Regression

Umaid Ahmed

2/18/2023

The source of the data taken in this book is: https://www.kaggle.com/datasets/shibumohapatra/house-price (https://www.kaggle.com/datasets/shibumohapatra/house-price)
(Note: Few of the code snippets used in this notebook is taken from Professor's notebooks on github repo)

Reading the data from a CSV file

First we read the data into a variable call df

```
df<-read.csv("California House Price.csv")
```

Dividing data into training and testing

We randomly sample the rows to get a vector i with row indices. This is used to divide into train and test sets.

```
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

Once we have loaded the data we would like to explore the data.

First, we would like to check the dimensions of the data so that we know that there is enough data to train and test a new model

```
dim(train)
```

```
## [1] 16512    10
```

As we can see, we have 10 predictors and 20640 rows of data which would be sufficient for training and testing purposes.

Now lets find out what are the column names of the data so that we can have a better understanding

```
colnames(train)
```

```
##  [1] "longitude"          "latitude"           "housing_median_age"
##  [4] "total_rooms"        "total_bedrooms"     "population"
##  [7] "households"         "median_income"      "ocean_proximity"
## [10] "median_house_value"
```

From the source where the data is taken the features are described as:

longitude (signed numeric - float) : Longitude value for the block in California, USA
latitude (numeric - float) : Latitude value for the block in California, USA
housing_median_age (numeric - int) : Median age of the house in the block
total_rooms (numeric - int) : Count of the total number of rooms (excluding bedrooms) in all houses in the block
total_bedrooms (numeric - float) : Count of the total number of bedrooms in all houses in the block
population (numeric - int) : Count of the total number of population in the block
households (numeric - int) : Count of the total number of households in the block
median_income (numeric - float) : Median of the total household income of all the houses in the block
ocean_proximity (numeric - categorical) : Type of the landscape of the block [Unique Values : 'NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND']
median_house_value (numeric - int) : Median of the household prices of all the houses in the block
(Note the description above is directly taken from source: https://www.kaggle.com/datasets/shibumohapatra/house-price (https://www.kaggle.com/datasets/shibumohapatra/house-price))

Finding out columns which have NA's

```
sapply(train, function(x) sum(is.na(x)))
```

```
##          longitude           latitude housing_median_age        total_rooms
##                  0                  0                  0                  0
##     total_bedrooms         population         households      median_income
##                165                  0                  0                  0
##    ocean_proximity median_house_value
##                  0                  0
```

str() can be used to see some values of the data

```
str(train)
```

```
## 'data.frame':    16512 obs. of  10 variables:
##  $ longitude         : num  -118 -118 -118 -118 -120 ...
##  $ latitude          : num  33.9 33.8 34 33.8 37.3 ...
##  $ housing_median_age: int  17 37 31 45 14 52 16 15 23 43 ...
##  $ total_rooms       : int  1145 2059 1014 944 2391 1114 2512 539 4247 88 ...
##  $ total_bedrooms    : int  209 349 252 178 451 206 356 71 835 21 ...
##  $ population        : int  499 825 1064 533 798 425 795 287 2357 119 ...
##  $ households        : int  202 334 247 193 308 207 353 66 823 19 ...
##  $ median_income     : num  4.64 4.06 2.42 3.48 3.09 ...
##  $ ocean_proximity   : chr  "<1H OCEAN" "<1H OCEAN" "<1H OCEAN" "NEAR OCEAN" ...
##  $ median_house_value: int  165500 225200 125500 206900 114600 175000 369100 305200 211300
67500 ...
```

summary() provide the summarized version of each column. It provides some useful information like mean, meadian min and max

