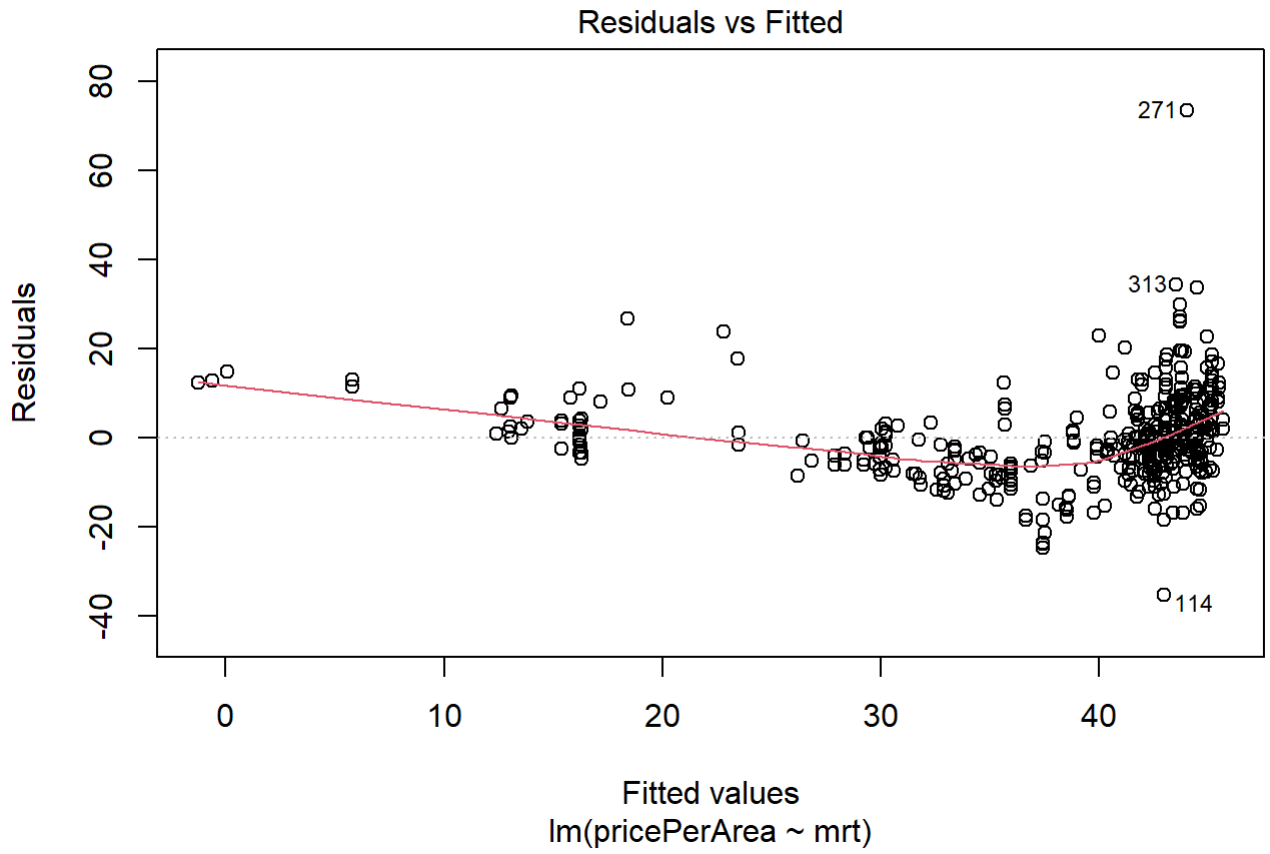

```
summary(mrtModel)
```

```
##
## Call:
## lm(formula = pricePerArea ~ mrt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.396  -6.007  -1.195   4.831  73.483
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 45.8514271  0.6526105   70.26   <2e-16 ***
## mrt         -0.0072621  0.0003925  -18.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.07 on 412 degrees of freedom
## Multiple R-squared:  0.4538, Adjusted R-squared:  0.4524
## F-statistic: 342.2 on 1 and 412 DF,  p-value: < 2.2e-16
```

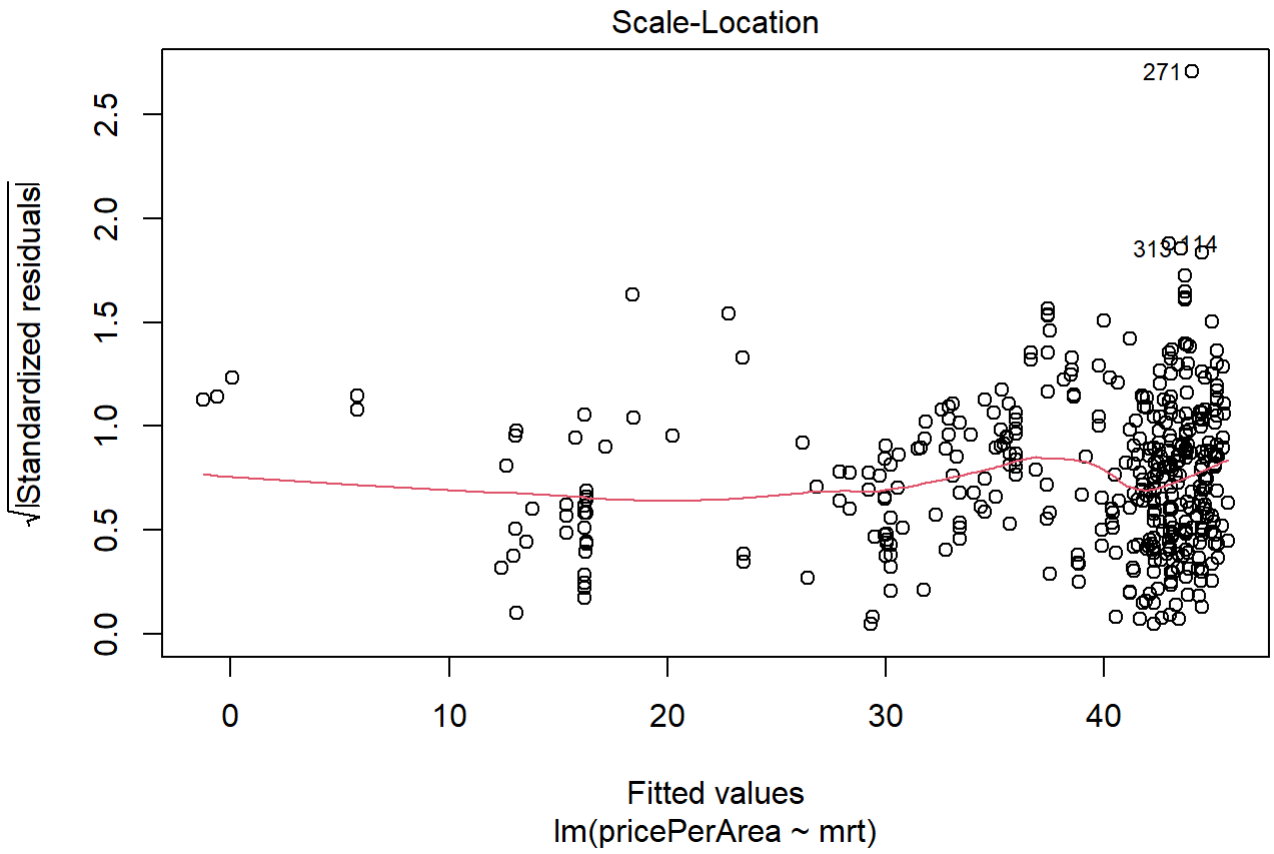
```
plot(pricePerArea~mrt, xlab = 'Distance to MRT', ylab = 'Price per area')
abline(mrtModel, col='red')
```



