# HM10_final

Sedreh

5/29/2019

```
library("ggplot2")
library(data.table)
library(dplyr)
library(GGally)
library(DataExplorer)
library(plyr)
```

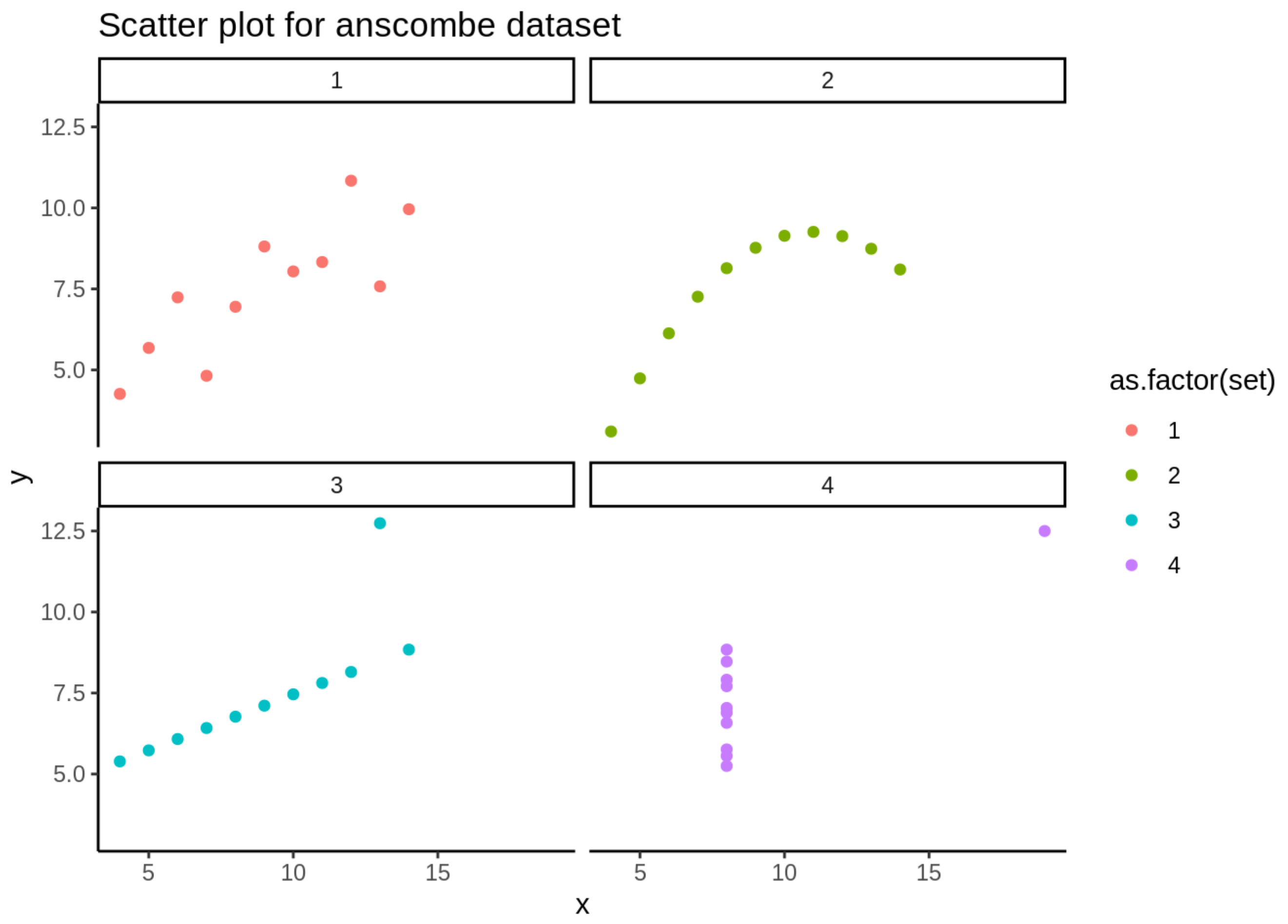
```
#Anscombe’s quartet comprises four datasets

anscombe <- readRDS("/home/manu/Downloads/anscombe.rds")
head(anscombe)
```

```
##    x    y set
## 1 10 8.04   1
## 2  8 6.95   1
## 3 13 7.58   1
## 4  9 8.81   1
## 5 11 8.33   1
## 6 14 9.96   1
```

```
str(anscombe)
```

```
## 'data.frame':    44 obs. of  3 variables:
##  $ x  : num  10 8 13 9 11 14 6 4 12 7 ...
##  $ y  : num  8.04 6.95 7.58 8.81 8.33 ...
##  $ set: num  1 1 1 1 1 1 1 1 1 1 ...
```

```
#Scatter plot facetted by set
p <- ggplot(anscombe, aes(x, y, color = as.factor(set))) +
  geom_point()+
  facet_wrap(.~set)+
  ggtitle("Scatter plot for anscombe dataset ")+
  theme_classic()
p
```



```
#1: possitive linear relationship
#2: the relationship is non_linear(but correlation is 0.8 and knowing the value of x
 will give us the value of y).there is no statistical noise! it is just non_linear
#3:except the outlier,the relation between x and y is positive and it is linear but o
utlier will cause lower correlation coefficient
#4:no relationship! knowing x does not tell anything about y! also we have outlier
```

```
#Summary calculation (mean, sd), Pearson’s correlation by set, and non-parametric, an
d p-value
#Correlation is a statistical measure that suggests the level of linear dependence be
tween two variables

# summary_calc <- anscombe %>%
#   group_by(set) %>%
#   summarize(mean.x = mean(x), sd.x = sd(x), mean.y = mean(y), sd.y = sd(y),cor(x,
y))
#
# summary_calc

summary_calc <- anscombe %>%
group_by(set) %>%
mutate(mean.x = mean(x), sd.x = sd(x), mean.y = mean(y), sd.y = sd(y))
summary_calc
```

```
## # A tibble: 44 x 7
## # Groups:   set [4]
##        x     y   set mean.x  sd.x mean.y  sd.y
##    <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl>
##  1    10  8.04     1      9  3.20   7.50  1.96
##  2     8  6.95     1      9  3.20   7.50  1.96
##  3    13  7.58     1      9  3.20   7.50  1.96
##  4     9  8.81     1      9  3.20   7.50  1.96
##  5    11  8.33     1      9  3.20   7.50  1.96
##  6    14  9.96     1      9  3.20   7.50  1.96
##  7     6  7.24     1      9  3.20   7.50  1.96
##  8     4  4.26     1      9  3.20   7.50  1.96
##  9    12 10.8      1      9  3.20   7.50  1.96
## 10     7  4.82     1      9  3.20   7.50  1.96
## # … with 34 more rows
```

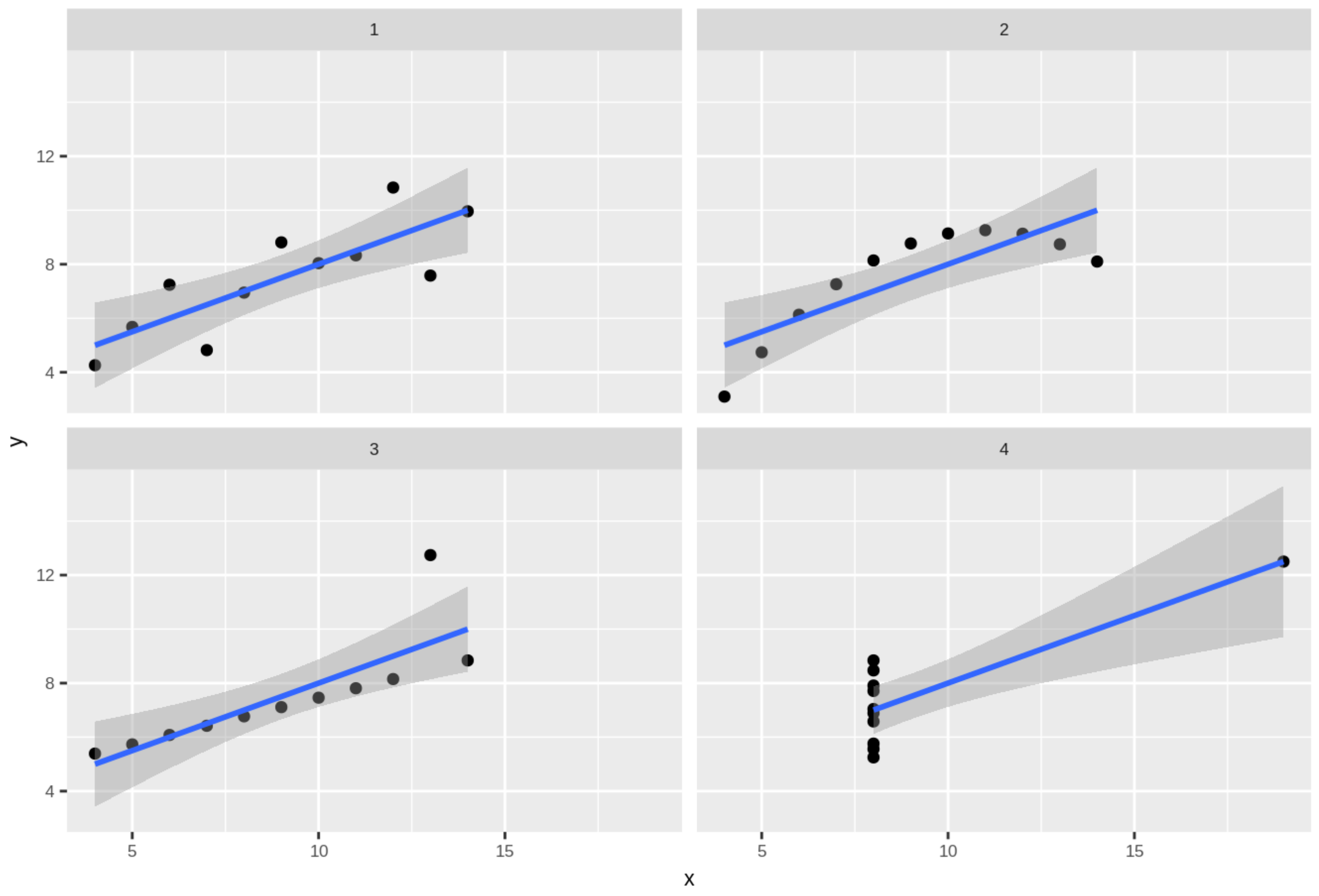
```
#non-parametric, and p-value
ddply(anscombe, "set", summarise, corr=cor(x, y),
cor_spear = cor(x,y, method = "spearman"),
p.value = cor.test(x,y)$p.value)
```

```
##   set      corr cor_spear     p.value
## 1   1 0.8164205 0.8181818 0.002169629
## 2   2 0.8162365 0.6909091 0.002178816
## 3   3 0.8162867 0.9909091 0.002176305
## 4   4 0.8165214 0.5000000 0.002164602
```

```
#There are two main options for data aggregation:
#built-in functions, often referred to as the apply family of functions, the plyr add
-on package
#The heart of plyr is a set a functions with names like this: XYply where X specifies
what sort of input you're giving and Y specifies the sort of output you want.

    # a = array, where matrices and vectors are important special cases
    # d = data.frame
    # l = list

# The usage is very similar across these functions. Here are the main arguments:
# .data is the first argument = the input
# the next argument specifies how to split up the input into bits; it is does not exi
st when the input is a list, because the pieces are obviously the list components
# then comes the function and further arguments needed to describe the computation to
be applied to the bits
# here ddply() will accept a data.frame, group it be "set", computes the correlation,
then returns the results as a data.frame.
```

```
q <- ggplot(anscombe, aes(x, y), color = as.factor(set)) +
  geom_point()+
  facet_wrap(.~set)+
  theme(text = element_text(size = 8))+
  geom_smooth(method="lm")+
  ggtitle("Scatter plot for anscombe dataset ")
q
```