```
summary(train, na.rm=TRUE)
```

```
##    longitude         latitude     housing_median_age  total_rooms
##  Min.   :-124.3   Min.   :32.54   Min.   : 1.00      Min.   :   12
##  1st Qu.:-121.8   1st Qu.:33.93   1st Qu.:18.00      1st Qu.: 1451
##  Median :-118.5   Median :34.26   Median :29.00      Median : 2134
##  Mean   :-119.6   Mean   :35.62   Mean   :28.59      Mean   : 2633
##  3rd Qu.:-118.0   3rd Qu.:37.71   3rd Qu.:37.00      3rd Qu.: 3142
##  Max.   :-114.5   Max.   :41.95   Max.   :52.00      Max.   :39320
##
##  total_bedrooms     population      households     median_income
##  Min.   :   3.0   Min.   :    3   Min.   :   2.0   Min.   : 0.4999
##  1st Qu.: 296.0   1st Qu.:  789   1st Qu.: 280.0   1st Qu.: 2.5656
##  Median : 434.0   Median : 1166   Median : 409.0   Median : 3.5387
##  Mean   : 537.2   Mean   : 1424   Mean   : 498.5   Mean   : 3.8770
##  3rd Qu.: 647.0   3rd Qu.: 1724   3rd Qu.: 604.0   3rd Qu.: 4.7511
##  Max.   :6210.0   Max.   :16305   Max.   :5358.0   Max.   :15.0001
##  NA's   :165
##  ocean_proximity    median_house_value
##  Length:16512       Min.   : 14999
##  Class :character   1st Qu.:120000
##  Mode  :character   Median :179300
##                     Mean   :206677
##                     3rd Qu.:264300
##                     Max.   :500001
##
```

Now we would like to see if there is any correlation between columns. This can help us by finding if the correlation of both variables are going up or going down or either one is going up and other is going down. Since the ocean_proximity is not a numeric value we would run into error so We can filter out the non numeric column and then try to find out the correlation

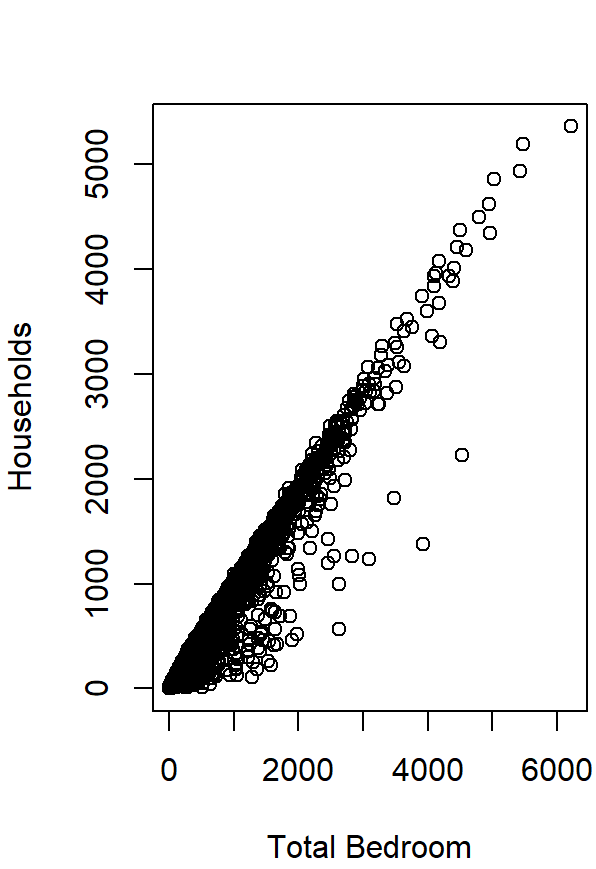
```
cor(train[, unlist(lapply(train, is.numeric))], use = "complete.obs")
```

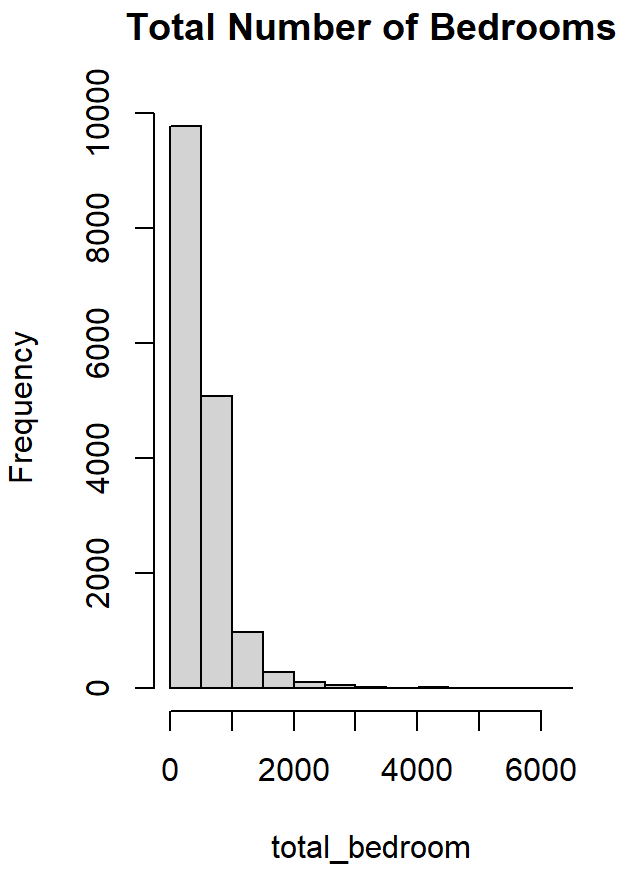
```
##                      longitude    latitude housing_median_age total_rooms
## longitude           1.00000000 -0.92509949        -0.10859294  0.04803226
## latitude           -0.92509949  1.00000000         0.01365934 -0.03936206
## housing_median_age -0.10859294  0.01365934         1.00000000 -0.36244710
## total_rooms         0.04803226 -0.03936206        -0.36244710  1.00000000
## total_bedrooms      