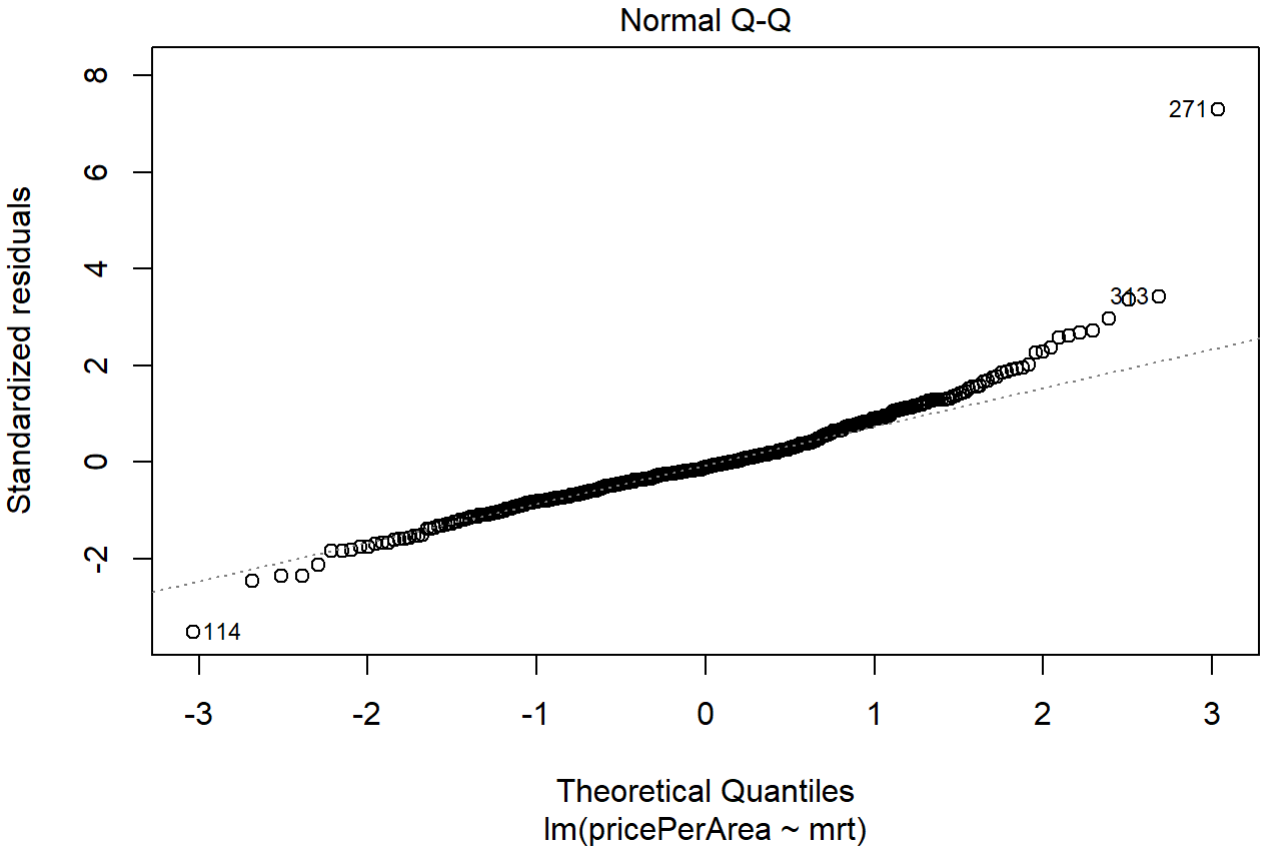

```
plot(mrtModel, which = 1)
```



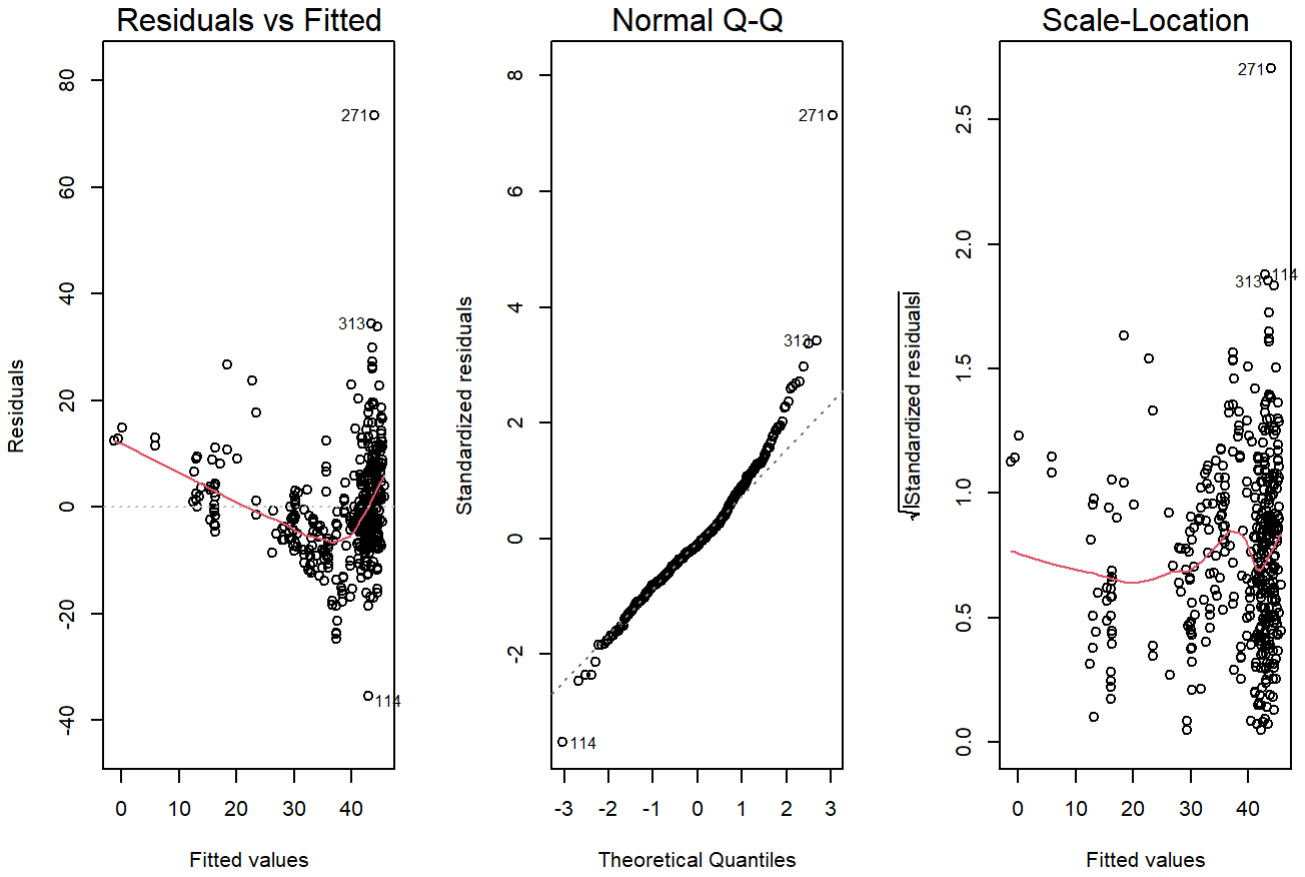

```
plot(mrtModel, which = 3)
```



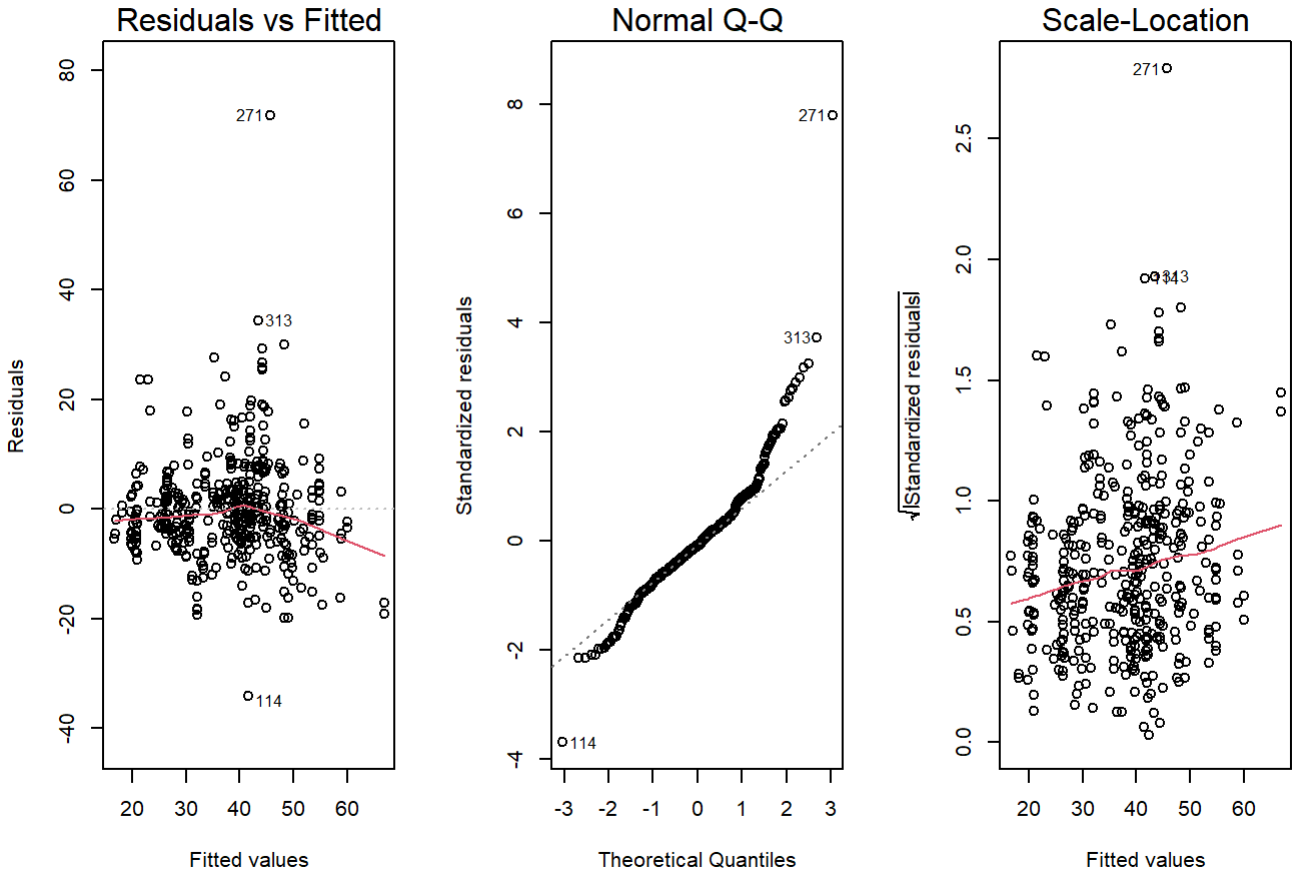

```
plot(mrtModel, which = 2)
```




```
par(mfrow = c(1,3))
for(j in 1:3){
  plot(mrtModel, which = j)
}
```




```
#Transformed model
transModel = lm(pricePerArea~log(mrt))
par(mfrow=c(1,3))
for(j in 1:3){
  plot(transModel, which = j)
}
```




```
TransmrtModel = lm(pricePerArea~log(mrt))
summary(TransmrtModel)
```

```
##
## Call:
## lm(formula = pricePerArea ~ log(mrt))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.103  -5.023  -0.827   3.449  71.846
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  95.0169     2.6369   36.03   <2e-16 ***
## log(mrt)     -8.9235     0.4064  -21.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.247 on 412 degrees of freedom
## Multiple R-squared:  0.5393, Adjusted R-squared:  0.5381
## F-statistic: 482.2 on 1 and 412 DF,  p-value: < 2.2e-16
```