Scatter plot for anscombe dataset



```
#gray shading is standard error assiciating with the line!
```

```
#airquality data set
# -200 is missing values in the data

airquality <- read.csv("//home/manu/Downloads/AirQualityUCI/AirQualityUCI.csv", heade
r=TRUE, sep=";", na.strings=c("-200", "-200,0"))
head(airquality)
```

```
##         Date     Time CO.GT. PT08.S1.CO. NMHC.GT. C6H6.GT. PT08.S2.NMHC.
## 1 10/03/2004 18.00.00    2,6        1360      150     11,9          1046
## 2 10/03/2004 19.00.00      2        1292      112      9,4           955
## 3 10/03/2004 20.00.00    2,2        1402       88      9,0           939
## 4 10/03/2004 21.00.00    2,2        1376       80      9,2           948
## 5 10/03/2004 22.00.00    1,6        1272       51      6,5           836
## 6 10/03/2004 23.00.00    1,2        1197       38      4,7           750
##   NOx.GT. PT08.S3.NOx. NO2.GT. PT08.S4.NO2. PT08.S5.O3.    T   RH     AH
## 1     166         1056     113         1692        1268 13,6 48,9 0,7578
## 2     103         1174      92         1559         972 13,3 47,7 0,7255
## 3     131         1140     114         1555        1074 11,9 54,0 0,7502
## 4     172         1092     122         1584        1203 11,0 60,0 0,7867
## 5     131         1205     116         1490        1110 11,2 59,6 0,7888
## 6      89         1337      96         1393         949 11,2 59,2 0,7848
##    X X.1
## 1 NA  NA
## 2 NA  NA
## 3 NA  NA
## 4 NA  NA
## 5 NA  NA
## 6 NA  NA
```

```
airquality  <- data.frame(airquality)

# Drop the columns of the dataframe with NA value using select function in dplyr
airquality <- select (airquality,-c(X,X.1))
head(airquality)
```

```
##         Date     Time CO.GT. PT08.S1.CO. NMHC.GT. C6H6.GT. PT08.S2.NMHC.
## 1 10/03/2004 18.00.00    2,6        1360      150     11,9          1046
## 2 10/03/2004 19.00.00      2        1292      112      9,4           955
## 3 10/03/2004 20.00.00    2,2        1402       88      9,0           939
## 4 10/03/2004 21.00.00    2,2        1376       80      9,2           948
## 5 10/03/2004 22.00.00    1,6        1272       51      6,5           836
## 6 10/03/2004 23.00.00    1,2        1197       38      4,7           750
##   NOx.GT. PT08.S3.NOx. NO2.GT. PT08.S4.NO2. PT08.S5.O3.    T   RH     AH
## 1     166         1056     113         1692        1268 13,6 48,9 0,7578
## 2     103         1174      92         1559         972 13,3 47,7 0,7255
## 3     131         1140     114         1555        1074 11,9 54,0 0,7502
## 4     172         1092     122         1584        1203 11,0 60,0 0,7867
## 5     131         1205     116         1490        1110 11,2 59,6 0,7888
## 6      89         1337      96         1393         949 11,2 59,2 0,7848
```

```
# apply function anyNA() on all columns of airquality dataset and we can see that alm
ost all columns (not Date,Time) have NA value!
```

```
str(airquality)
```

```
## 'data.frame':    9471 obs. of  15 variables:
##  $ Date         : Factor w/ 392 levels "","01/01/2005",..: 116 116 116 116 116 116
129 129 129 129 ...
##  $ Time         : Factor w/ 25 levels "","00.00.00",..: 20 21 22 23 24 25 2 3 4 5
...
##  $ CO.GT.       : Factor w/ 103 levels "","0,1","0,2",..: 33 26 29 29 18 14 14 11
10 7 ...
##  $ PT08.S1.CO.  : int  1360 1292 1402 1376 1272 1197 1185 1136 1094 1010 ...
##  $ NMHC.GT.     : int  150 112 88 80 51 38 31 31 24 19 ...
##  $ C6H6.GT.     : Factor w/ 408 levels "","0,1","0,2",..: 40 403 399 401 373 326 2
36 233 124 18 ...
##  $ PT08.S2.NMHC.: int  1046 955 939 948 836 750 690 672 609 561 ...
##  $ NOx.GT.      : int  166 103 131 172 131 89 62 62 45 NA ...
##  $ PT08.S3.NOx. : int  1056 1174 1140 1092 1205 1337 1462 1453 1579 1705 ...
##  $ NO2.GT.      : int  113 92 114 122 116 96 77 76 60 NA ...
##  $ PT08.S4.NO2. : int  1692 1559 1555 1584 1490 1393 1333 1333 1276 1235 ...
##  $ PT08.S5.O3.  : int  1268 972 1074 1203 1110 949 733 730 620 501 ...
##  $ T            : Factor w/ 437 levels "","-0,1","-0,2",..: 67 64 50 41 43 43 44 3
8 38 34 ...
##  $ RH           : Factor w/ 754 levels "","10,0","10,2",..: 376 364 427 487 483 47
9 455 487 484 489 ...
##  $ AH           : Factor w/ 6684 levels "","0,1847","0,1862",..: 1897 1728 1854 20
57 2067 2046 1910 1963 1936 1864 ...
```

```r
# here decimal is comma and comma is decimal

  sub_clean_commas <- function(x) {gsub(",",".", x)}

# T, RH, AH, C6H6.GT., CO.GT. these are factors due to commas

airquality$T <- sub_clean_commas(airquality$T)
airquality$RH <- sub_clean_commas(airquality$RH)
airquality$AH <- sub_clean_commas(airquality$AH)
airquality$C6H6.GT. <- sub_clean_commas(airquality$C6H6.GT.)
airquality$CO.GT. <- sub_clean_commas(airquality$CO.GT.)

str(airquality)
```

```
## 'data.frame':    9471 obs. of  15 variables:
##  $ Date         : Factor w/ 392 levels "","01/01/2005",..: 116 116 116 116 116 116
129 129 129 129 ...
##  $ Time         : Factor w/ 25 levels "","00.00.00",..: 20 21 22 23 24 25 2 3 4 5
...
##  $ CO.GT.       : chr  "2.6" "2" "2.2" "2.2" ...
##  $ PT08.S1.CO.  : int  1360 1292 1402 1376 1272 1197 1185 1136 1094 1010 ...
##  $ NMHC.GT.     : int  150 112 88 80 51 38 31 31 24 19 ...
##  $ C6H6.GT.     : chr  "11.9" "9.4" "9.0" "9.2" ...
##  $ PT08.S2.NMHC.: int  1046 955 939 948 836 750 690 672 609 561 ...
##  $ NOx.GT.      : int  166 103 131 172 131 89 62 62 45 NA ...
##  $ PT08.S3.NOx. : int  1056 1174 1140 1092 1205 1337 1462 1453 1579 1705 ...
##  $ NO2.GT.      : int  113 92 114 122 116 96 77 76 60 NA ...
##  $ PT08.S4.NO2. : int  1692 1559 1555 1584 1490 1393 1333 1333 1276 1235 ...
##  $ PT08.S5.O3.  : int  1268 972 1074 1203 1110 949 733 730 620 501 ...
##  $ T            : chr  "13.6" "13.3" "11.9" "11.0" ...
##  $ RH           : chr  "48.9" "47.7" "54.0" "60.0" ...
##  $ AH           : chr  "0.7578" "0.7255" "0.7502" "0.7867" ...
```

```
airquality <- airquality %>%
  select('C6H6.GT.', 'PT08.S1.CO.','PT08.S2.NMHC.','PT08.S3.NOx.','PT08.S4.NO2.','PT0
8.S5.O3.','CO.GT.', 'NMHC.GT.','NOx.GT.', 'NO2.GT.', 'T', 'RH', 'AH')
head(airquality)
```

```
##   C6H6.GT. PT08.S1.CO. PT08.S2.NMHC. PT08.S3.NOx. PT08.S4.NO2. PT08.S5.O3.
## 1     11.9        1360          1046         1056         1692        1268
## 2      9.4        1292           955         1174         1559         972
## 3      9.0        1402           939         1140         1555        1074
## 4      9.2        1376           948         1092         1584        1203
## 5      6.5        1272           836         1205         1490        1110
## 6      4.7        1197           750         1337         1393         949
##   CO.GT. NMHC.GT. NOx.GT. NO2.GT.    T   RH     AH
## 1    2.6      150     166     113 13.6 48.9 0.7578
## 2      2      112     103      92 13.3 47.7 0.7255
## 3    2.2       88     131     114 11.9 54.0 0.7502
## 4    2.2       80     172     122 11.0 60.0 0.7867
## 5    1.6       51     131     116 11.2 59.6 0.7888
## 6    1.2       38      89      96 11.2 59.2 0.7848
```

```
# convert all variables into numeric:
airquality <- data.matrix(airquality)
airquality <- data.frame(airquality)
str(airquality)
```

```
## 'data.frame':    9471 obs. of  13 variables:
##  $ C6H6.GT.     : num  11.9 9.4 9 9.2 6.5 4.7 3.6 3.3 2.3 1.7 ...
##  $ PT08.S1.CO.  : num  1360 1292 1402 1376 1272 ...
##  $ PT08.S2.NMHC.: num  1046 955 939 948 836 ...
##  $ PT08.S3.NOx. : num  1056 1174 1140 1092 1205 ...
##  $ PT08.S4.NO2. : num  1692 1559 1555 1584 1490 ...
##  $ PT08.S5.O3.  : num  1268 972 1074 1203 1110 ...
##  $ CO.GT.       : num  2.6 2 2.2 2.2 1.6 1.2 1.2 1 0.9 0.6 ...
##  $ NMHC.GT.     : num  150 112 88 80 51 38 31 31 24 19 ...
##  $ NOx.GT.      : num  166 103 131 172 131 89 62 62 45 NA ...
##  $ NO2.GT.      : num  113 92 114 122 116 96 77 76 60 NA ...
##  $ T            : num  13.6 13.3 11.9 11 11.2 11.2 11.3 10.7 10.7 10.3 ...
##  $ RH           : num  48.9 47.7 54 60 59.6 59.2 56.8 60 59.7 60.2 ...
##  $ AH           : num  0.758 0.726 0.75 0.787 0.789 ...
```

```
# the data is ready for analysis now
```

```
sum(is.na(airquality))
```

```
## [1] 18183
```

```
airquality_summary <- colSums(is.na(airquality))
head(airquality_summary)
```

```
##      C6H6.GT.   PT08.S1.CO. PT08.S2.NMHC.  PT08.S3.NOx.  PT08.S4.NO2. 
##           480           480           480           480           480 
##   PT08.S5.O3. 
##           480
```

```
# We can see that NMHC.GT. column has too many NA's we must delete it, or else we wil
l have little data to work with
airquality$NMHC.GT. <- NULL
airquality_clean <- na.omit(airquality)
head(airquality_clean)
```

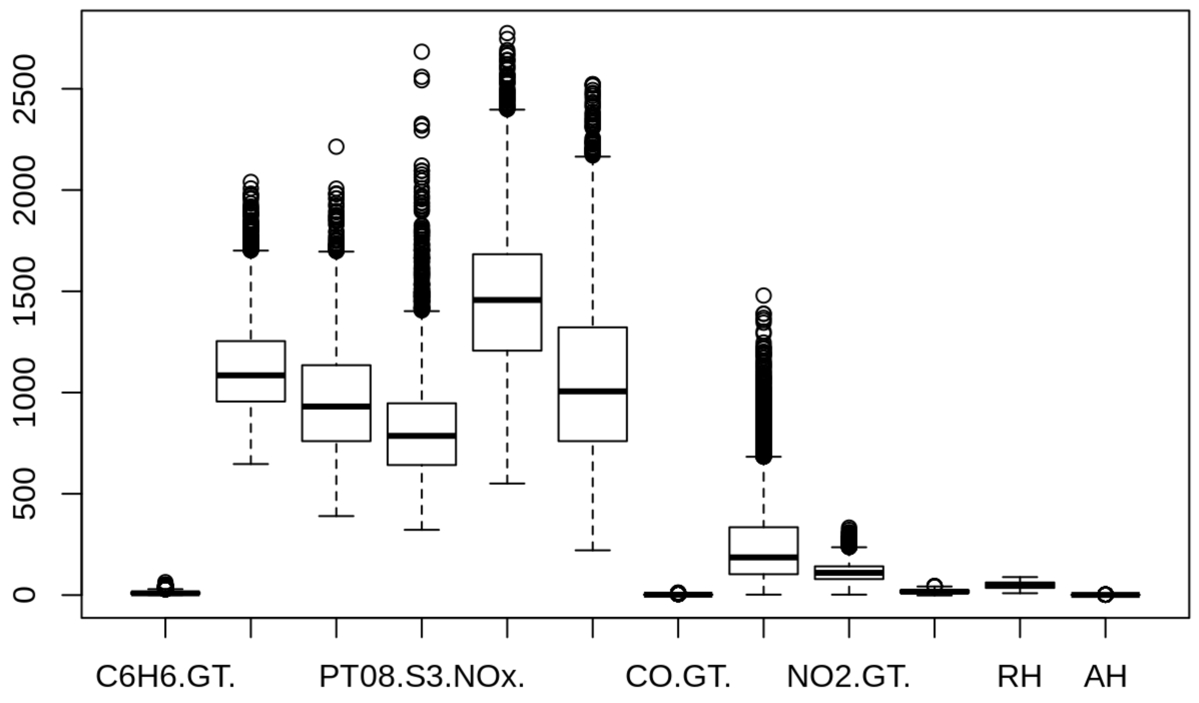
```
##   C6H6.GT. PT08.S1.CO. PT08.S2.NMHC. PT08.S3.NOx. PT08.S4.NO2. PT08.S5.O3.
## 1     11.9        1360          1046         1056         1692        1268
## 2      9.4        1292           955         1174         1559         972
## 3      9.0        1402           939         1140         1555        1074
## 4      9.2        1376           948         1092         1584        1203
## 5      6.5        1272           836         1205         1490        1110
## 6      4.7        1197           750         1337         1393         949
##   CO.GT. NOx.GT. NO2.GT.    T   RH     AH
## 1    2.6     166     113 13.6 48.9 0.7578
## 2    2.0     103      92 13.3 47.7 0.7255
## 3    2.2     131     114 11.9 54.0 0.7502
## 4    2.2     172     122 11.0 60.0 0.7867
## 5    1.6     131     116 11.2 59.6 0.7888
## 6    1.2      89      96 11.2 59.2 0.7848
```

```
any(is.na(airquality_clean))
```

```
## [1] FALSE
```