0.07255227 -0.06987908        -0.32155442  0.92826989
## population          0.10605576 -0.11498946        -0.30314301  0.86139843
## households          0.05886095 -0.07429928        -0.30354203  0.91611529
## median_income      -0.01286493 -0.08349875        -0.11870783  0.19726543
## median_house_value -0.04324194 -0.14641085         0.10522992  0.13251351
##                    total_bedrooms   population  households median_income
## longitude              0.07255227  0.106055758  0.05886095  -0.012864929
## latitude              -0.06987908 -0.114989460 -0.07429928  -0.083498752
## housing_median_age    -0.32155442 -0.303143012 -0.30354203  -0.118707831
## total_rooms            0.92826989  0.861398428  0.91611529   0.197265432
## total_bedrooms         1.00000000  0.884169919  0.97854982  -0.011133144
## population             0.88416992  1.000000000  0.91473991   0.002433209
## households             0.97854982  0.914739906  1.00000000   0.010254855
## median_income         -0.01113314  0.002433209  0.01025486   1.000000000
## median_house_value     0.04622169 -0.028378921  0.06150795   0.691508861
##                    median_house_value
## longitude                  -0.04324194
## latitude                   -0.14641085
## housing_median_age          0.10522992
## total_rooms                 0.13251351
## total_bedrooms              0.04622169
## population                 -0.02837892
## households                  0.06150795
## median_income               0.69150886
## median_house_value          1.00000000
```

Correlation between two variable lies in the range 1 < cor < -1 . Correlations that are near 1 or -1 shows that there is a strong positive or negative correlation between them, while correlation near 0 shows that there is no correlation at all. Here we are ignoring correlation between latitude and longitude as they are the location of the blocks. We can see here that there is a strong correlation between total_rooms with total_bedrooms and household which is about 0.92 and 0.91 and there is also a good correlation between total_rooms and population which is 0.86. There is also a strong correlation between households with total_bedrooms, population and total_rooms which is 0.97, 0.91 and 0.91.

Data Visualization