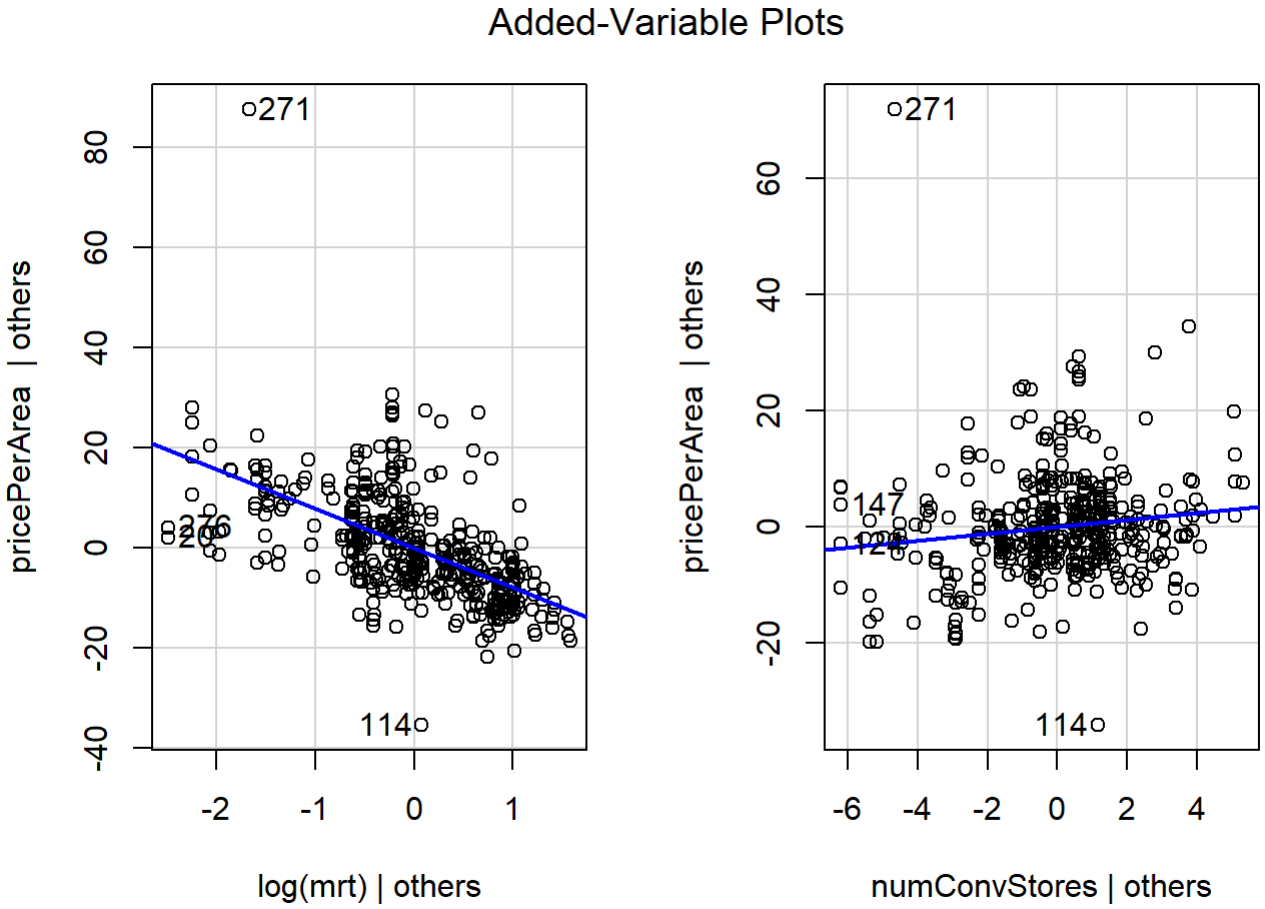
```
library(car)
```

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

```
full.lm = lm(pricePerArea ~ log(mrt) + numConvStores)
summary(full.lm)
```

```
##
## Call:
## lm(formula = pricePerArea ~ log(mrt) + numConvStores)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.783  -5.106  -0.756   3.462  74.582
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    85.8141     4.2006   20.43  < 2e-16 ***
## log(mrt)       -7.8611     0.5536  -14.20  < 2e-16 ***
## numConvStores   0.5891     0.2104    2.80  0.00536 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.171 on 411 degrees of freedom
## Multiple R-squared:  0.5479, Adjusted R-squared:  0.5457
## F-statistic:   249 on 2 and 411 DF,  p-value: < 2.2e-16
```