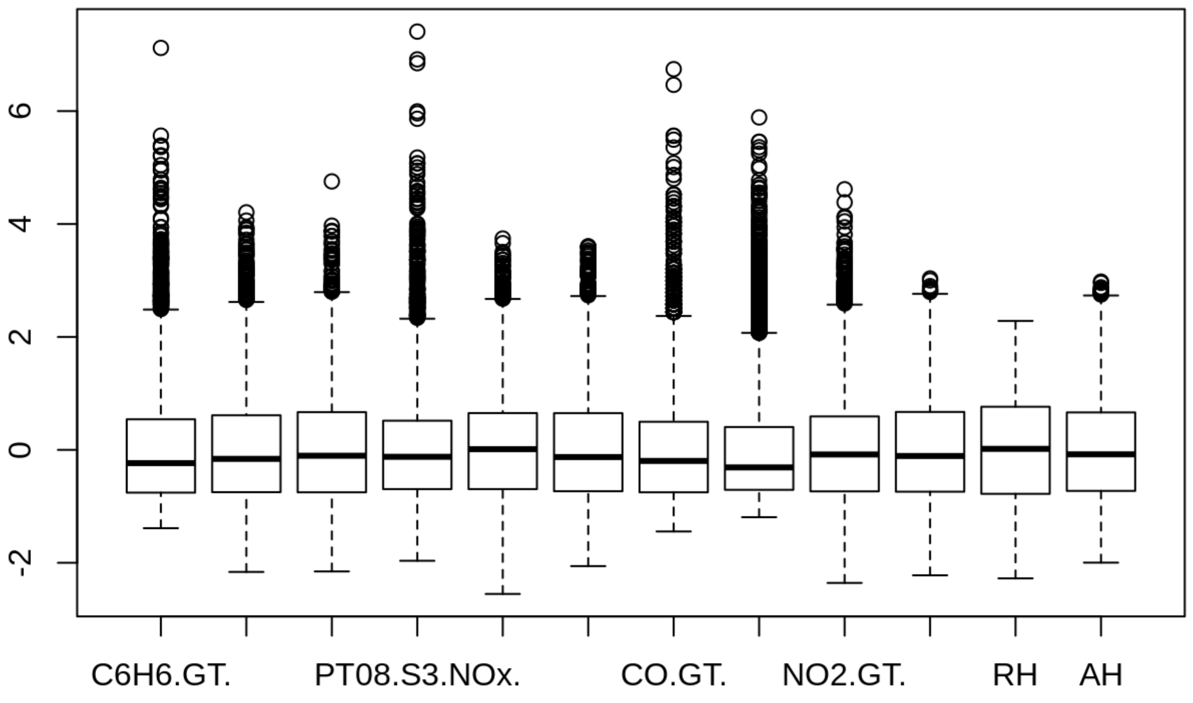
```
#################Explore each variable independently###################
```

```
#Box plot:we should always perform an exploratory analysis of our variables before an
y formal modeling. This will give us a sense of our variable's distributions, any out
liers, and any patterns that might be useful when contructing our eventual model.
plot <- boxplot(airquality_clean)
```



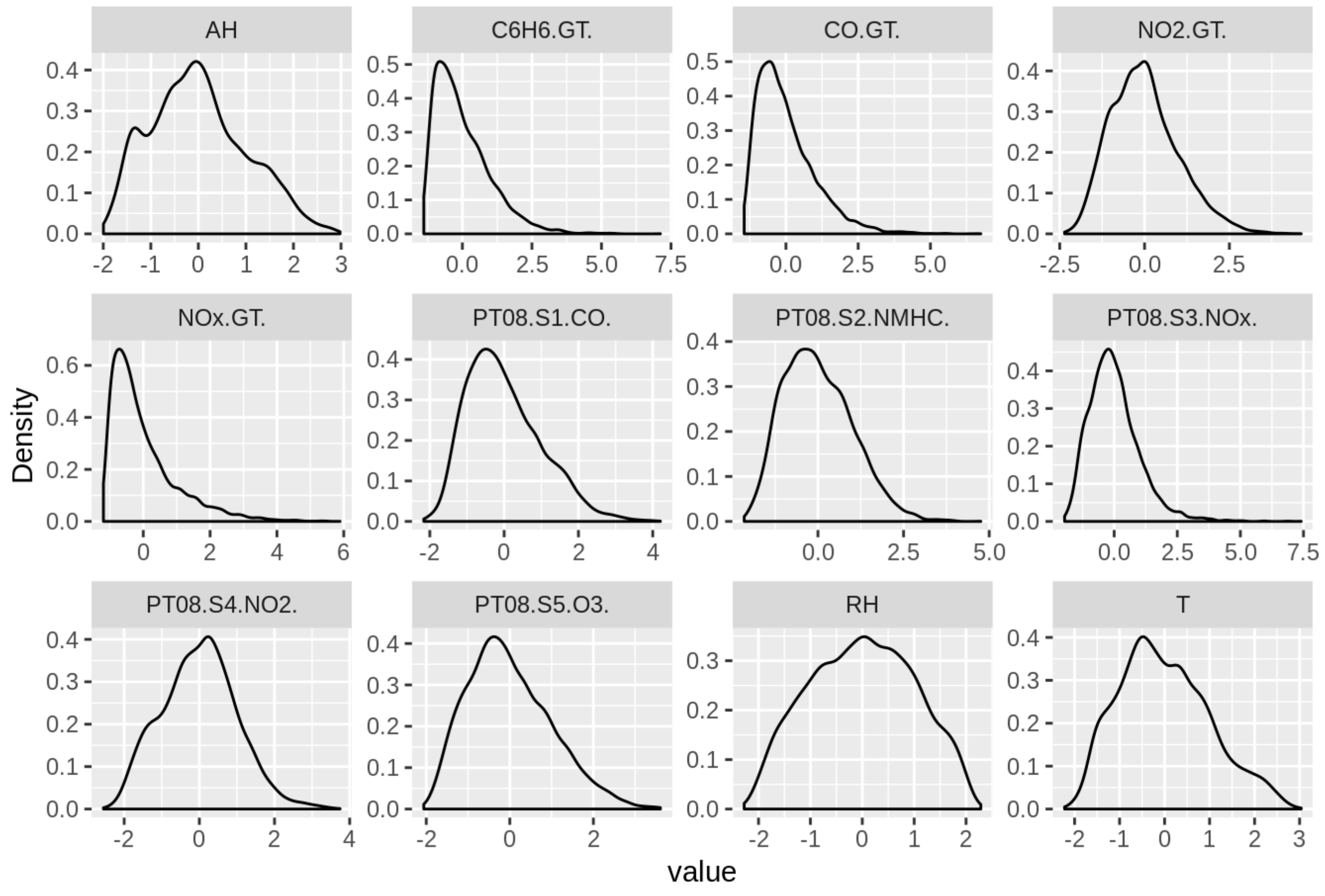
```
# We can see that the data is out of scale
```

```
airquality_clean1 <- scale(airquality_clean, center = TRUE, scale = TRUE)

boxplot(airquality_clean1)
```

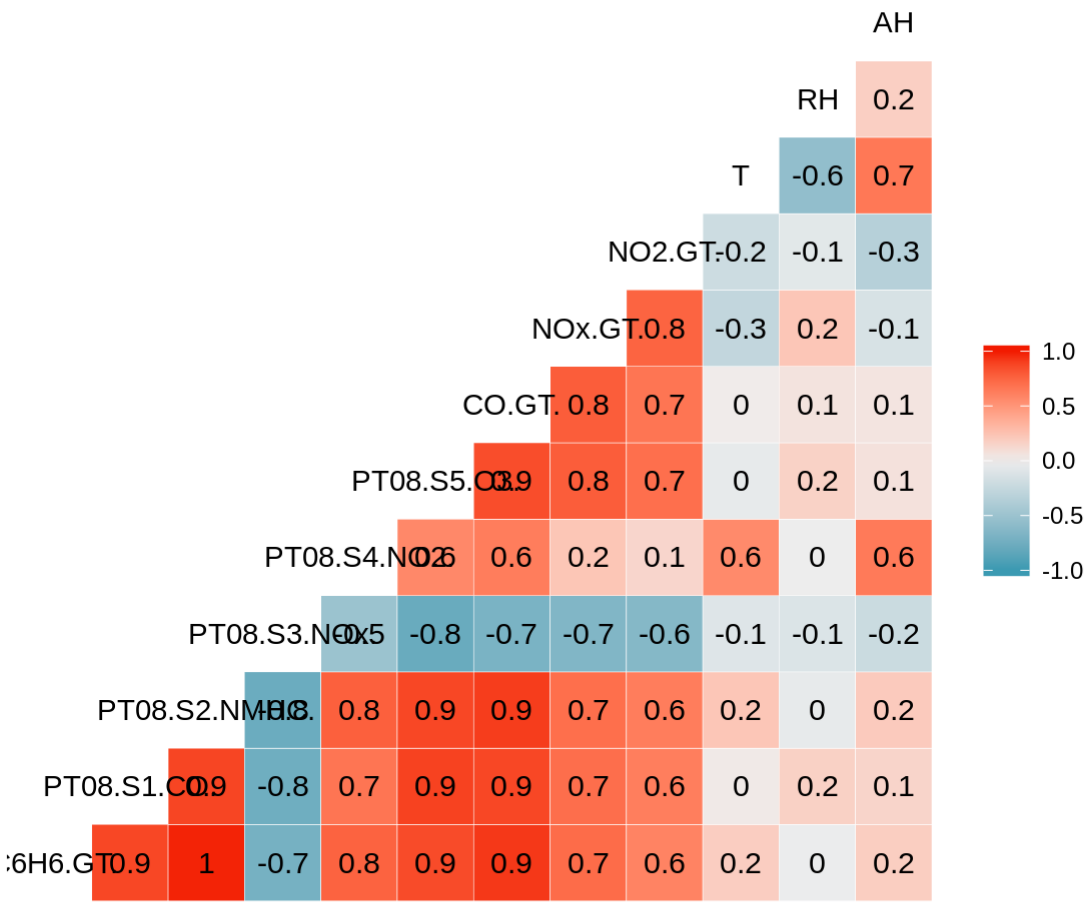


```
# Some vaiable are in 10s but others are in 1000S this means that the variables in 10
s will have no effect on the model even if we include/ exclude them, this is why we s
hould bring them to the same scale to remove this effect.(normalization)


# Now the data is in comparable scale.
```

```
#Density plot: To see the distribution of the predictor variable. Ideally, a close to
normal distribution (a bell shaped curve), without being skewed to the left or right
 is preferred. Let see how to make each one of them.
plot_density(airquality_clean1)
```



```
#cross correlation
ggcorr(airquality_clean1, palette = "RdBu", label = TRUE)
```



```
#simple linear models with each predictor, check assumptions
#The summary for the linear model provides information regarding the quality of the m
odel:

airquality_clean1 <- data.frame(airquality_clean1)

mod2 <- lm(C6H6.GT. ~ NO2.GT., data=airquality_clean1)
summary(mod2)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ NO2.GT., data = airquality_clean1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3922 -0.4785 -0.0821  0.3463  5.7960
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.078e-15  9.574e-03    0.00        1
## NO2.GT.      6.032e-01  9.574e-03   63.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7976 on 6939 degrees of freedom
## Multiple R-squared:  0.3639, Adjusted R-squared:  0.3638
## F-statistic:  3970 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
mod3 <-lm(C6H6.GT. ~ PT08.S1.CO., data=airquality_clean1)
summary(mod3)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ PT08.S1.CO., data = airquality_clean1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6014 -0.3027 -0.0245  0.2619  6.0196
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.947e-17  5.758e-03     0.0        1
## PT08.S1.CO.  8.774e-01  5.759e-03   152.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4797 on 6939 degrees of freedom
## Multiple R-squared:  0.7699, Adjusted R-squared:  0.7699
## F-statistic: 2.322e+04 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
mod4 <-lm(C6H6.GT. ~ PT08.S4.NO2., data=airquality_clean1)
summary(mod4)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ PT08.S4.NO2., data = airquality_clean1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3317 -0.4794 -0.0616  0.3911  6.7388
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.111e-16  7.776e-03    0.00        1
## PT08.S4.NO2.  7.618e-01  7.777e-03   97.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6479 on 6939 degrees of freedom
## Multiple R-squared:  0.5803, Adjusted R-squared:  0.5803
## F-statistic:  9596 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
mod5 <-lm(C6H6.GT. ~ PT08.S5.O3., data=airquality_clean1)
summary(mod5)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ PT08.S5.O3., data = airquality_clean1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1249 -0.3125 -0.0050  0.2829  4.4642
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.385e-16  6.102e-03     0.0        1
## PT08.S5.O3.  8.612e-01  6.103e-03   141.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5084 on 6939 degrees of freedom
## Multiple R-squared:  0.7416, Adjusted R-squared:  0.7415
## F-statistic: 1.991e+04 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
mod6 <-lm(C6H6.GT. ~ PT08.S3.NOx., data=airquality_clean1)
summary(mod6)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ PT08.S3.NOx., data = airquality_clean1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2151 -0.4953 -0.1085  0.3089  5.9497
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.443e-16  8.259e-03    0.00        1
## PT08.S3.NOx. -7.257e-01  8.259e-03  -87.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.688 on 6939 degrees of freedom
## Multiple R-squared:  0.5267, Adjusted R-squared:  0.5266
## F-statistic:  7721 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
mod7 <-lm(C6H6.GT. ~ PT08.S2.NMHC., data=airquality_clean1)
summary(mod7)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ PT08.S2.NMHC., data = airquality_clean1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.15638 -0.12878 -0.06695  0.06782  2.44678
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -6.795e-16  2.223e-03     0.0        1
## PT08.S2.NMHC.  9.827e-01  2.223e-03   442.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1852 on 6939 degrees of freedom
## Multiple R-squared:  0.9657, Adjusted R-squared:  0.9657
## F-statistic: 1.954e+05 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
mod8 <-lm(C6H6.GT. ~ NOx.GT., data=airquality_clean1)
summary(mod8)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ NOx.GT., data = airquality_clean1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7014 -0.5321 -0.1491  0.3738  5.2240
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.374e-16  8.351e-03    0.00        1
## NOx.GT.      7.183e-01  8.352e-03   86.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6957 on 6939 degrees of freedom
## Multiple R-squared:  0.516,  Adjusted R-squared:  0.5159
## F-statistic:  7398 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
mod9 <-lm(C6H6.GT. ~ CO.GT., data=airquality_clean1)
summary(mod9)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ CO.GT., data = airquality_clean1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0359 -0.2123 -0.0212  0.2005  6.5270
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.218e-16  4.412e-03     0.0        1
## CO.GT.       9.300e-01  4.412e-03   210.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3676 on 6939 degrees of freedom
## Multiple R-squared:  0.8649, Adjusted R-squared:  0.8649
## F-statistic: 4.443e+04 on 1 and 6939 DF,  p-value: < 2.2e-16
```

```
#intercept = B0'
#slop = B1'
```

```
#Check all variables for their p-values

summary(mod2)$coefficients[4]
```

```
## [1] 0.009574462
```

```
summary(mod3)$coefficients[4]
```

```
## [1] 0.005758711
```

```
summary(mod4)$coefficients[4]
```

```
## [1] 0.007776722
```

```
summary(mod5)$coefficients[4]
```

```
## [1] 0.006102531
```

```
summary(mod6)$coefficients[4]
```

```
## [1] 0.008259098
```

```
summary(mod7)$coefficients[4]
```

```
## [1] 0.002223035
```

```
summary(mod8)$coefficients[4]
```