```
opar <- par()
par(mfrow=c(1,2))
plot(train$total_bedrooms, train$households,
     xlab="Total Bedroom", ylab="Households", )
hist(train$total_bedrooms, main="Total Number of Bedrooms", xlab="total_bedroom")
```





The graph on the left shows a plot between total_bedrooms and households with total_bedrooms on the x-axis and households on the y-axis. As we can see clearly that there is a linear trend between total bedroom and households. The graph on the right shows an histogram of total_bedrooms showing much that mostly total number of bedroom lies between 0 and 1000 in a certain block in California.

Building a Linear Regression Model

Here we are building a simple Linear Regression model which uses one predictor

```
lm1 <- lm(total_bedrooms~households, data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = total_bedrooms ~ households, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -248.24  -29.07  -10.77    8.93 2438.55
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.379174   1.114135  -1.238    0.216
## households   1.079820   0.001778 607.277   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 86.21 on 16345 degrees of freedom
##   (165 observations deleted due to missingness)
## Multiple R-squared:  0.9576, Adjusted R-squared:  0.9576
## F-statistic: 3.688e+05 on 1 and 16345 DF,  p-value: < 2.2e-16
```

The first thing the summary shows as is the **formula** that we use for the linear regression. Here, total_bedroom as a function of households is the formula.

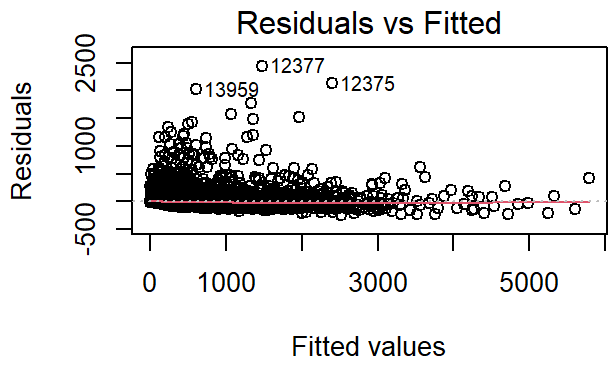
The second thing it is showing is **Residual errors** which indicate how far off the values were from the regression line.

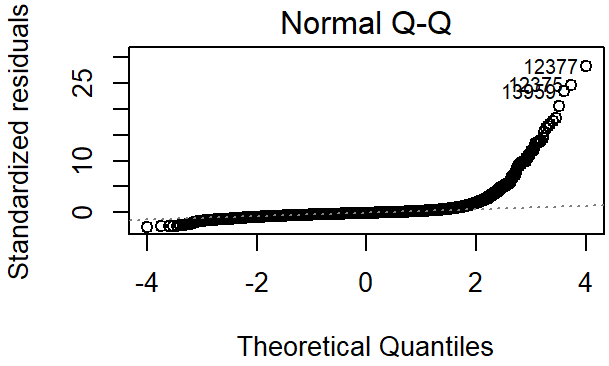
Next it shows the **Coefficients**:
a. The *Estimate* tells us that the for every households total_bedrooms increase by 1.079820
b. The *std. Error* (Standard Error) is the estimate that how much variance is between them.
c. The *t value* shows the number the estimate coefficient is from 0 from standard deviation. If the value is 0 it shows that there is no relationship between the target and the predictor.
d. The *p-value* is used to prove that if null hypothesis is true or not. If the p-value is < .05 it means it is a good fit.

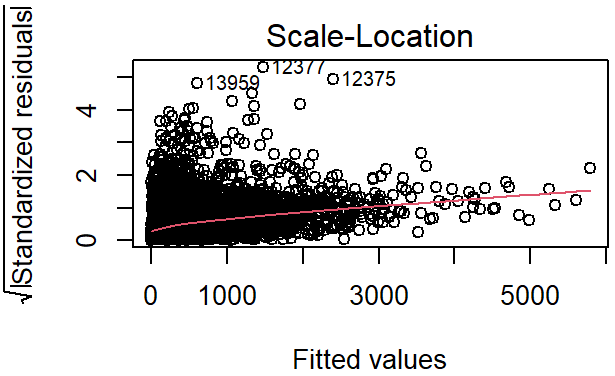
The last part shows the statistics of the overall model and tells us if the model is good. **Multiple R-square** value and **Adjusted R-square value** is 0.9576 which is closer to 1 and the **p-value** is closer to 0 showing that the model here overall is a good model.

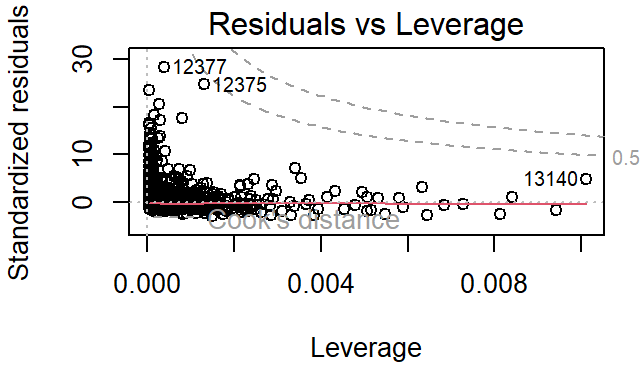
Plotting the Residuals