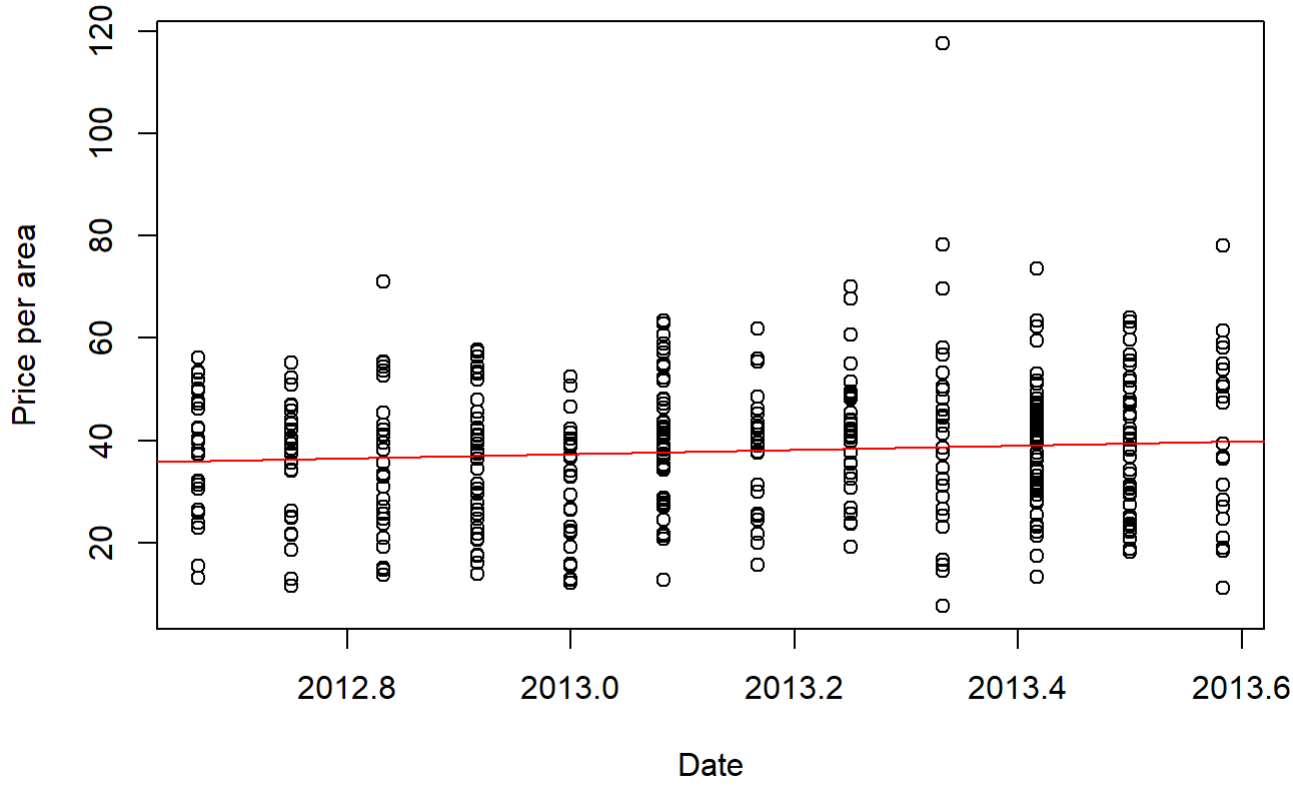
```
avPlots(full.lm)
```



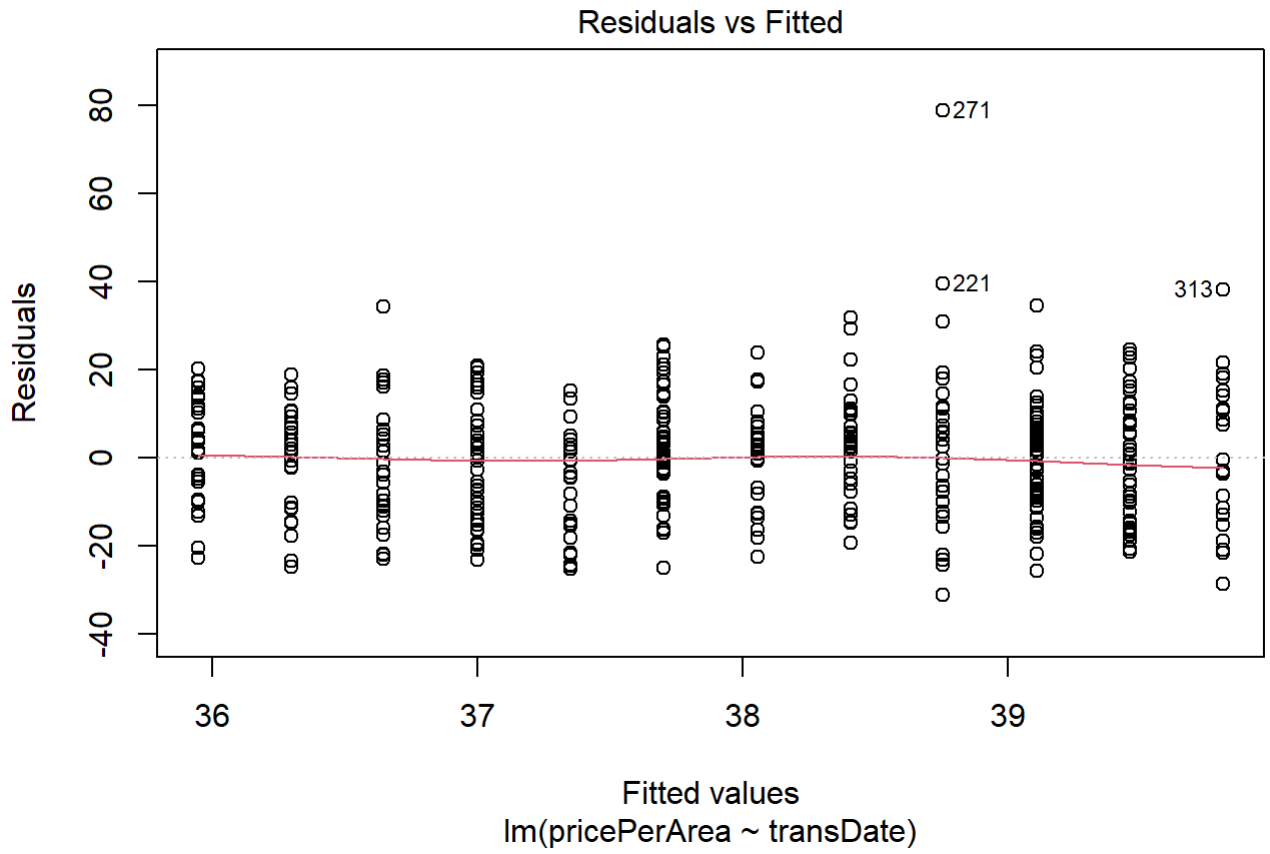

```
summary(dateModel)
```

```
##
## Call:
## lm(formula = pricePerArea ~ transDate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.157 -10.083   0.921   8.528  78.743
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8461.350   4767.669  -1.775   0.0767 .
## transDate       4.222      2.368   1.783   0.0754 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.57 on 412 degrees of freedom
## Multiple R-squared:  0.007655,   Adjusted R-squared:  0.005246
## F-statistic: 3.178 on 1 and 412 DF,  p-value: 0.07537
```

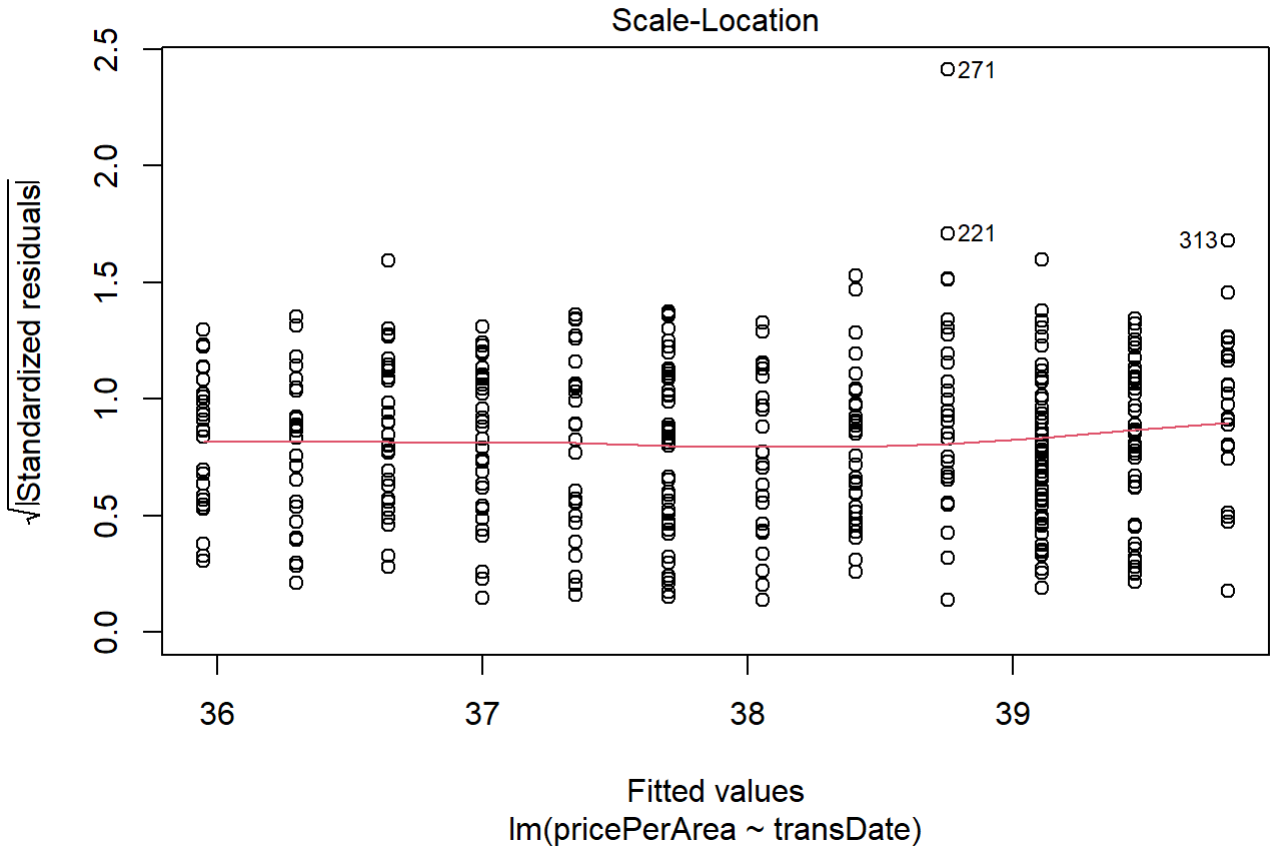
```
plot(pricePerArea~transDate, xlab = 'Date', ylab = 'Price per area')
abline(dateModel, col='red')
```