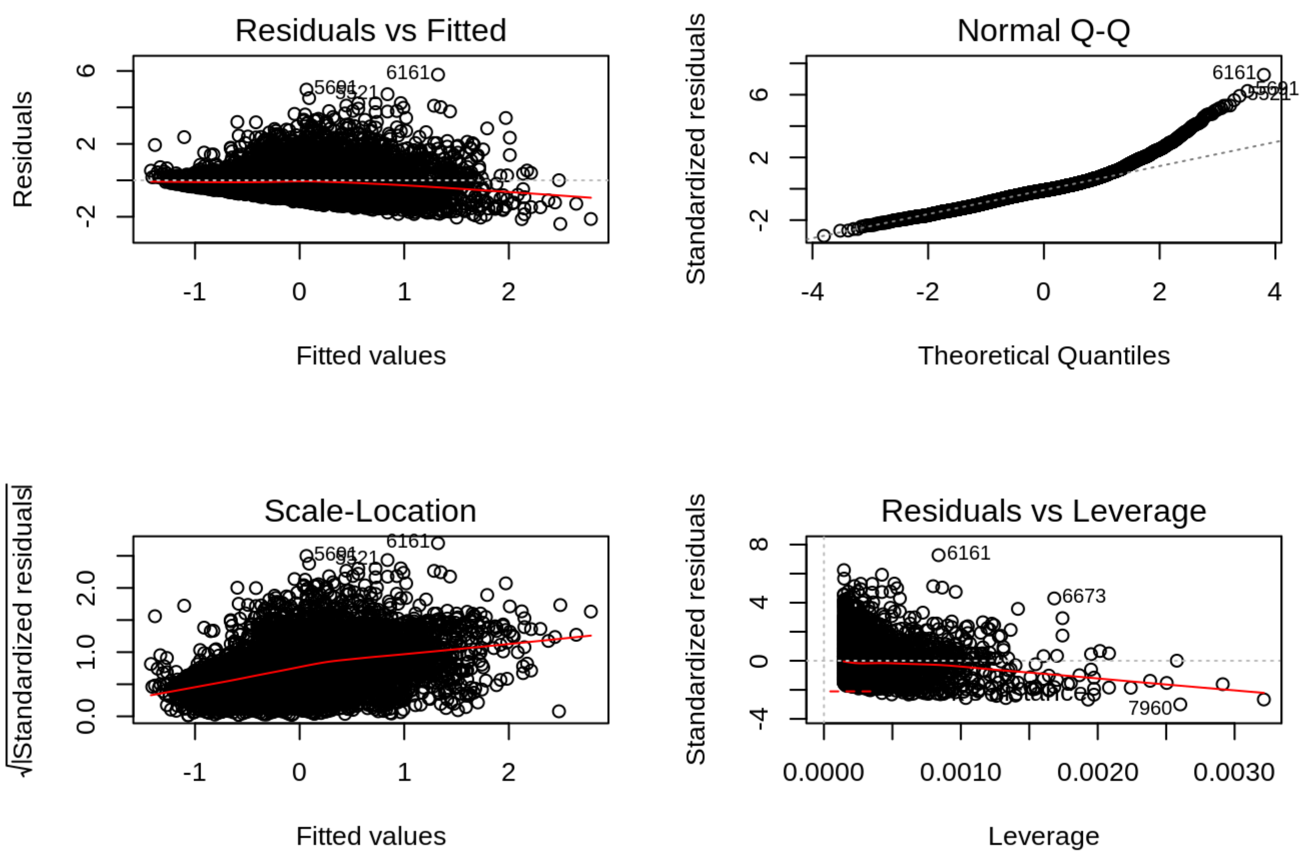
```
## [1] 0.008351533
```

```
summary(mod9)$coefficients[4]
```

```
## [1] 0.004412188
```

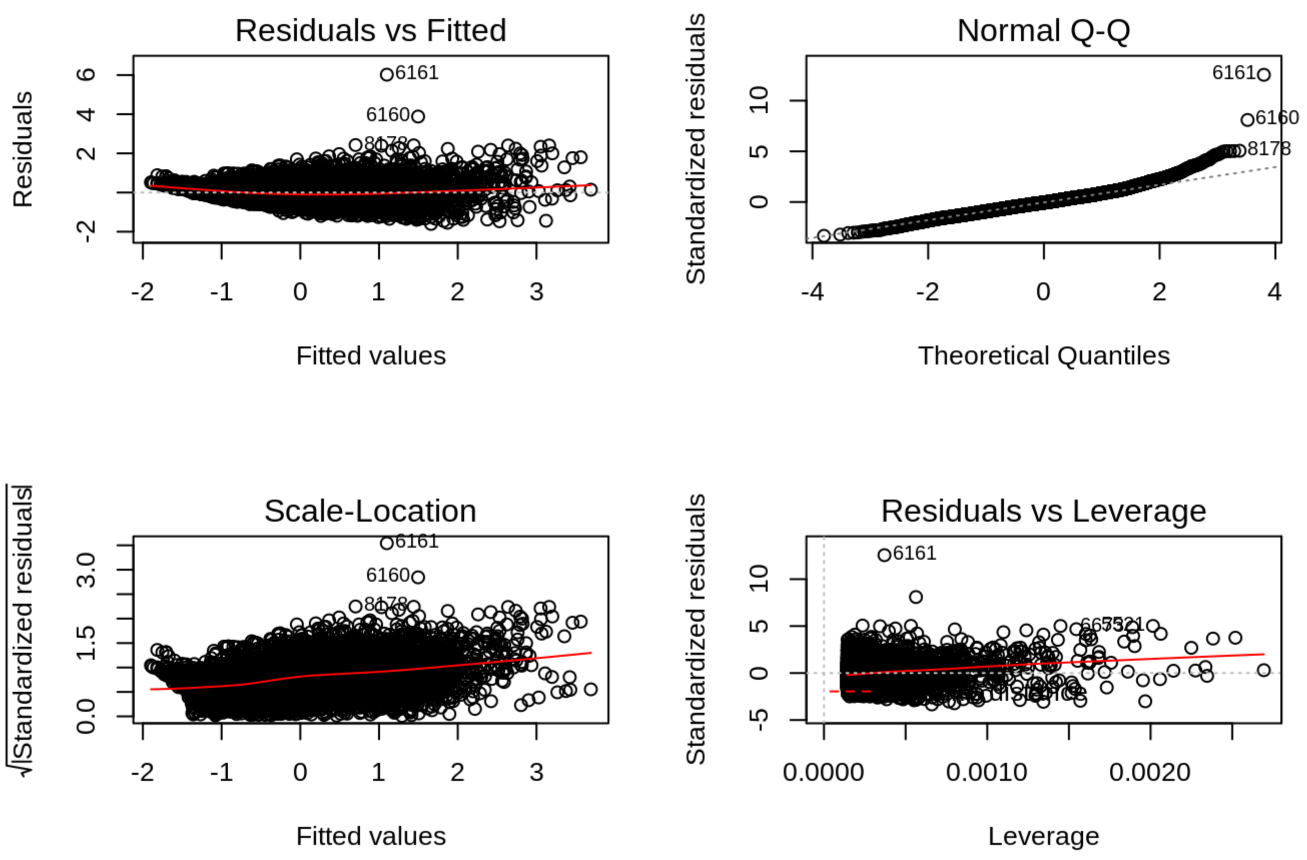
```
# options(digits=3)
# mod9[["coefficients"]][["(Intercept)"]] ==
# summary(mod9)$coefficients[4]
#  # 3 is coefficients, 4 is value
# summary(mod9)$coefficients[4]
```

```
#############  Diagnostic plots for the model ###############

#Assumptions
#y-values or the errors are independent
#variation of observations around the regression line (residual SE) is constant(homos
cedasticity)!
#for given value of x, y values (or the errors) are normaly distribured

par(mfrow=c(2,2))
plot(mod2)
```

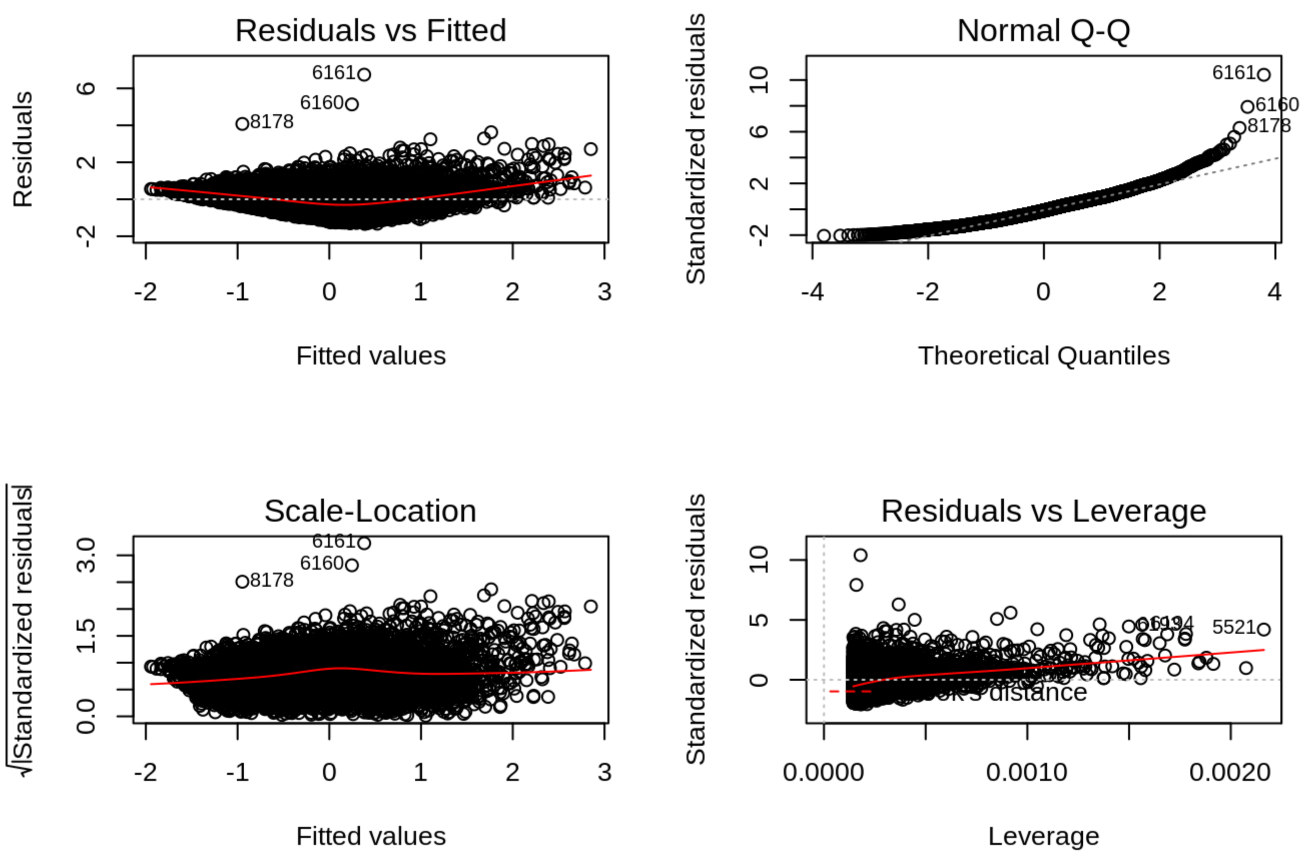


```
#x: predicted y values(y')
#y:errors or reseduals
#1)Residuals vs fitted values: if the linearity assumption is met we should see no pa
tern here! the red line should be flat
#2) quantile quantile plot: y = observed and standardize residuals, x: theoritical re
seduals.expected reseduals if errors/reseduals normally distributed.points shuld fall
in diognal line!
#3)other 2 plots also can help to see non-linearities and non-constant varients and s
o on!

#y' value for each data point!fitted velues generate the residual! residual is the di
fference between the actual observed value of the response variable and the expected
 value of the response according to our model!
```