```
par(mfrow=c(2,2))
plot(lm1)
```









1. For **Residuals vs Fitted**, we are looking for the residuals to be evenly distributed around the red line. Since the model have large amount of data we can see that some of the residuals are plotted evenly but other have a higher y-values
2. The second plot **Normal Q-Q** looks for normal distribution in which residuals are along the dotted line. In this graph we can see that residuals follow the straight line till 2 on the x-axis till it falls off.
3. The third plot **Scale-Location** shows us if the residuals are spread equally along the ranges. It can be seen that the residuals varies alot. This also means that the data doesn't have the property of homoscedasticity
4. The fourth plot is **Residuals vs Leverage** which helps us to find influential cases in the data. Influential cases are those which are listed beyond the *Cook's distance* (the dotted line). It can be seen that 12377, 12375 and 13140 are some unusual cases

Multiple Linear Regression Model

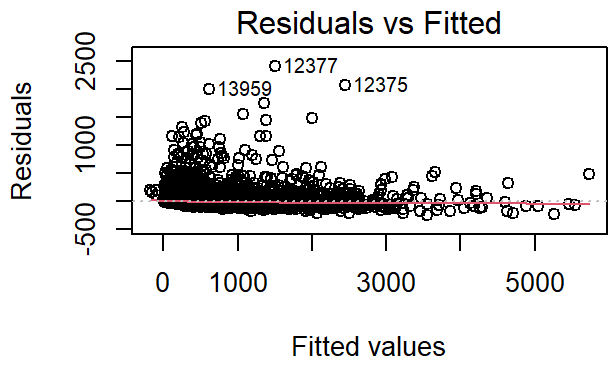
Linear regression model with more than one predictor is called Multiple Linear Regression. For multiple linear regression we would like to see the effect of households and population on total_bedrooms

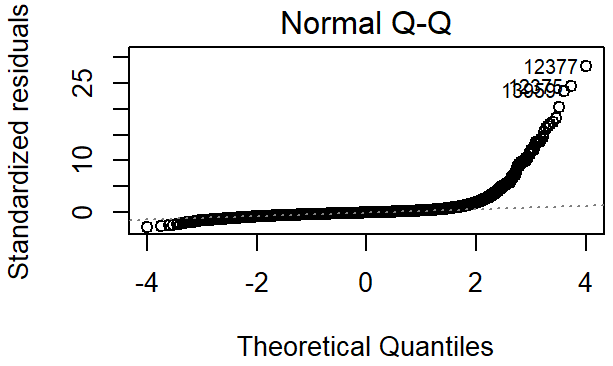
```
mlm <- lm(total_bedrooms~households+population, data=train)
summary(mlm)
```

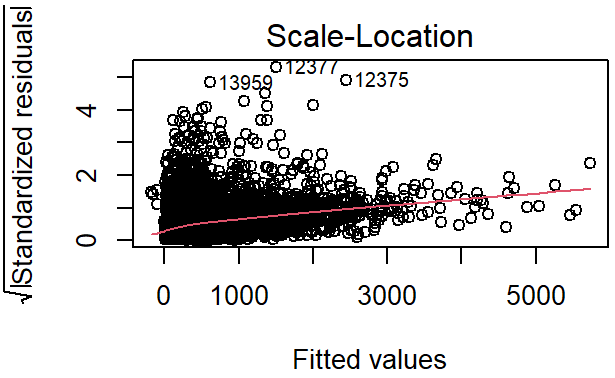
```
##
## Call:
## lm(formula = total_bedrooms ~ households + population, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -254.05  -29.38  -10.75    9.40 2416.42
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.345546   1.116107   1.206    0.228
## households   1.147517   0.004363 263.026   <2e-16 ***
## population  -0.025608   0.001510 -16.963   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85.47 on 16344 degrees of freedom
##   (165 observations deleted due to missingness)
## Multiple R-squared:  0.9583, Adjusted R-squared:  0.9583
## F-statistic: 1.878e+05 on 2 and 16344 DF,  p-value: < 2.2e-16
```

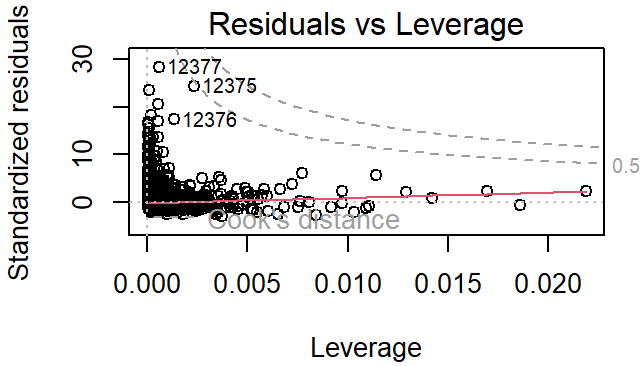
Multiple Linear Regression Model Residual Plot