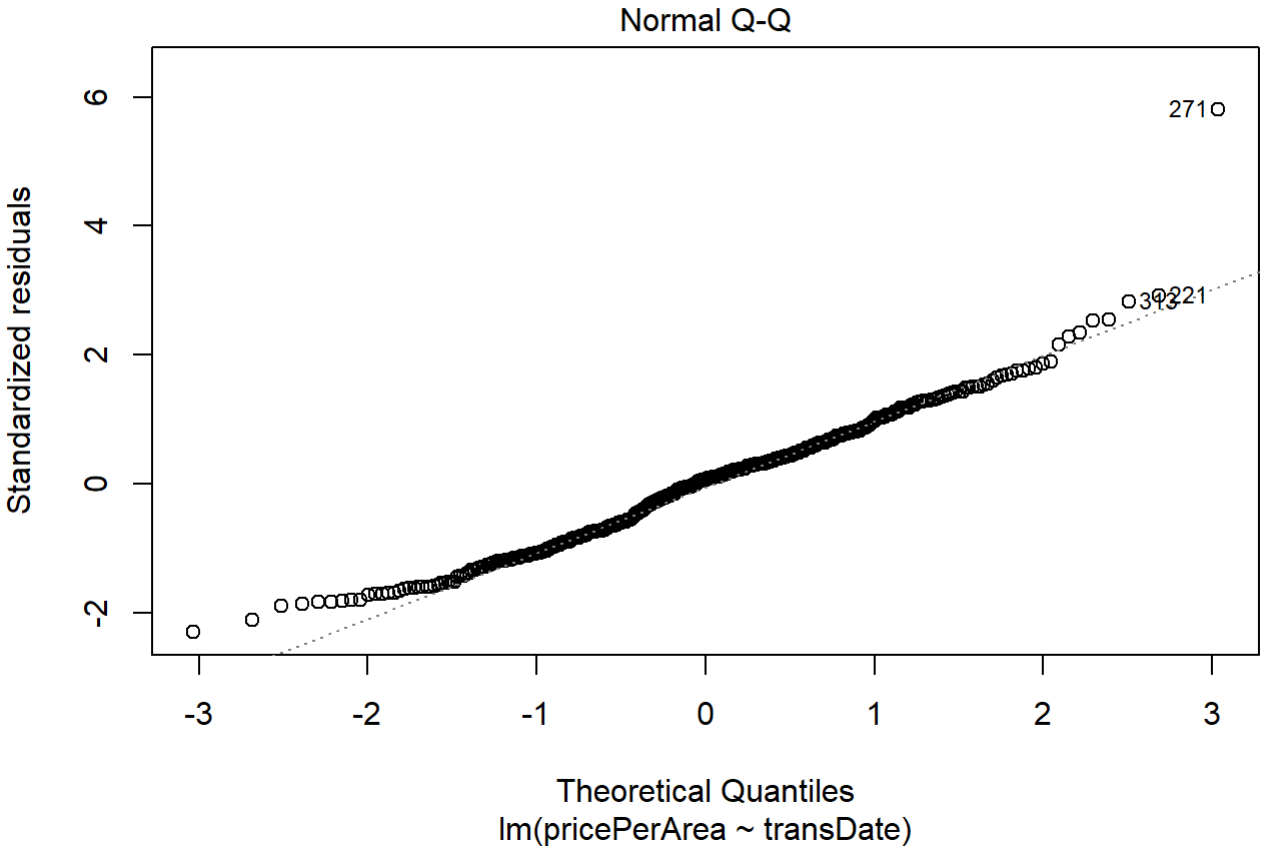

```
plot(dateModel, which = 1)
```



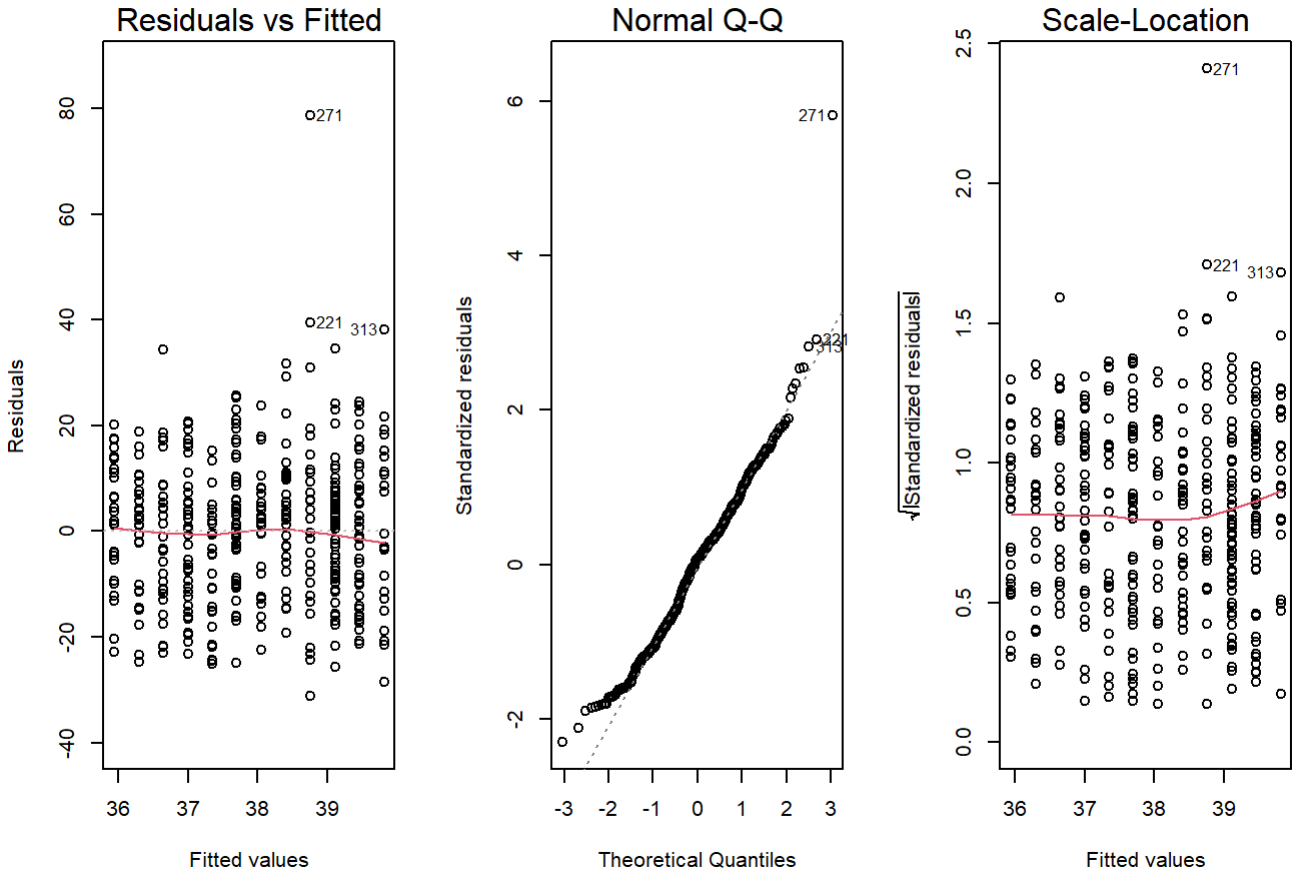

```
plot(dateModel, which = 3)
```




```
plot(dateModel, which = 2)
```




```
par(mfrow=c(1,3))
for(j in 1:3){
  plot(dateModel, which = j)
}
```



From these plots, we do not need to transform our predictor.

```
summary(dateModel)
```

```
##
## Call:
## lm(formula = pricePerArea ~ transDate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.157 -10.083   0.921   8.528  78.743
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8461.350   4767.669  -1.775   0.0767 .
## transDate       4.222      2.368   1.783   0.0754 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.57 on 412 degrees of freedom
## Multiple R-squared:  0.007655,   Adjusted R-squared:  0.005246
## F-statistic: 3.178 on 1 and 412 DF,  p-value: 0.07537
```

The p value is relatively high, which means that the time/season when the home is not a significant predictor of price per area.