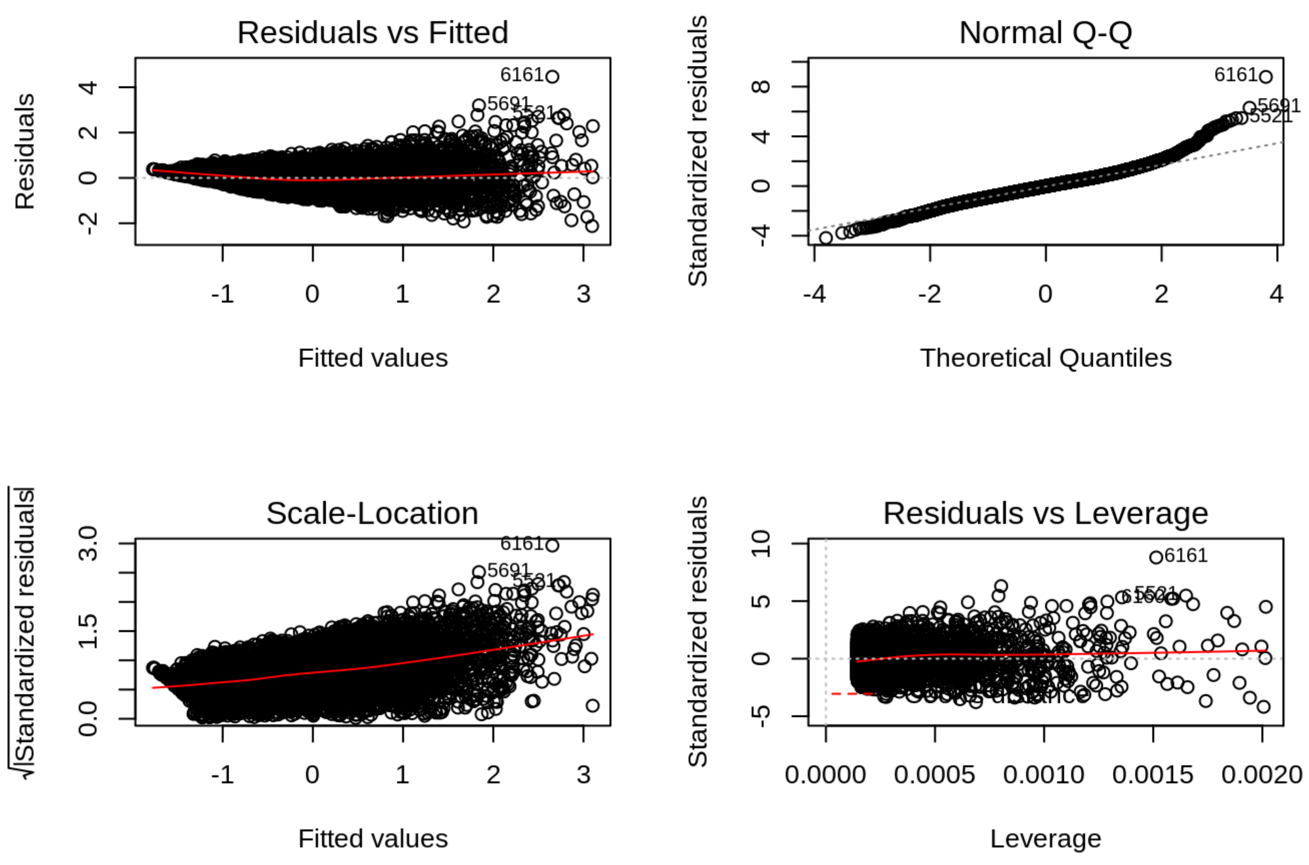
```
par(mfrow=c(2,2))
plot(mod3)
```



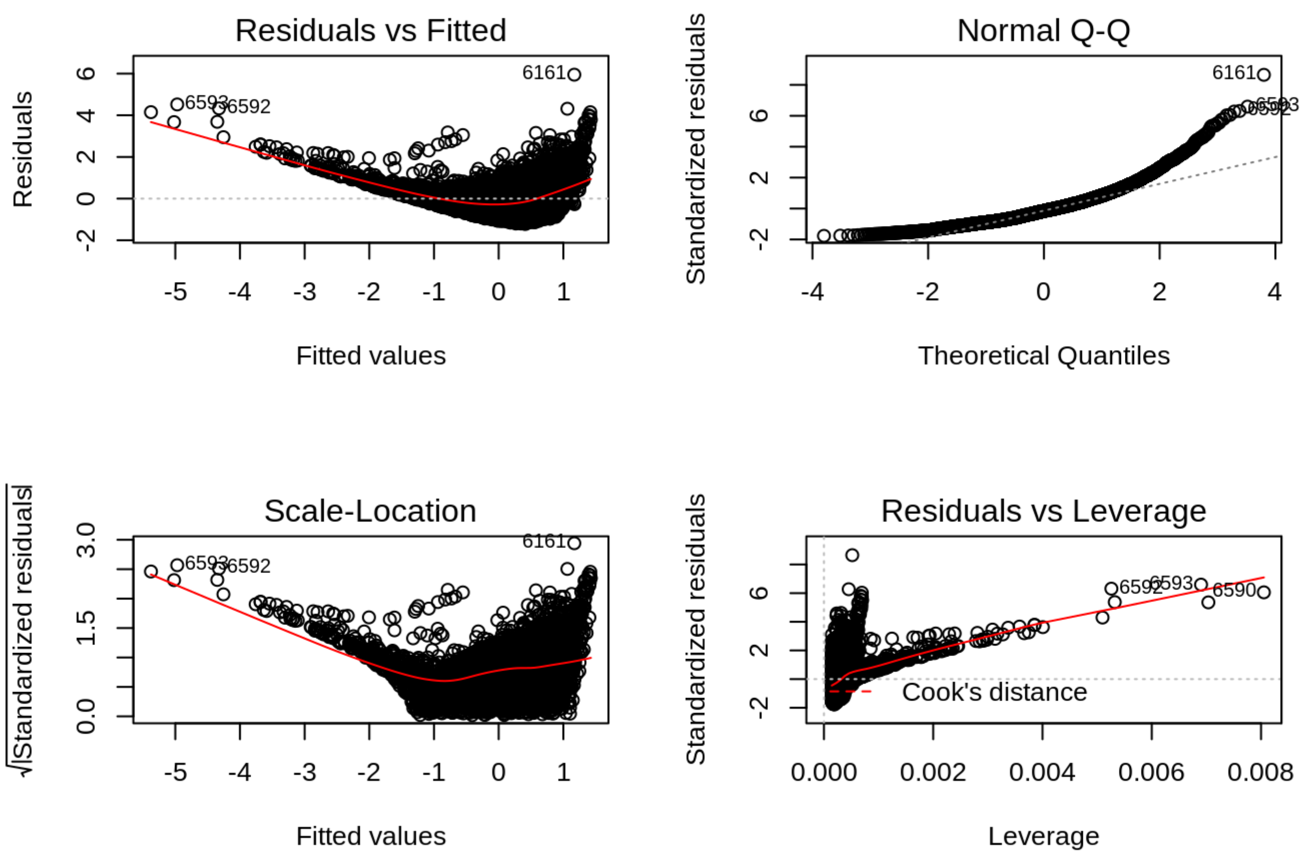
```
par(mfrow=c(2,2))
plot(mod4)
```



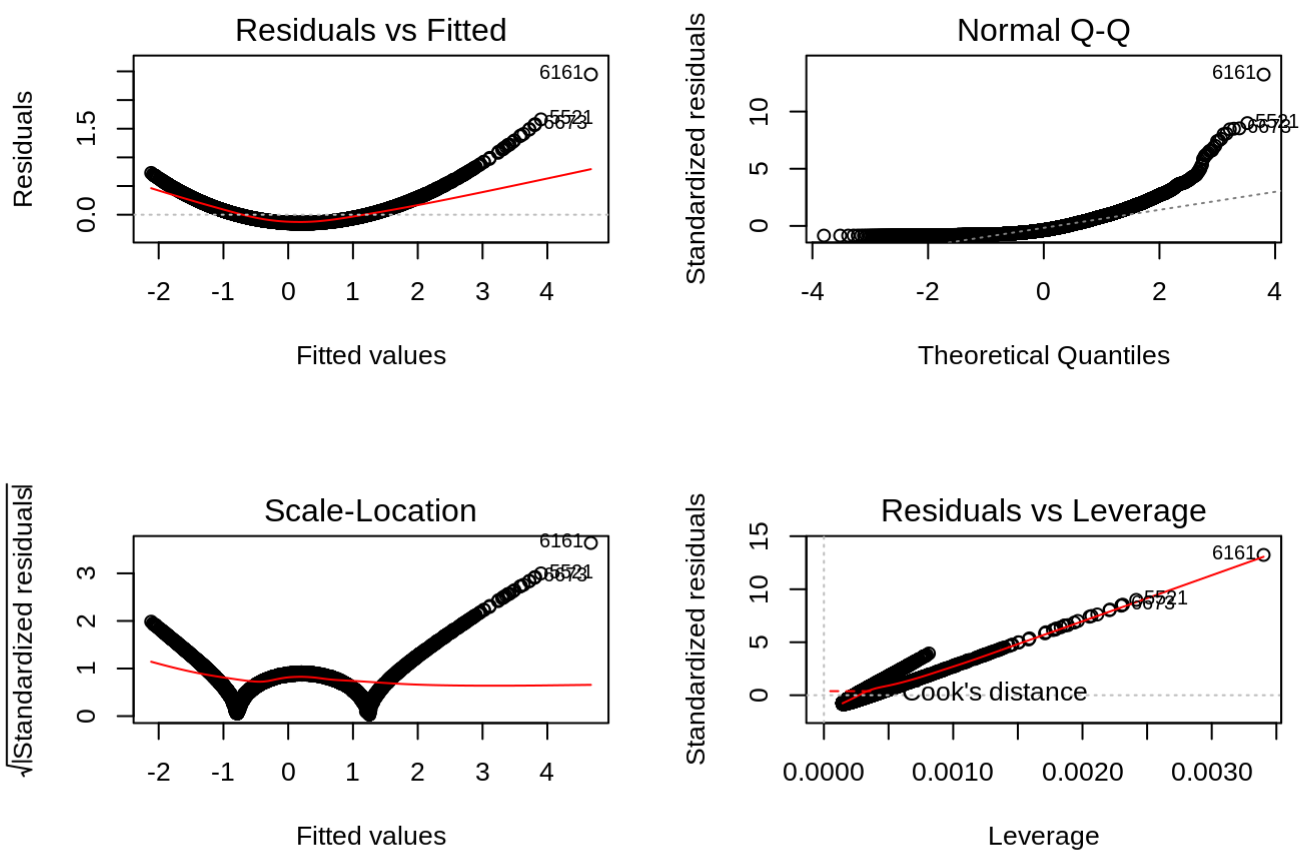
```
par(mfrow=c(2,2))
plot(mod5)
```



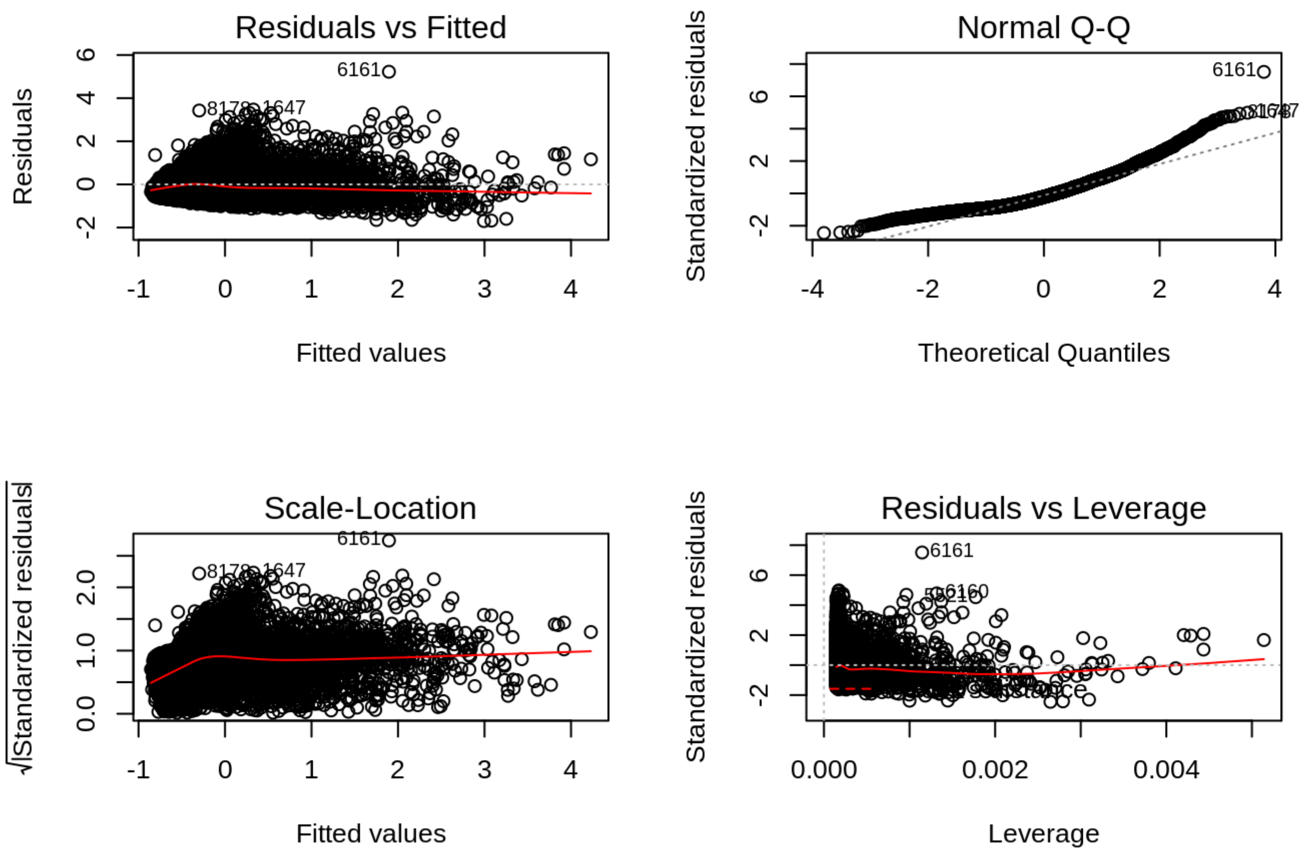
```
par(mfrow=c(2,2))
plot(mod6)
```



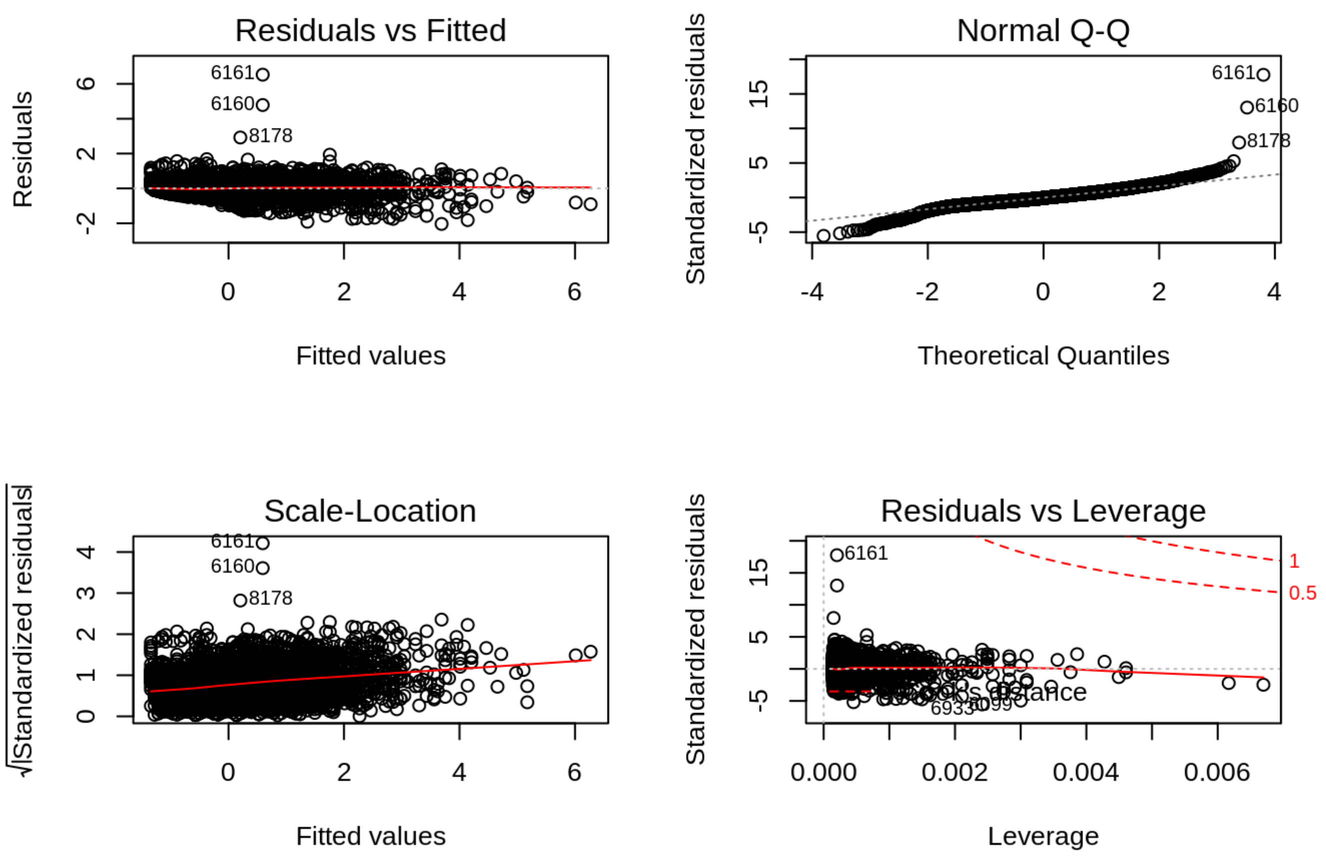
```
par(mfrow=c(2,2))
plot(mod7)
```



```
par(mfrow=c(2,2))
plot(mod8)
```



```
par(mfrow=c(2,2))
plot(mod9)
```



```
#For one of the models create train-test sets, plot the model, for the test set color
real and predicted points differently; R^2 and p-value to title
```

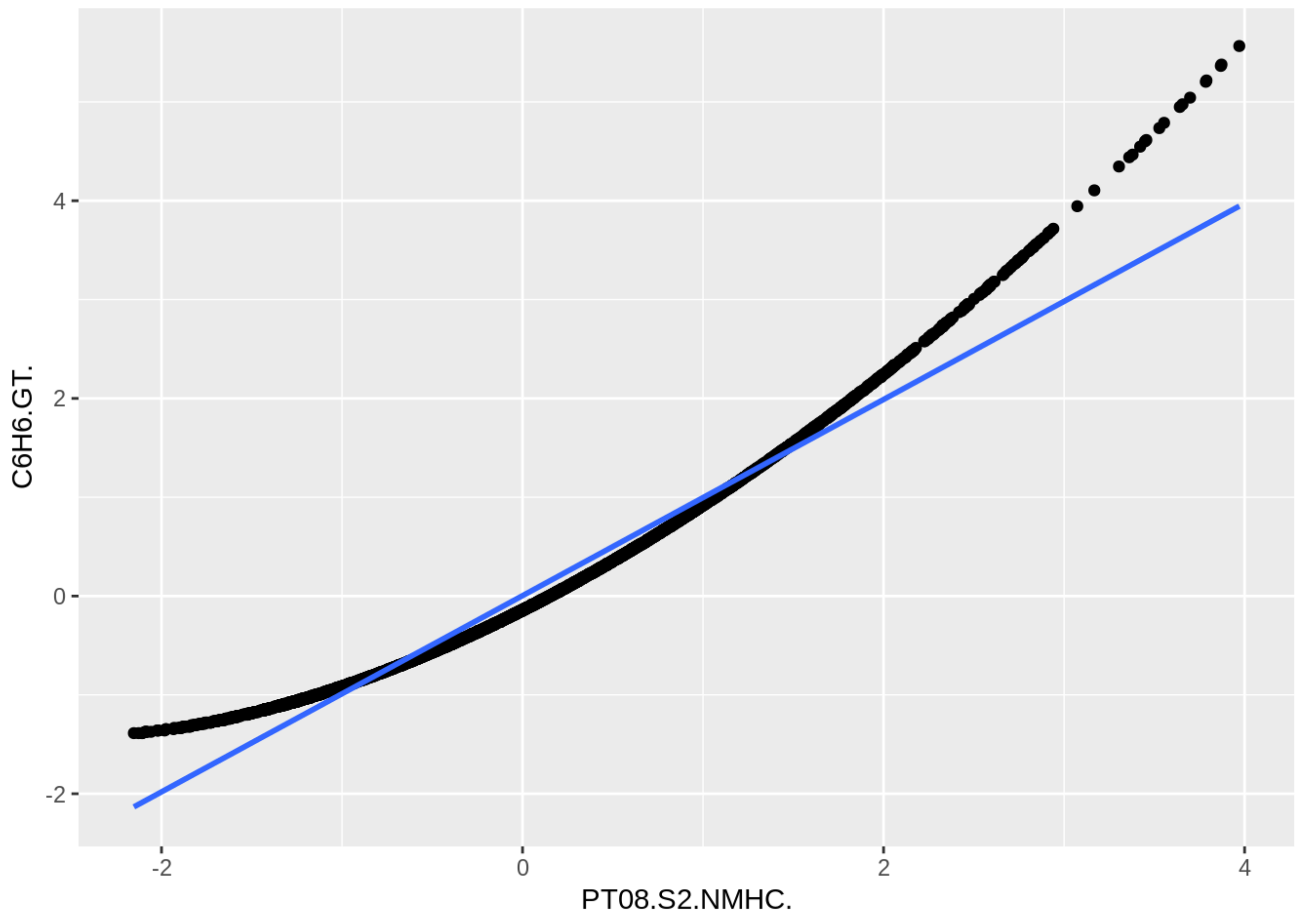
```
################################ TASK 11 ########################################

library(caTools)
# let's use model 7

# splitting the dataset into test and train
split = sample.split(airquality_clean1, SplitRatio = 3/5) # split data
training_set = subset(airquality_clean1, split == TRUE) #assign it to train set
test_set = subset(airquality_clean1, split == FALSE) #the remaining assign to test se
t

# Fitting and predicting on the training set which is 20% of the entire data

new_fit<- lm(C6H6.GT. ~ PT08.S2.NMHC., data=training_set)

summary(new_fit)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ PT08.S2.NMHC., data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.16248 -0.13343 -0.06823  0.07214  1.62009