```
par(mfrow=c(2,2))
plot(mlm)
```









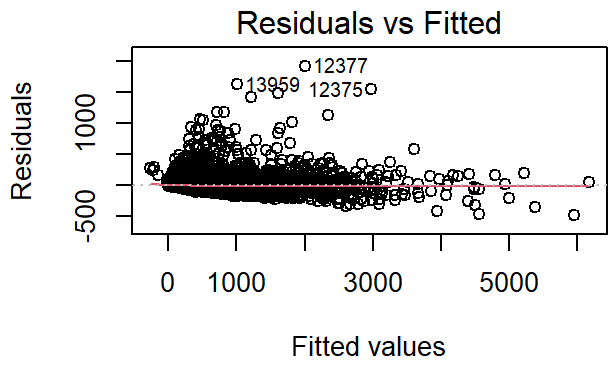
Linear Regression with multiple predictors

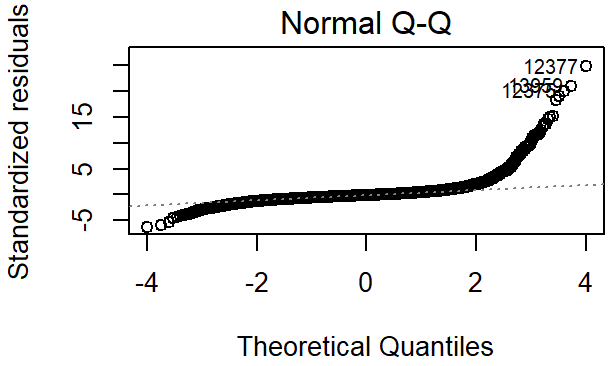
```
lm3 <- lm(total_bedrooms ~ households + population + total_rooms, data=train)
summary(lm3)
```

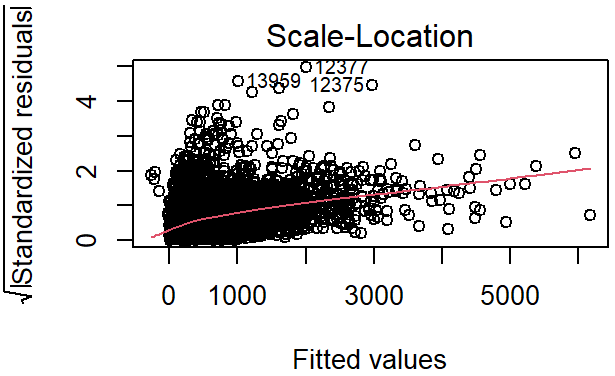
```
##
## Call:
## lm(formula = total_bedrooms ~ households + population + total_rooms,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -486.11  -32.33   -8.01   17.86 1919.62
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.5489742  1.0170451   1.523    0.128
## households   0.9638117  0.0050900 189.353   <2e-16 ***
## population  -0.0372097  0.0013902 -26.766   <2e-16 ***
## total_rooms  0.0409690  0.0007089  57.794   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 77.88 on 16343 degrees of freedom
##   (165 observations deleted due to missingness)
## Multiple R-squared:  0.9654, Adjusted R-squared:  0.9654
## F-statistic: 1.519e+05 on 3 and 16343 DF,  p-value: < 2.2e-16
```

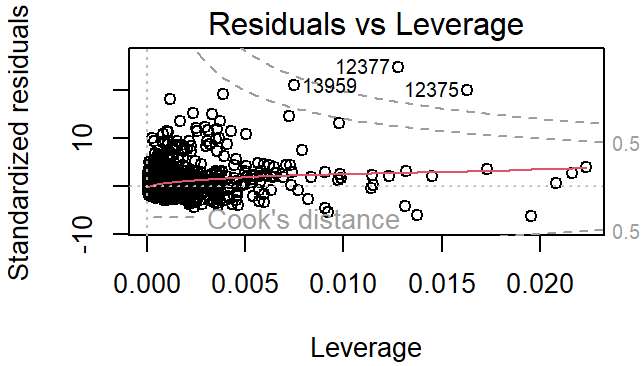
Residual Plots for third model