Here we factor dates and create substrings for the months:

```
dataModel_factor = factor(transDate)
dataModel_factor
```

```
##   [1] 2012.917 2012.917 2013.583 2013.5   2012.833 2012.667 2012.667 2013.417
##   [9] 2013.5   2013.417 2013.083 2013.333 2012.917 2012.667 2013.5   2013.583
##  [17] 2013.25  2012.75  2013.417 2012.667 2013.417 2013.417 2012.917 2013.083
##  [25] 2013     2013.083 2012.667 2013.25  2013.5   2013.083 2013.5   2012.75
##  [33] 2012.75  2013.25  2012.75  2013.5   2012.917 2013.167 2012.667 2013.167
##  [41] 2013     2013.5   2013.417 2012.75  2013.583 2013.083 2013.417 2013.583
##  [49] 2013.417 2012.667 2013.417 2013.083 2013.583 2013.083 2013.083 2012.833
##  [57] 2013.417 2012.917 2013.5   2013.083 2013.417 2013.5   2012.917 2013.583
##  [65] 2013.333 2013.417 2013     2013.5   2013.417 2012.833 2013.583 2013.083
##  [73] 2013.583 2013.167 2012.917 2013.5   2013.583 2012.833 2012.917 2013
##  [81] 2013.5   2013     2013.083 2012.917 2013.083 2012.75  2012.833 2013.583
##  [89] 2012.917 2013.5   2012.833 2013.25  2012.917 2012.917 2012.917 2012.917
##  [97] 2013.417 2013.083 2013.417 2013.417 2013.5   2012.833 2013.083 2012.75
## [105] 2012.667 2012.833 2013.083 2013.333 2013.417 2013.583 2013.083 2013.583
## [113] 2013.417 2013.333 2012.667 2013.083 2013     2013     2013.5   2013.5
## [121] 2013.167 2013.5   2013.25  2013.417 2012.917 2013.167 2013.083 2013.25
## [129] 2013.083 2013.417 2013.25  2013.5   2013.167 2012.833 2012.667 2012.917
## [137] 2012.75  2013.5   2013.167 2012.667 2013.25  2013.333 2013.417 2013.5
## [145] 2013.083 2012.917 2012.75  2012.75  2013.5   2012.667 2013.25  2013.5
## [153] 2013.333 2013.25  2013.5   2013.167 2013.583 2013.25  2013     2012.667
## [161] 2012.917 2013.417 2012.75  2013.5   2012.833 2012.917 2013.417 2013.417
## [169] 2013.083 2013.417 2013.333 2013.083 2013.583 2013.083 2013.417 2013.083
## [177] 2012.833 2013.083 2013.5   2013.083 2012.667 2013.167 2013.5   2013.5
## [185] 2012.75  2012.75  2013.167 2013     2012.917 2012.917 2013.5   2013.167
## [193] 2013.167 2013.417 2013.5   2013.333 2013     2013.25  2013.083 2013.417
## [201] 2013.417 2013.417 2012.917 2012.667 2013     2013.083 2013.25  2013.083
## [209] 2012.75  2012.833 2013.5   2013.083 2013.333 2013.083 2013.583 2013.333
## [217] 2013.25  2012.917 2013.417 2012.75  2013.333 2013.333 2013.583 2013.25
## [225] 2013.333 2013.25  2013     2012.917 2013.417 2013.583 2013.5   2012.833
## [233] 2012.917 2013.333 2013.25  2012.75  2013.167 2013.167 2013.083 2013.5
## [241] 2013.083 2013.5   2012.833 2013.417 2013.083 2013.417 2013.417 2013.333
## [249] 2013     2012.833 2013.167 2012.917 2012.833 2012.667 2012.667 2013.417
## [257] 2012.667 2013.25  2013.417 2013.083 2013.25  2013.167 2012.917 2013.417
## [265] 2013.167 2012.833 2013.25  2012.833 2013.417 2013     2013.333 2012.917
## [273] 2012.75  2013.417 2013.167 2012.667 2013     2013.417 2012.75  2013.417
## [281] 2013.25  2013.333 2012.917 2013.417 2012.917 2013.167 2012.917 2013
## [289] 2013.583 2013.333 2013.083 2012.833 2013.083 2012.667 2013.5   2013.167
## [297] 2012.75  2012.833 2013.333 2013.167 2013.083 2012.75  2013.5   2013.5
## [305] 2013.417 2013.083 2013.5   2012.833 2013.417 2013.25  2013.583 2013.167
## [313] 2013.583 2013.333 2013.25  2013.083 2013.25  2012.75  2013.333 2013.25
## [321] 2012.75  2012.917 2013     2013.417 2012.667 2013.083 2013.5   2013.417
## [329] 2012.833 2013     2013.083 2013.333 2013.167 2012.75  2012.917 2013.583
## [337] 2012.833 2012.833 2012.917 2013.333 2013.333 2013     2012.667 2013
## [345] 2013.5   2012.667 2013.417 2013.583 2012.833 2012.75  2013     2012.833
## [353] 2012.833 2013.5   2013.417 2013.25  2012.833 2013.417 2013.167 2013.5
## [361] 2012.667 2013.083 2013.417 2013.5   2013.417 2012.917 2012.75  2012.833
## [369] 2013.417 2012.667 2012.75  2013.5   2013     2013.083 2013.25  2013.25
## [377] 2013.417 2013.333 2013.333 2013.333 2013.333 2013.417 2013     2012.667
## [385] 2012.75  2013     2012.833 2013.25  2013.5   2013.25  2013.5   2013.583
## [393] 2013.083 2013     2013.5   2012.917 2012.667 2013.417 2013.417 2012.917
## [401] 2013.25  2013.083 2012.833 2012.667 2013.333 2012.667 2013.167 2013
## [409] 2013.417 2013     2012.667 2013.25  2013     2013.5
## 12 Levels: 2012.667 2012.75 2012.833 2012.917 2013 2013.083 ... 2013.583
```

```
seasonStrings = substr(transDate,5,9)
seasonDate = as.numeric(seasonStrings)
seasonDate
```

```
##   [1] 0.917 0.917 0.583 0.500 0.833 0.667 0.667 0.417 0.500 0.417 0.083 0.333
##  [13] 0.917 0.667 0.500 0.583 0.250 0.750 0.417 0.667 0.417 0.417 0.917 0.083
##  [25]    NA 0.083 0.667 0.250 0.500 0.083 0.500 0.750 0.750 0.250 0.750 0.500
##  [37] 0.917 0.167 0.667 0.167    NA 0.500 0.417 0.750 0.583 0.083 0.417 0.583
##  [49] 0.417 0.667 0.417 0.083 0.583 0.083 0.083 0.833 0.417 0.917 0.500 0.083
##  [61] 0.417 0.500 0.917 0.583 0.333 0.417    NA 0.500 0.417 0.833 0.583 0.083
##  [73] 0.583 0.167 0.917 0.500 0.583 0.833 0.917    NA 0.500    NA 0.083 0.917
##  [85] 0.083 0.750 0.833 0.583 0.917 0.500 0.833 0.250 0.917 0.917 0.917 0.917
##  [97] 0.417 0.083 0.417 0.417 0.500 0.833 0.083 0.750 0.667 0.833 0.083 0.333
## [109] 0.417 0.583 0.083 0.583 0.417 0.333 0.667 0.083    NA    NA 0.500 0.500
## [121] 0.167 0.500 0.250 0.417 0.917 0.167 0.083 0.250 0.083 0.417 0.250 0.500
## [133] 0.167 0.833 0.667 0.917 0.750 0.500 0.167 0.667 0.250 0.333 0.417 0.500
## [145] 0.083 0.917 0.750 0.750 0.500 0.667 0.250 0.500 0.333 0.250 0.500 0.167
## [157] 0.583 0.250    NA 0.667 0.917 0.417 0.750 0.500 0.833 0.917 0.417 0.417
## [169] 0.083 0.417 0.333 0.083 0.583 0.083 0.417 0.083 0.833 0.083 0.500 0.083
## [181] 0.667 0.167 0.500 0.500 0.750 0.750 0.167    NA 0.917 0.917 0.500 0.167
## [193] 0.167 0.417 0.500 0.333    NA 0.250 0.083 0.417 0.417 0.417 0.917 0.667
## [205]    NA 0.083 0.250 0.083 0.750 0.833 0.500 0.083 0.333 0.083 0.583 0.333
## [217] 0.250 0.917 0.417 0.750 0.333 0.333 0.583 0.250 0.333 0.250    NA 0.917
## [229] 0.417 0.583 0.500 0.833 0.917 0.333 0.250 0.750 0.167 0.167 0.083 0.500
## [241] 0.083 0.500 0.833 0.417 0.083 0.417 0.417 0.333    NA 0.833 0.167 0.917
## [253] 0.833 0.667 0.667 0.417 0.667 0.250 0.417 0.083 0.250 0.167 0.917 0.417
## [265] 0.167 0.833 0.250 0.833 0.417    NA 0.333 0.917 0.750 0.417 0.167 0.667
## [277]    NA 0.417 0.750 0.417 0.250 0.333 0.917 0.417 0.917 0.167 0.917    NA
## [289] 0.583 0.333 0.083 0.833 0.083 0.667 0.500 0.167 0.750 0.833 0.333 0.167
## [301] 0.083 0.750 0.500 0.500 0.417 0.083 0.500 0.833 0.417 0.250 0.583 0.167
## [313] 0.583 0.333 0.250 0.083 0.250 0.750 0.333 0.250 0.750 0.917    NA 0.417
## [325] 0.667 0.083 0.500 0.417 0.833    NA 0.083 0.333 0.167 0.750 0.917 0.583
## [337] 0.833 0.833 0.917 0.333 0.333    NA 0.667    NA 0.500 0.667 0.417 0.583
## [349] 0.833 0.750    NA 0.833 0.833 0.500 0.417 0.250 0.833 0.417 0.167 0.500
## [361] 0.667 0.083 0.417 0.500 0.417 0.917 0.750 0.833 0.417 0.667 0.750 0.500
## [373]    NA 0.083 0.250 0.250 0.417 0.333 0.333 0.333 0.333 0.417    NA 0.667
## [385] 0.750    NA 0.833 0.250 0.500 0.250 0.500 0.583 0.083    NA 0.500 0.917
## [397] 0.667 0.417 0.417 0.917 0.250 0.083 0.833 0.667 0.333 0.667 0.167    NA
## [409] 0.417    NA 0.667 0.250    NA 0.500
```

Here we encode specific ranges to their corresponding seasons or convert dates to seasons:

```
x = 1
while (x <= length(seasonDate)) {
  if (seasonDate[x] <= 0.250 | is.na(seasonDate[x]) | seasonDate[x]>=.9){
    seasonDate[x] = "Winter"
  }
  else if (seasonDate[x] >= 0.251 & seasonDate[x]<=0.5){
    seasonDate[x] = "Spring"
  }
  else if (seasonDate[x] >= 0.51 & seasonDate[x]<=.75){
    seasonDate[x] = "Summer"
  }
  else{
    seasonDate[x] = "Fall"
  }
  x = x + 1
}

factor(seasonDate)
```

```
##   [1] Winter Winter Summer Spring Fall   Summer Summer Spring Spring Spring
##  [11] Winter Spring Winter Summer Spring Summer Winter Summer Spring Summer
##  [21] Spring Spring Winter Winter Winter Winter Summer Winter Spring Winter
##  [31] Spring Summer Summer Winter Summer Spring Winter Winter Summer Winter
##  [41] Winter Spring Spring Summer Summer Winter Spring Summer Spring Summer
##  [51] Spring Winter Summer Winter Winter Fall   Spring Winter Spring Winter
##  [61] Spring Spring Winter Summer Spring Spring Winter Spring Spring Fall
##  [71] Summer Winter Summer Winter Winter Spring Summer Fall   Winter Winter
##  [81] Spring Winter Winter Winter Winter Summer Fall   Summer Winter Spring
##  [91] Fall   Winter Winter Winter Winter Winter Spring Winter Spring Spring
## [101] Spring Fall   Winter Summer Summer Fall   Winter Spring Spring Summer
## [111] Winter Summer Spring Spring Summer Winter Winter Winter Spring Spring
## [121] Winter Spring Winter Spring Winter Winter Winter Winter Winter Spring
## [131] Winter Spring Winter Fall   Summer Winter Summer Spring Winter Summer
## [141] Winter Spring Spring Spring Winter Winter Summer Summer Spring Summer
## [151] Winter Spring Spring Winter Spring Winter Summer Winter Winter Summer
## [161] Winter Spring Summer Spring Fall   Winter Spring Spring Winter Spring
## [171] Spring Winter Summer Winter Spring Winter Fall   Winter Spring Winter
## [181] Summer Winter Spring Spring Summer Summer Winter Winter Winter Winter
## [191] Spring Winter Winter Spring Spring Spring Winter Winter Winter Spring
## [201] Spring Spring Winter Summer Winter Winter Winter Winter Summer Fall
## [211] Spring Winter Spring Winter Summer Spring Winter Winter Spring Summer
## [221] Spring Spring Summer Winter Spring Winter Winter Winter Spring Summer
## [231] Spring Fall   Winter Spring Winter Summer Winter Winter Winter Spring
## [241] Winter Spring Fall   Spring Winter Spring Spring Spring Winter Fall
## [251] Winter Winter Fall   Summer Summer Spring Summer Winter Spring Winter
## [261] Winter Winter Winter Spring Winter Fall   Winter Fall   Spring Winter
## [271] Spring Winter Summer Spring Winter Summer Winter Spring Summer Spring
## [281] Winter Spring Winter Spring Winter Winter Winter Winter Summer Spring
## [291] Winter Fall   Winter Summer Spring Winter Summer Fall   Spring Winter
## [301] Winter Summer Spring Spring Spring Winter Spring Fall   Spring Winter
## [311] Summer Winter Summer Spring Winter Winter Winter Summer Spring Winter
## [321] Summer Winter Winter Spring Summer Winter Spring Spring Fall   Winter
## [331] Winter Spring Winter Summer Winter Summer Fall   Fall   Winter Spring
## [341] Spring Winter Summer Winter Spring Summer Spring Summer Fall   Summer
## [351] Winter Fall   Fall   Spring Spring Winter Fall   Spring Winter Spring
## [361] Summer Winter Spring Spring Spring Winter Summer Fall   Spring Summer
## [371] Summer Spring Winter Winter Winter Winter Spring Spring Spring Spring
## [381] Spring Spring Winter Summer Summer Winter Fall   Winter Spring Winter
## [391] Spring Summer Winter Winter Spring Winter Summer Spring Spring Winter
## [401] Winter Winter Fall   Summer Spring Summer Winter Winter Spring Winter
## [411] Summer Winter Winter Spring
## Levels: Fall Spring Summer Winter
```