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.004061   0.002976   1.365    0.172
## PT08.S2.NMHC. 0.992558   0.002915 340.527   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1893 on 4047 degrees of freedom
## Multiple R-squared:  0.9663, Adjusted R-squared:  0.9663
## F-statistic: 1.16e+05 on 1 and 4047 DF,  p-value: < 2.2e-16
```

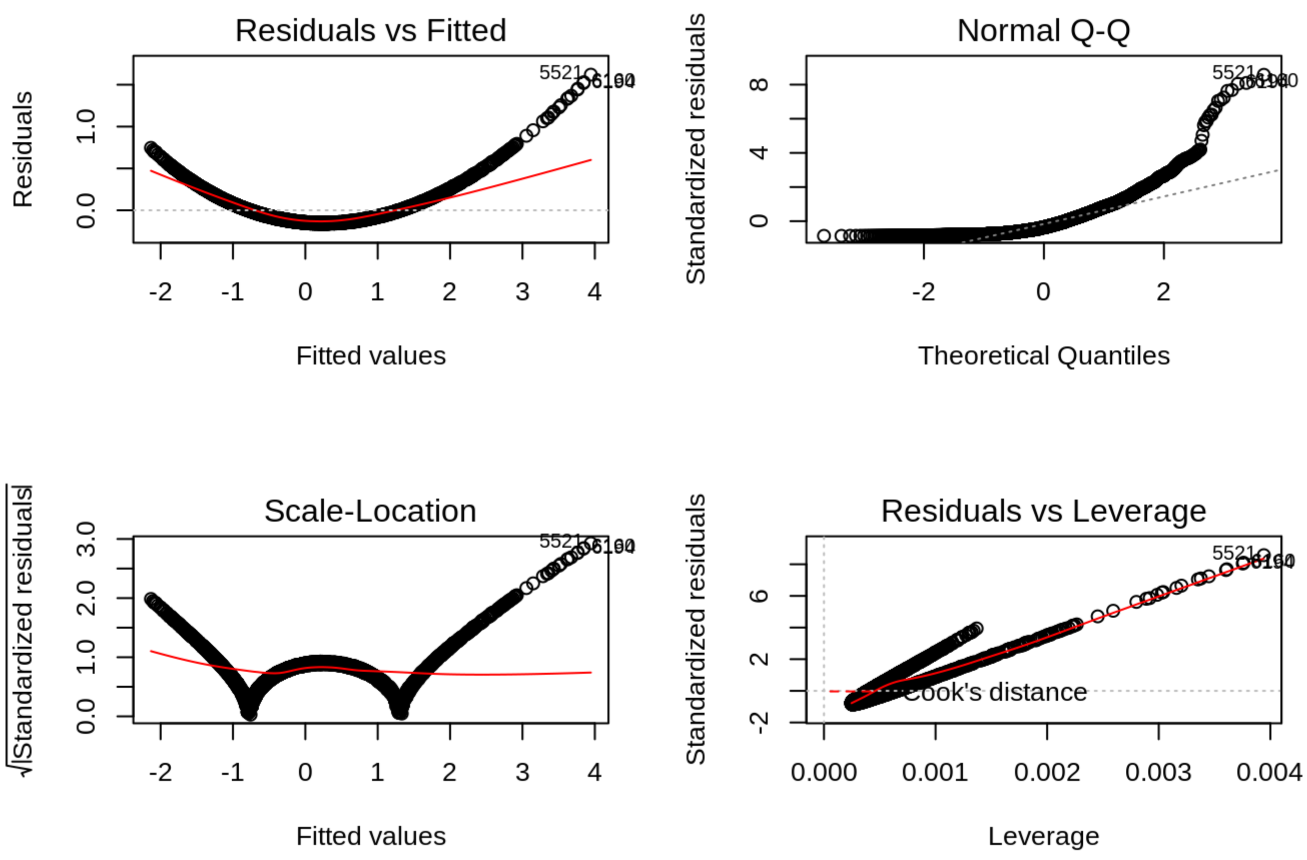
```
# data <- airquality_clean1
# sample <- sample.int(n = nrow(data),size = floor(.20*nrow(data)))
# train <- data[sample, ]
# test <- data[-sample, ]
# new_fit <- lm(C6H6.GT. ~ PT08.S2.NMHC., data=train)
# summary(new_fit)
```

```
# STATISTIC CRITERION
# R-Squared Higher the better (> 0.70)
# Adj R-Squared Higher the better
# F-Statistic   Higher the better
# Std. Error    Closer to zero the better
# t-statistic   Should be greater 1.96 for p-value to be less than 0.05
# AIC   Lower the better
# BIC   Lower the better
# Mallows cp    Should be close to the number of predictors in model
# MAPE (Mean absolute percentage error) Lower the better
# MSE (Mean squared error)  Lower the better
# Min_Max Accuracy => mean(min(actual, predicted)/max(actual, predicted))   Higher th
e better
```

```
ggplot(data = training_set, aes(x =PT08.S2.NMHC. , y =C6H6.GT.))+
  geom_point()+
  geom_smooth(method = "lm")
```



```
# our model explains 96% variation in the data
par(mfrow=c(2,2))
plot(new_fit)
```