```
par(mfrow=c(2,2))
plot(lm3)
```









Comparing The Results of different Models

```
anova(lm1, mlm, lm3)
```

	Res.Df <dbl>	RSS <dbl>	Df <dbl>	Sum of Sq <dbl>	F <dbl>	Pr(>F) <dbl>
1	16345	121491191	*NA*	*NA*	*NA*	*NA*
2	16344	119389189	1	2102001	346.5472	1.48026e-76
3	16343	99129367	1	20259822	3340.1432	0.00000e+00

3 rows

If we focus on the third part of the summary of each model we can clearly see that lm3 is the better model. lm3 have Adjusted R-squared of 0.9654 while lm1 had 0.9576 and mlm had a value of 0.9583. In addition to this lm3 had the lowest Residual standard error with the value of 77.88 while lm1 and mlm had 86.21 and 85.47 respectively. We can also see the difference by using anova function. lm3 has the lowest Res.Df and RSS.

Prediction And Evaluation on the Test Data

```
pred1 <- predict(lm1, newdata = test, use="complete.obs")
cor1 <- cor(pred1, test$total_bedrooms, use="complete.obs")
mse1 <- mean((pred1 - test$total_bedrooms)^2, na.rm=TRUE)

print(paste('correlation for lm1:', cor1))
```

```
## [1] "correlation for lm1: 0.984122807468871"
```

```
print(paste('mse of lm1:', mse1))
```

```
## [1] "mse of lm1: 5900.70328975162"
```

```
pred2 <- predict(mlm, newdata = test, use="complete.obs")
cor2 <- cor(pred2, test$total_bedrooms, use="complete.obs")
mse2 <- mean((pred2 - test$total_bedrooms)^2, na.rm=TRUE)

print(paste('correlation for lm2:', cor2))
```

```
## [1] "correlation for lm2: 0.98438036232588"
```

```
print(paste('mse of lm2:', mse2))
```

```
## [1] "mse of lm2: 5810.46997625596"
```

```
pred3 <- predict(lm3, newdata = test, use="complete.obs")
cor3 <- cor(pred3, test$total_bedrooms, use="complete.obs")
mse3 <- mean((pred3 - test$total_bedrooms)^2, na.rm=TRUE)

print(paste('correlation for lm3:', cor3))
```

```
## [1] "correlation for lm3: 0.986567882429882"
```

```
print(paste('mse of lm3:', mse3))
```

```
## [1] "mse of lm3: 4999.2953955097"
```

As it is displayed that the correlation is increasing from lm1 to mlm to lm3 and the mean square error is decreasing from lm1 to mlm to lm3. This is due to the fact that more predictors were added to the model, so the model was trained on more data and it was able to make better predictions.