Quick summary of seasons as the sole predictors:

```
np.lm = lm(pricePerArea~seasonDate, data = realEstateData)
summary(np.lm)
```

```
##
## Call:
## lm(formula = pricePerArea ~ seasonDate, data = realEstateData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.409 -10.103   0.571   8.527  78.491
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        35.684      2.447  14.580   <2e-16 ***
## seasonDateSpring    3.325      2.716   1.224    0.222
## seasonDateSummer    2.165      2.883   0.751    0.453
## seasonDateWinter    1.964      2.663   0.738    0.461
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.63 on 410 degrees of freedom
## Multiple R-squared:  0.004255,   Adjusted R-squared:  -0.003031
## F-statistic: 0.584 on 3 and 410 DF,  p-value: 0.6258
```

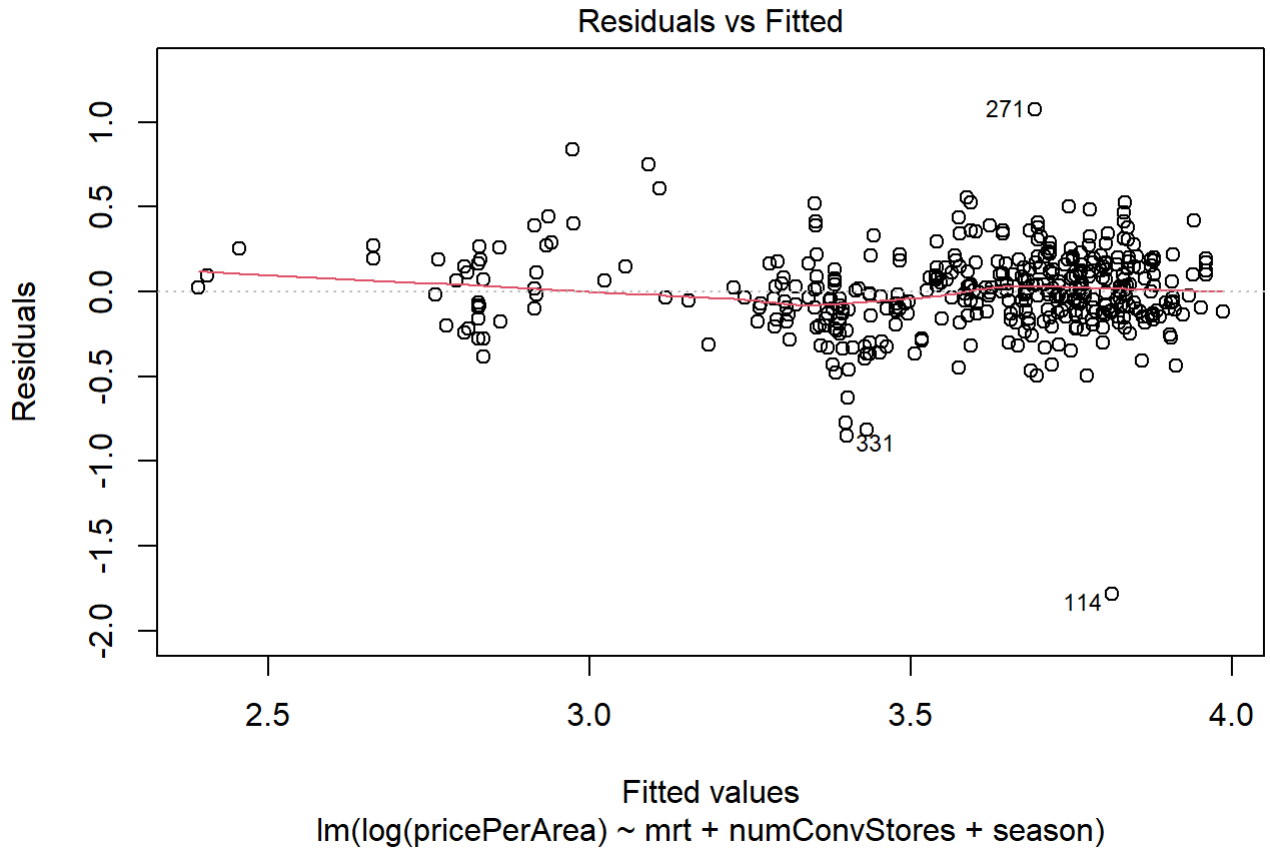
```
#Adding a season column to realEstateData
realEstateData['season'] = seasonDate
seasonModel1 = lm(pricePerArea~mrt + numConvStores + season, data = realEstateData)
seasonModel2 = lm(log(pricePerArea)~mrt + numConvStores + season, data = realEstateData)
summary(seasonModel1)
```

```
##
## Call:
## lm(formula = pricePerArea ~ mrt + numConvStores + season, data = realEstateData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.468  -5.804  -1.538   5.102  76.442
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   39.6014732  2.0619257  19.206  < 2e-16 ***
## mrt           -0.0057365  0.0004735 -12.116  < 2e-16 ***
## numConvStores  1.1634413  0.2025418   5.744 1.81e-08 ***
## seasonSpring   1.7418955  1.9250364   0.905    0.366
## seasonSummer  -0.9604116  2.0445234  -0.470    0.639
## seasonWinter  -1.3354666  1.8866507  -0.708    0.479
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.617 on 408 degrees of freedom
## Multiple R-squared:  0.5065, Adjusted R-squared:  0.5005
## F-statistic: 83.75 on 5 and 408 DF,  p-value: < 2.2e-16
```

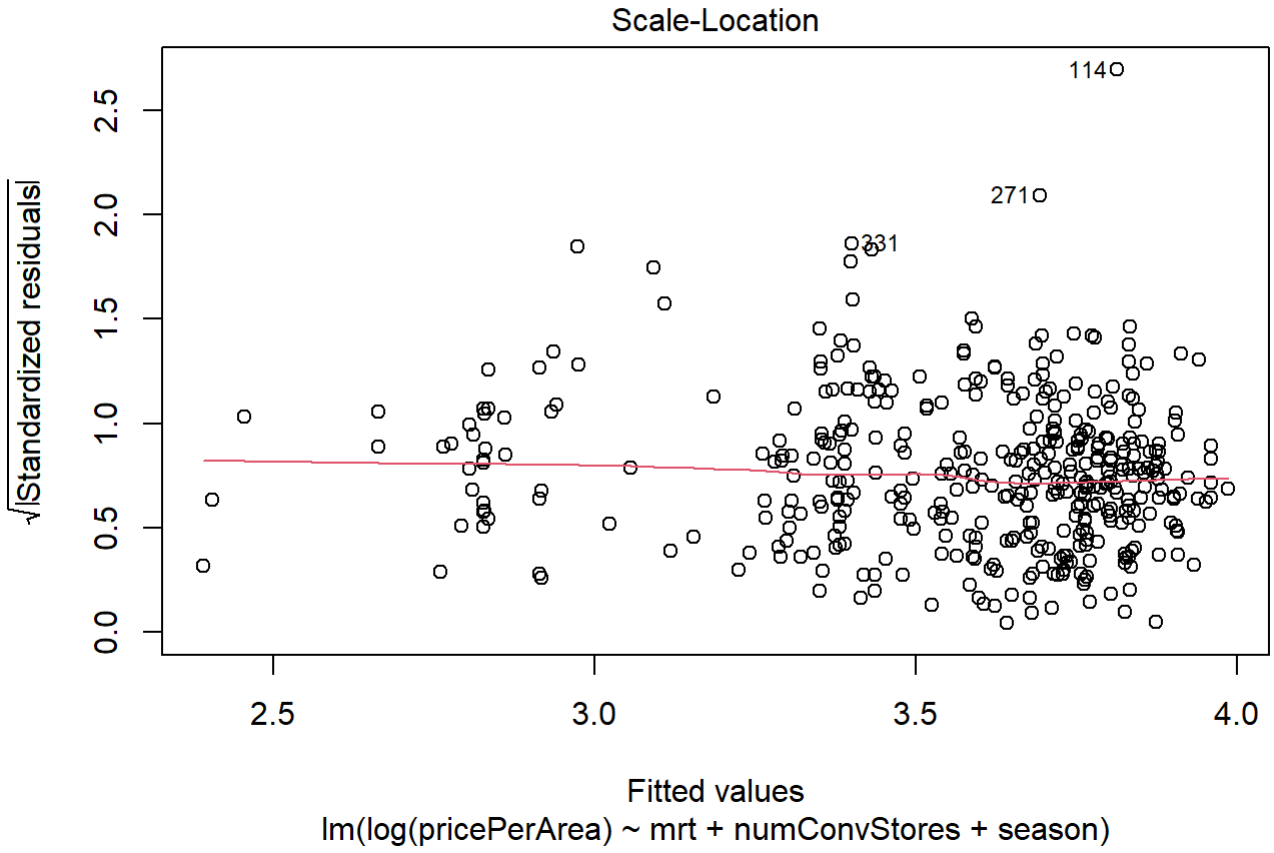
```
summary(seasonModel2)
```

```
##
## Call:
## lm(formula = log(pricePerArea) ~ mrt + numConvStores + season,
##     data = realEstateData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.78492 -0.12823 -0.00481  0.14118  1.07330
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.660e+00  5.294e-02  69.133  < 2e-16 ***
## mrt           -1.956e-04  1.216e-05 -16.090  < 2e-16 ***
## numConvStores  2.949e-02  5.200e-03   5.671 2.69e-08 ***
## seasonSpring   5.332e-02  4.942e-02   1.079    0.281
## seasonSummer  -2.875e-02  5.249e-02  -0.548    0.584
## seasonWinter  -3.302e-02  4.844e-02  -0.682    0.496
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2469 on 408 degrees of freedom
## Multiple R-squared:  0.609,  Adjusted R-squared:  0.6042
## F-statistic: 127.1 on 5 and 408 DF,  p-value: < 2.2e-16
```