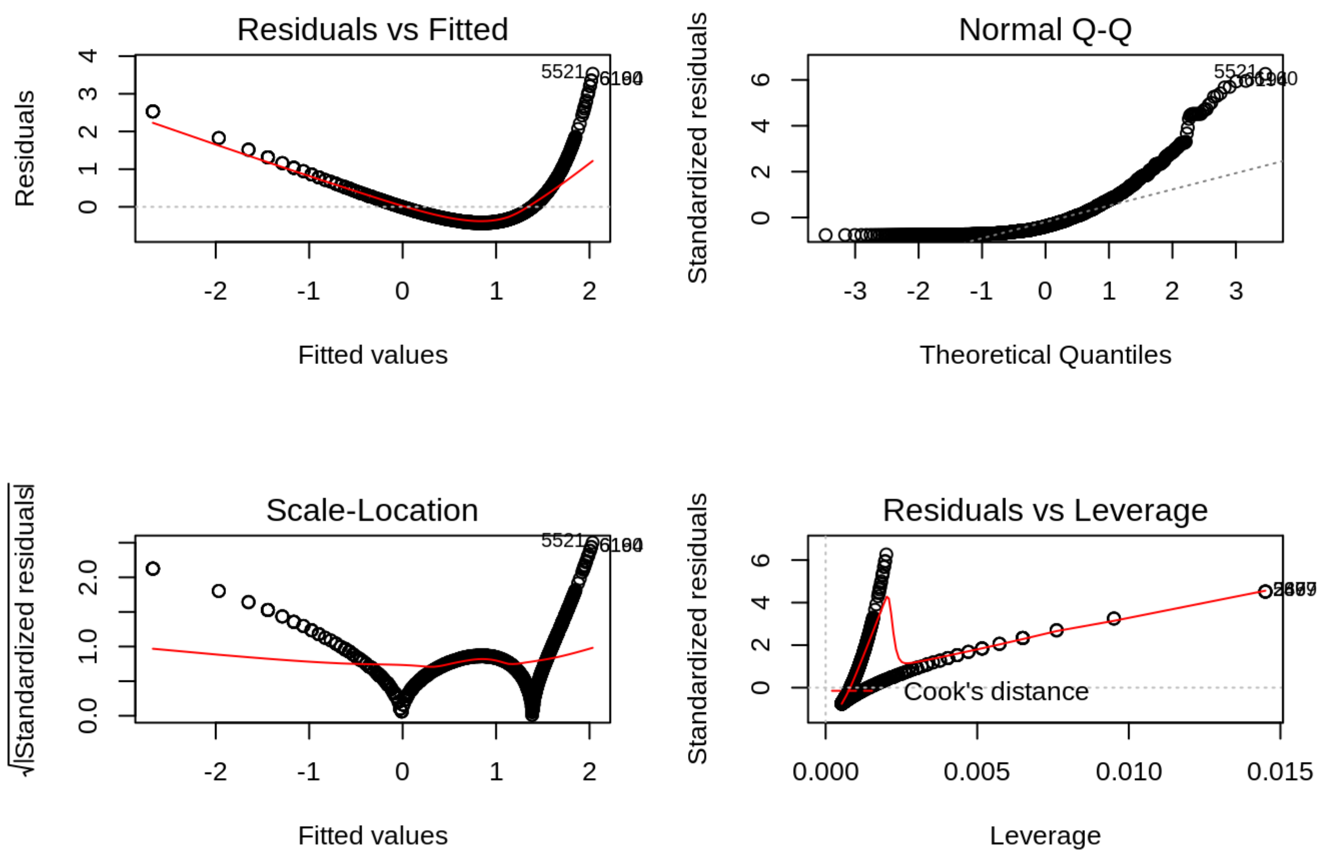
```
#### log transform this model

log_new_fit<- lm(C6H6.GT. ~ log(PT08.S2.NMHC.), data=training_set)
```

```
## Warning in log(PT08.S2.NMHC.): NaNs produced
```

```
summary(log_new_fit)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ log(PT08.S2.NMHC.), data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4299 -0.3792 -0.2152  0.1609  3.5318
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.19477    0.01428   83.68   <2e-16 ***
## log(PT08.S2.NMHC.)  0.60817    0.01142   53.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5646 on 1883 degrees of freedom
##   (2164 observations deleted due to missingness)
## Multiple R-squared:  0.6009, Adjusted R-squared:  0.6007
## F-statistic:  2835 on 1 and 1883 DF,  p-value: < 2.2e-16
```

```
# our model explains 96% variation in the data
par(mfrow=c(2,2))
plot(log_new_fit)
```



```
# log transforming data is not a good idea here
```

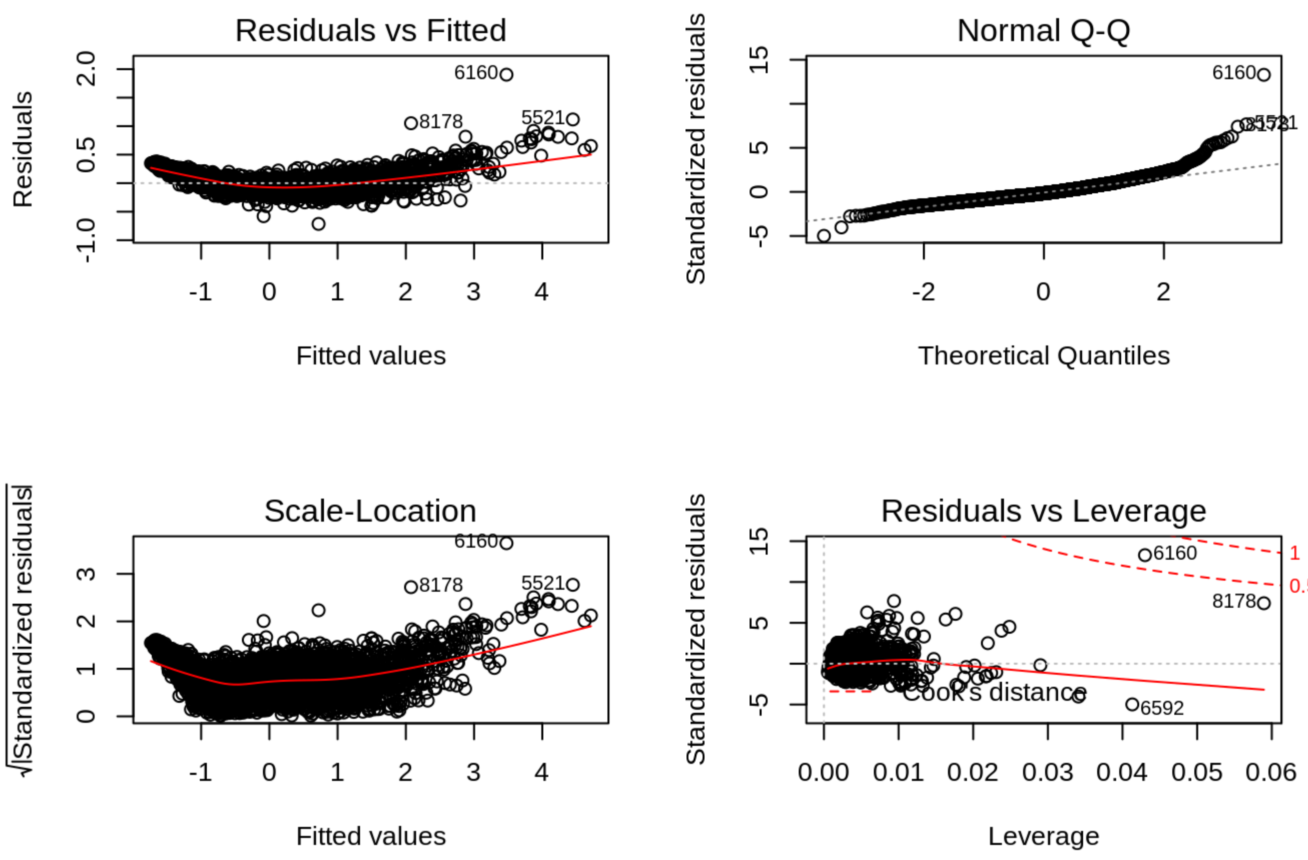
```
# pred <- predict(new_fit , new_data = test)
# head(pred)
#
# test$C6H6.GT._pred <- pred
# head(test)
```

```
## Let's Explore multicollinearity

adv_model <- lm(C6H6.GT. ~. -C6H6.GT., data = training_set)
summary(adv_model)
```

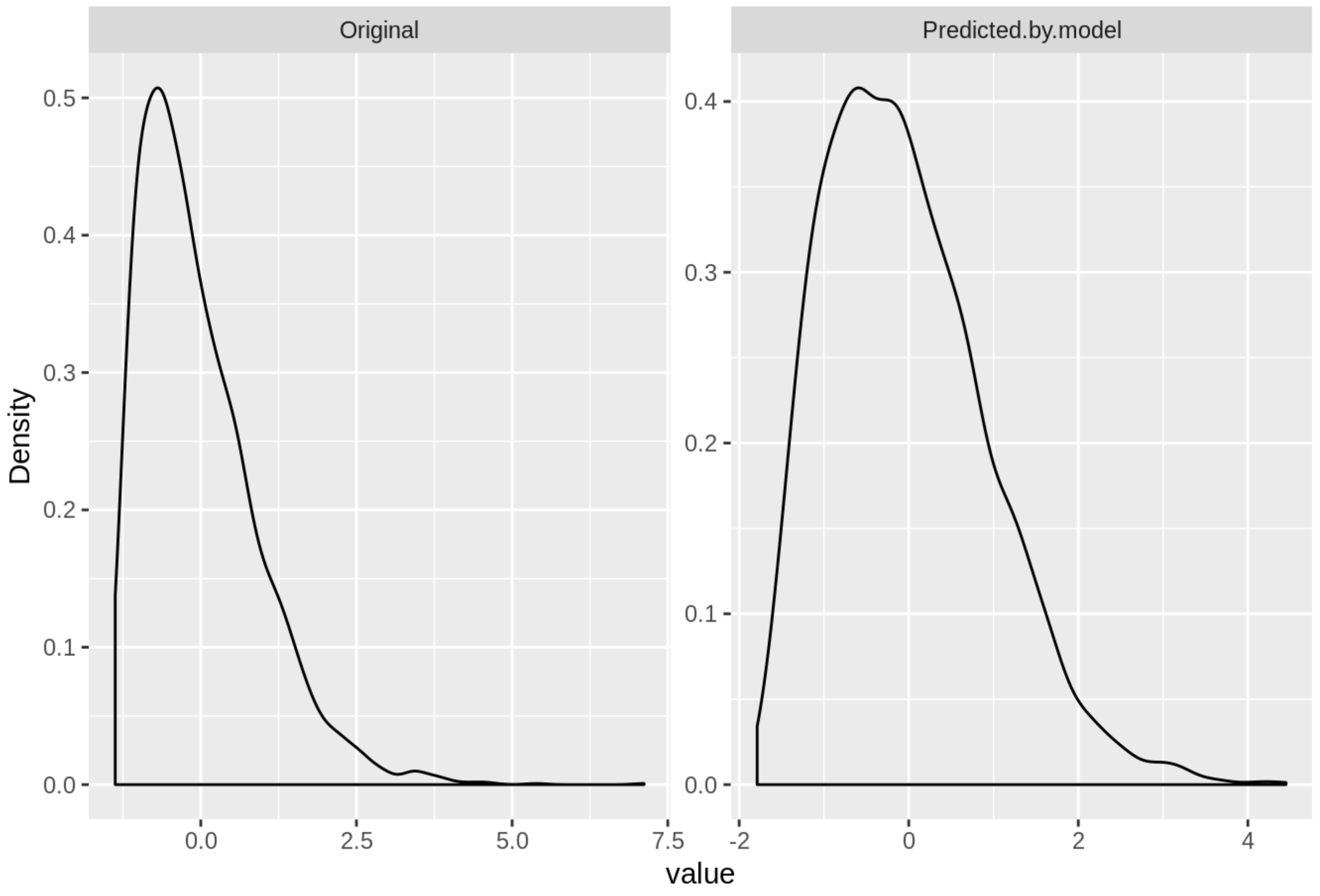
```
##
## Call:
## lm(formula = C6H6.GT. ~ . - C6H6.GT., data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71490 -0.09113 -0.02229  0.07184  1.90262
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.002072   0.002302   0.900  0.36795
## PT08.S1.CO.    0.005177   0.006797   0.762  0.44630
## PT08.S2.NMHC.  0.873368   0.012374  70.582  < 2e-16 ***
## PT08.S3.NOx.   0.086808   0.004958  17.509  < 2e-16 ***
## PT08.S4.NO2.   0.060454   0.009646   6.267 4.06e-10 ***
## PT08.S5.O3.   -0.021615   0.006855  -3.153  0.00163 **
## CO.GT.         0.156573   0.007170  21.836  < 2e-16 ***
## NOx.GT.        0.090760   0.006260  14.498  < 2e-16 ***
## NO2.GT.       -0.072586   0.004765 -15.234  < 2e-16 ***
## T             -0.090379   0.008629 -10.473  < 2e-16 ***
## RH            -0.063404   0.006615  -9.585  < 2e-16 ***
## AH             0.040478   0.007553   5.359 8.84e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1464 on 4037 degrees of freedom
## Multiple R-squared:  0.9799, Adjusted R-squared:  0.9798
## F-statistic: 1.788e+04 on 11 and 4037 DF,  p-value: < 2.2e-16
```

```
# we can see that  *** means 0.001 p values, these are the best predictors
# in our case we don't need to select good predictors as our model now explains 98% v
ariation, it is better than previous one
# But if we have to choose some predictors, we apply a cut off say 0.05 and take all
 the predictors that are less than this cut off, for example here we will select all
 variable with three stars, remaining don't pass this cutoff

# PT08.S2.NMHC.  0.82449    0.02245   36.72  < 2e-16 ***
# PT08.S3.NOx.   0.07743    0.00822    9.43  < 2e-16 ***
# PT08.S4.NO2.   0.09561    0.01729    5.53  4.0e-08 ***
# CO.GT.         0.17500    0.01270   13.78  < 2e-16 ***
# NOx.GT.        0.09844    0.01067    9.23  < 2e-16 ***
# NO2.GT.       -0.07345    0.00835   -8.80  < 2e-16 ***
# T             -0.07541    0.01576   -4.78  1.9e-06 ***
# RH            -0.05864    0.01199   -4.89  1.1e-06 ***
```

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



```
# let's make predictions from our model and compare with original data

predicted_values <- predict(adv_model, test_set)
original_values <- test_set$C6H6.GT.

total<- cbind(predicted_values, original_values)
colnames(total) <- c('Predicted by model', 'Original')

# plot

plot_density(total)
```



```
# we can see that the distribution plots are similar for the model predicted values a
nd original values, our model is thus effective
```

```
# library(corrplot)
# visual_cor <- function(d) {
#   cormat <- cor(d , use = "pairwise.complete.obs")
#   pvalmat <- cor.mtest(d)$p
#
#
#   corrplot(abs(cormat),
#            method = 'color',
#            order = 'hslust',
#            addCoef.col = "black",
#            tl.col = "black", tl.srt = 45,
#            pmat = pvalmat,
#            sig.level = 0.05,
#            insig = "blank",
#            diag = "FALSE")
# }
# visual_cor(training_set[,-c(1:4)])
```

```
#find correlation between all variables and recognize highly corelated one
round(cor(training_set), 2)
```

```
##                C6H6.GT. PT08.S1.CO. PT08.S2.NMHC. PT08.S3.NOx. PT08.S4.NO2.
## C6H6.GT.           1.00        0.88          0.98        -0.73         0.77
## PT08.S1.CO.        0.88        1.00          0.89        -0.77         0.69
## PT08.S2.NMHC.      0.98        0.89          1.00        -0.79         0.78
## PT08.S3.NOx.      -0.73       -0.77         -0.79         1.00        -0.52
## PT08.S4.NO2.       0.77        0.69          0.78        -0.52         1.00
## PT08.S5.O3.        0.87        0.90          0.88        -0.80         0.59
## CO.GT.             0.93        0.88          0.92        -0.70         0.64
## NOx.GT.            0.72        0.71          0.71        -0.67         0.25
## NO2.GT.            0.60        0.63          0.63        -0.65         0.15
## T                  0.19        0.03          0.22        -0.09         0.56
## RH                -0.01        0.19         -0.03        -0.13         0.00
## AH                 0.20        0.17          0.22        -0.23         0.65
##                PT08.S5.O3. CO.GT. NOx.GT. NO2.GT.     T    RH    AH
## C6H6.GT.              0.87   0.93    0.72    0.60  0.19 -0.01  0.20
## PT08.S1.CO.           0.90   0.88    0.71    0.63  0.03  0.19  0.17
## PT08.S2.NMHC.         0.88   0.92    0.71    0.63  0.22 -0.03  0.22
## PT08.S3.NOx.         -0.80  -0.70   -0.67   -0.65 -0.09 -0.13 -0.23
## PT08.S4.NO2.          0.59   0.64    0.25    0.15  0.56  0.00  0.65
## PT08.S5.O3.           1.00   0.86    0.79    0.70 -0.04  0.18  0.09
## CO.GT.                0.86   1.00    0.79    0.67  0.02  0.08  0.08
## NOx.GT.               0.79   0.79    1.00    0.76 -0.28  0.24 -0.13
## NO2.GT.               0.70   0.67    0.76    1.00 -0.22 -0.06 -0.34
## T                    -0.04   0.02   -0.28   -0.22  1.00 -0.57  0.66
## RH                    0.18   0.08    0.24   -0.06 -0.57  1.00  0.18
## AH                    0.09   0.08   -0.13   -0.34  0.66  0.18  1.00
```

```
my_model <- lm(formula = C6H6.GT. ~ . , data = training_set)
summary(my_model)
```

```
##
## Call:
## lm(formula = C6H6.GT. ~ ., data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71490 -0.09113 -0.02229  0.07184  1.90262
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.002072   0.002302   0.900  0.36795
## PT08.S1.CO.    0.005177   0.006797   0.762  0.44630
## PT08.S2.NMHC.  0.873368   0.012374  70.582  < 2e-16 ***
## PT08.S3.NOx.   0.086808   0.004958  17.509  < 2e-16 ***
## PT08.S4.NO2.   0.060454   0.009646   6.267 4.06e-10 ***
## PT08.S5.O3.   -0.021615   0.006855  -3.153  0.00163 **
## CO.GT.         0.156573   0.007170  21.836  < 2e-16 ***
## NOx.GT.        0.090760   0.006260  14.498  < 2e-16 ***
## NO2.GT.       -0.072586   0.004765 -15.234  < 2e-16 ***
## T             -0.090379   0.008629 -10.473  < 2e-16 ***
## RH            -0.063404   0.006615  -9.585  < 2e-16 ***
## AH             0.040478   0.007553   5.359 8.84e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1464 on 4037 degrees of freedom
## Multiple R-squared:  0.9799, Adjusted R-squared:  0.9798
## F-statistic: 1.788e+04 on 11 and 4037 DF,  p-value: < 2.2e-16
```

```
#as we can see from the summary of the model, PT08.S2.NMHC., PT08.S3.NOx., PT08.S4.NO
2., CO.GT. , NOx.GT., NO2.GT.,T, RH, AH are independent variables that are highly sta
tically significant!also p value is very small that shows the model is very significa
nt!

#now we should check whether we have this multicolinearity or not! for that we can us
e function "vif" that means variance inflation factors for each of the independent va
riables.
 #If we have values bigger than 5 then we can understand that independent variables a
lso have correlation with each other and it will effect on the model! we should drop
 that variables and then run the model again without that variables!
#all the remaining values have vif value less than 5! Therefore,optimum model will ha
ve following explanatory variables:
#

#step by step variable selection leads to elimination of multi colinearity and fittin
g the model with the optimum number of explanaory variavles.
```

```
library(car)
```

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

```
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
vif(my_model)
```

```
##   PT08.S1.CO. PT08.S2.NMHC.  PT08.S3.NOx.  PT08.S4.NO2.   PT08.S5.O3.
##      9.041716     30.149810      4.805499     18.197733      9.184653
##        CO.GT.       NOx.GT.       NO2.GT.             T            RH
##     10.268085      7.705126      4.381729     14.081001      8.336310
##            AH
##     10.817006
```

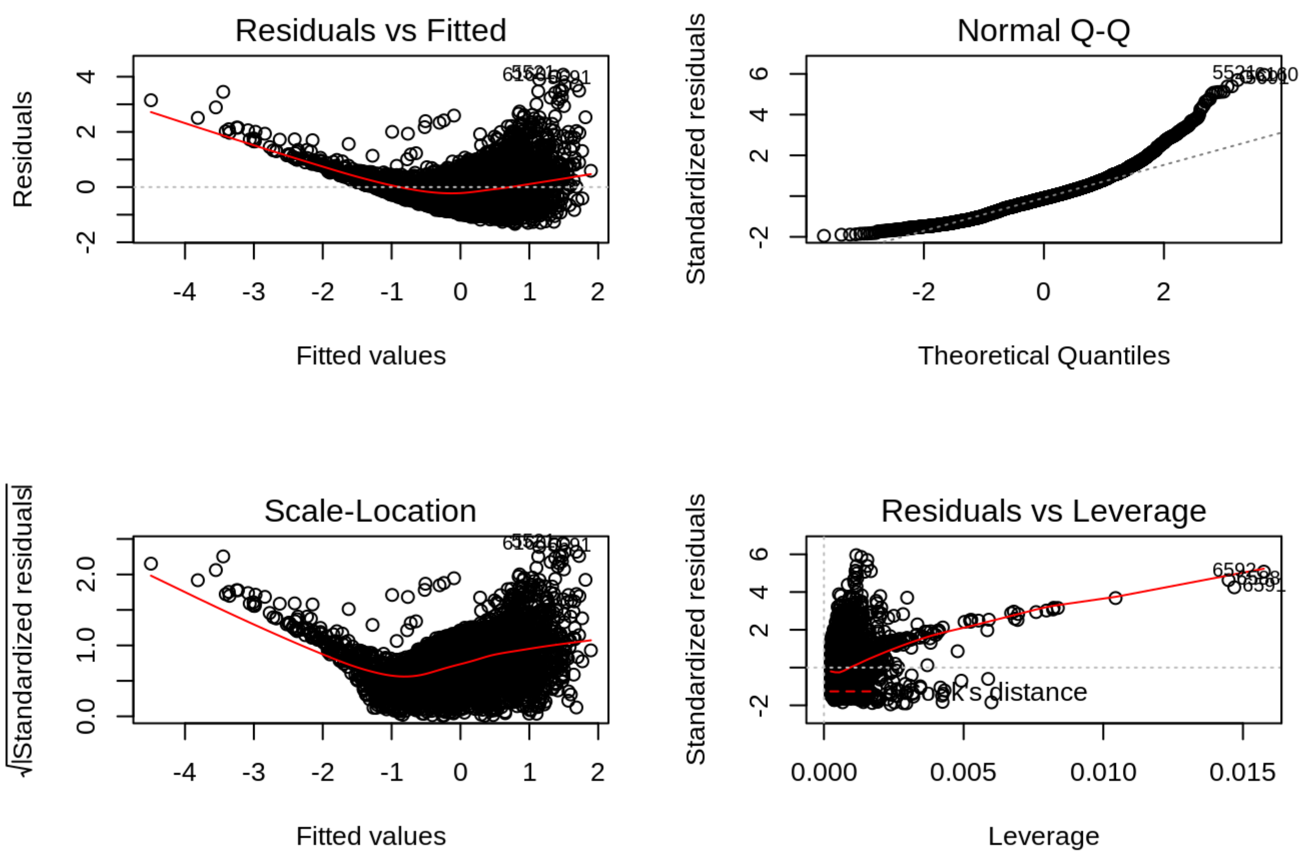
2

```
# so run the model again without variables with vif greater than 5:
my_model2 <- lm(formula = C6H6.GT. ~ PT08.S3.NOx. + NO2.GT. , data = training_set)
```

```
#Now run the vif function for new model and check the results.
vif(my_model2)
```

```
## PT08.S3.NOx.      NO2.GT.
##     1.712682     1.712682
```

```
#as we cav see the resulting explanatory variables vif do not have multicolinearity p
roblem!if we see again we should drop the values till we do not have any vif value gr
eater than 5! then make model again!

par(mfrow=c(2,2))
plot(my_model2)
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q-Q
Standardized residuals
Theoretical Quantiles
Scale-Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance

```
#######################usefulexplanation########################
#before creating model do some exploratory data analysis!
#1)looking at data
#2)visualization
#3)summary statistics
#the role of modeling is to explore the relationship between variables. using scater
 plot you can show this relationship.
#Build simple linear models with each predictor.Linear regression is used to predict
 the value of an outcome variable Y based on one or more input predictor variables X.
The aim is to establish a linear relationship (a mathematical formula) between the pr
edictor variable(s) and the response variable, so that, we can use this formula to es
timate the value of the response Y, when only the predictors (Xs) values are known.

#he aim of linear regression is to model a continuous variable Y as a mathematical fu
nction of one or more X variable(s), so that we can use this regression model to pred
ict the Y when only the X is known.

#Y=β1+β2X+ϵ
#where, β1 is the intercept and β2 is the slope. Collectively, they are called regres
sion #coefficients. ϵ is the error term, the part of Y the regression model is unable
to explain.
#consists of 1231 observations(rows) and 15 variables (columns)

#Before we begin building the regression model, it is a good practice to analyze and
 understand the variables. The graphical analysis and correlation study below will he
lp with this.

#the aim of this exercise is to build a simple regression model that we can use to pr
edict NMHC.GT. by establishing a statistically significant linear relationship with a
ll other variables. But before jumping in to the syntax, lets try to understand these
variables graphically. Typically, for each of the independent variables (predictors),
the following plots are drawn to visualize the following behavior:

#It adds a small amount of random variation to the location of each point, and is a u
seful way of handling overplotting caused by discreteness in smaller datasets.
#we can understand the correlation between variables(possitive or negative) using cor
relation coefficiant(summary statistic between negative one and one measuring strengt
h linear association of 2 numerical variables(for example in line that as well as x i
ncrease y will decrease we can tell that it is negatively line slope!) when x and y b
ehave independently we can say that there is 0 relationship)
#cor() function takes 2 numerical variable and gives correlation (can not be greater
 than 1):data %>% summarize(correlation = cor(x,y)) and when the value is negative we
can say that it is negative relationship(weakly negative). correlation near the 1 wil
l shows possitivle strong relationship
#prediction with modeling: if i give input x to to f() can i get prediction of y(hat)
that is close to the true value of y!
#both variables are numeric.the name of output will be response variable(and it is th
e quantity that we believe might be related to the input!)
#explanatory variable:something that you think might be related to the response
#x is independent(predictor)
#y is dependent (response)
#graghical representation: response=y axis, explanatory/predictor=x axis
#The relationship between two variables may not be linear. In these cases we can some
times see strange and even inscrutable patterns in a scatterplot of the data. Sometim
es there really is no meaningful relationship between the two variables. Other times,
a careful transformation of one or both of the variables can reveal a clear relations
hip.
```

```
#we can understand that how outliers can affect the results of a linear regression mo
del and how we can deal with them. For now, it is enough to simply identify them and
 note how the relationship between two variables may change as a result of removing o
utliers.
#we can summarize relationship between 2 variable with fitting line(linear regression
line that separating signals from the noise) by adding geom_smooth() with method= lm
 for linear model and se = FALSE to meetstandard error bars to the ggplot.
#to show the relation between x and y we need to draw a line! and understand which li
ne will fit better!we need a numerical measurment of how good fit of each possible li
ne is? in regression we use "least square criterian" to find the best fit line! line
 that minimize the sum square of distances  between the line and a set of data point
s! a unique line exist! that line is called "least square regression line". we can ad
d the line to the plot using the geom_smooth function!
#regression model combine some explanatory variables to estimate single numerical res
ponse variable!
#The function used for building linear models is lm(). The lm() function takes in two
main arguments, namely: 1. Formula 2. Data.
#making model for prediction! like making model for prediction of price of apartment
 based on some features like size, condition, number of floors and so on!
#slop:شیب خط رگرژن
#response = f(explanatory) + noise
#regression = intercept + f(slop*explanatory)+noise <- y = B0+B1.x + e
#y' = B0' +B1'.X    y = actual observed values of the response, y' = expected values
 of the response based on the model
#reduals : difference between y and y'     e' = y-y'
#e is unlnown quantity(noise) but e' is known estimate of e
#residuals: is the difference between the observed y value and predicted or fitted y
 value (y')
#standard deviation of this errors or residuals called residuals standard error in R
 and is presented in the model summary. residuals or error terms are useful for chack
ing the model assumptions
#assumptions:
  #y-values or the errors are independent
  #y-values can be expressed as a linear function of the x-values
#variation of observations around the regression line (residual SE) is constant!
#for given value of x, y values are normaly distribured
```

```
#Null hypothesis is a general statement that there is no relationship between two mea
sured phenomena.
```