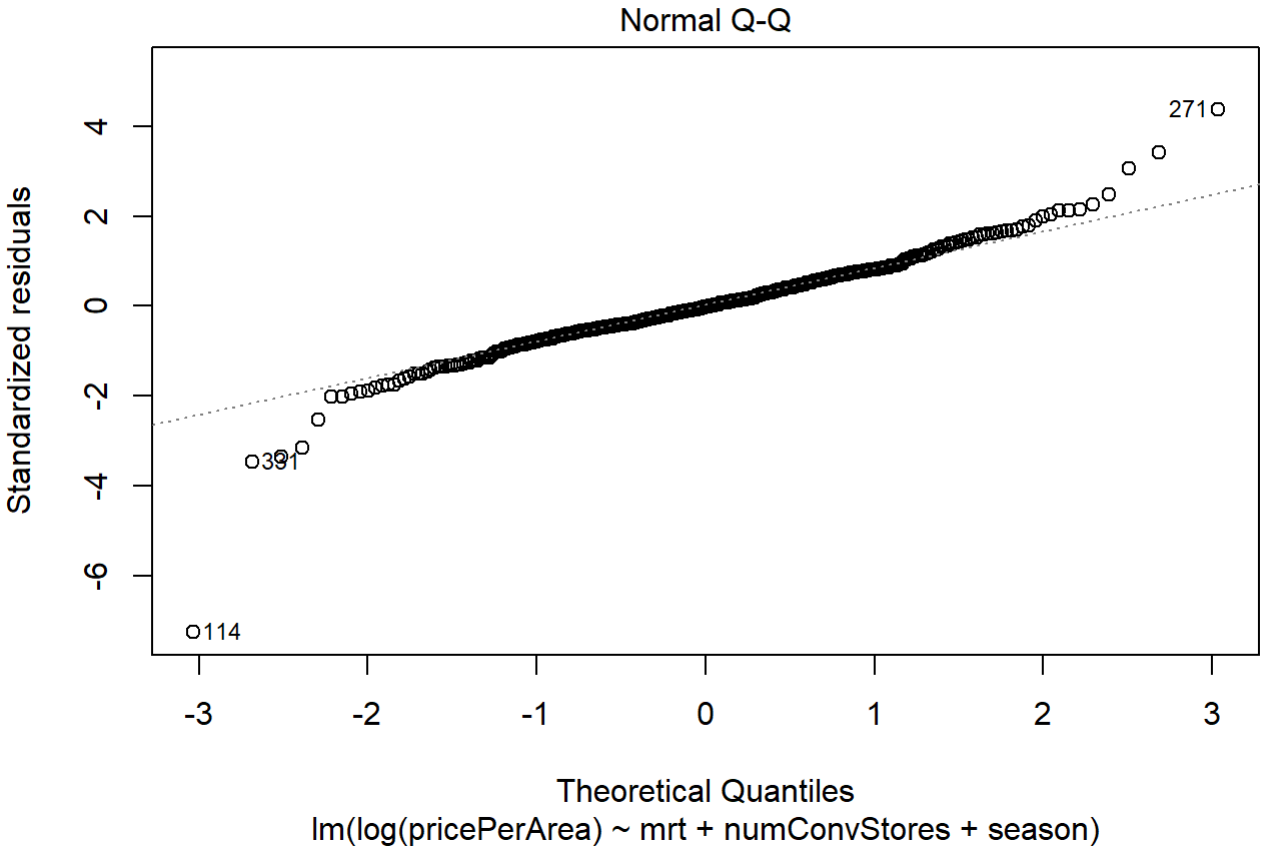
```
plot(seasonModel2, which = 1)
```



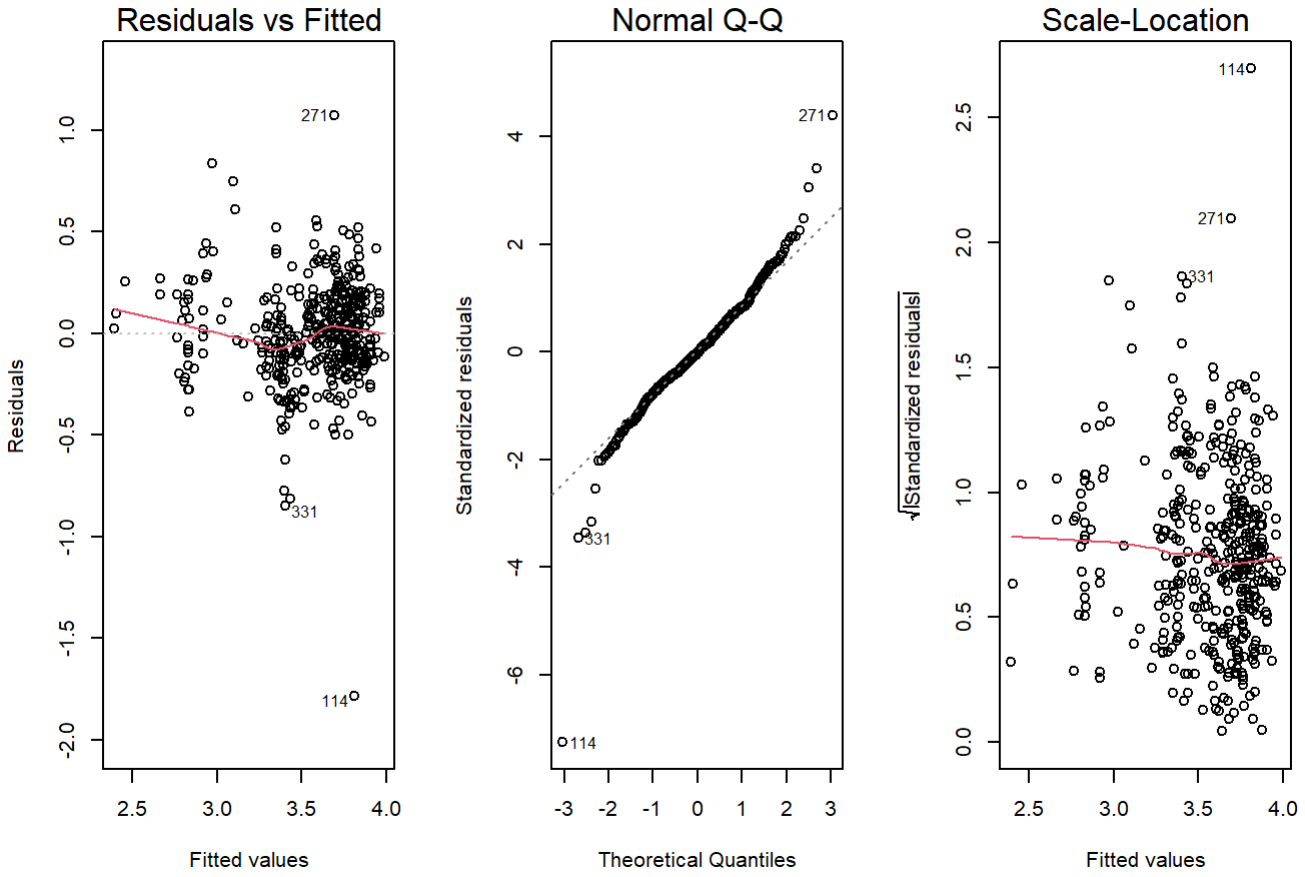
```
plot(seasonModel2, which = 3)
```




```
plot(seasonModel2, which = 2)
```




```
par(mfrow=c(1,3))
for(j in 1:3){
  plot(seasonModel2, which = j)
}
```



Appendix 2

```
Summer = 0
Winter = 0
Spring = 0
Fall = 0
y = 1
while (y <= length(seasonDate)) {
  if (seasonDate[y]=="Winter"){
    Winter = Winter + 1

  }
  else if (seasonDate[y] == "Spring"){
    Spring = Spring + 1
  }
  else if (seasonDate[y] == "Summer"){
    Summer = Summer + 1
  }
  else{
    Fall = Fall + 1
  }
  y = y + 1
}

Summer
```

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

```
Winter
```

```
## [1] 169
```

```
Spring
```

```
## [1] 134
```

```
Fall
```

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

Just for exploratory purposes, our dataset showed that house purchases were made during winter and spring.