Experiment No 1

Title : Regression Analysis and Plot Interpretations in R

Theory

Regression

Regression is a parametric technique used to predict continuous (dependent) variable given a set of independent variables. It is parametric in nature because it makes certain assumptions (discussed next) based on the data set. If the data set follows those assumptions, regression gives incredible results. Otherwise, it struggles to provide convincing accuracy.

Mathematically, regression uses a linear function to approximate (predict) the dependent variable given as:

$Y = \beta o + \beta 1X + \in$

where, Y - Dependent variable ,X - Independent variable, βo - Intercept
β1 – Slope, ∈ - Error

βo and β1 are known as coefficients. This is the equation of simple linear regression. It's called 'linear' because there is just one independent variable (X) involved. In multiple regression, we have many independent variables (Xs). If you recall, the equation above is nothing but a line equation (y = mx + c) we studied in schools. Let's understand what these parameters say:

Y - This is the variable we predict
X - This is the variable we use to make a prediction
βo - This is the intercept term. It is the prediction value you get when X = 0
β1 - This is the slope term. It explains the change in Y when X changes by 1 unit. ∈ - This represents the residual value, i.e. the difference between actual and predicted values.

Error is an inevitable part of the prediction-making process. No matter how powerful the algorithm we choose, there will always remain an (∈) irreducible error which reminds us that the "future is uncertain."

Yet, we humans have a unique ability to persevere, i.e. we know we can't completely eliminate the (∈) error term, but we can still try to reduce it to the lowest. Right? To do this, regression uses a technique known as **Ordinary Least Square** (OLS).

So the next time when you say, I am using *linear /multiple regression*, you are actually referring to the *OLS technique*. Conceptually, OLS technique tries to reduce the sum of squared errors $\sum$[Actual(y) - Predicted(y')]^2 by finding the best possible value of regression coefficients (β0, β1, etc).

Is OLS the only technique regression can use? No! There are other techniques such as Generalized Least Square, Percentage Least Square, Total Least Squares, Least absolute deviation, and many more. Then, why OLS? Let's see.

1. It uses squared error which has nice mathematical properties, thereby making it easier to differentiate and compute gradient descent.
2. OLS is easy to analyze and computationally faster, i.e. it can be quickly applied to data sets having 1000s of features.
3. Interpretation of OLS is much easier than other regression techniques.

Assumptions made in regression

Regression is a parametric technique, so it makes assumptions. Let's look at the assumptions it makes:

1. There exists a **linear** and **additive** relationship between dependent (DV) and independent variables (IV). By linear, it means that the change in DV by 1 unit change in IV is constant. By additive, it refers to the effect of X on Y is independent of other variables.

2. There must be no correlation among independent variables. Presence of correlation in independent variables lead to **Multicollinearity**. If variables are correlated, it becomes extremely difficult for the model to determine the true effect of IVs on DV.
3. The error terms must possess constant variance. Absence of constant variance leads to **heteroskedestacity**.
4. The error terms must be uncorrelated i.e. error at $\in t$ must not indicate the at error at $\in t+1$. Presence of correlation in error terms is known as **Autocorrelation**. It drastically affects the regression coefficients and standard error values since they are based on the assumption of uncorrelated error terms.
5. The dependent variable and the error terms must possess a **normal distribution**.

Presence of these assumptions make regression quite restrictive. By **restrictive** I meant, the performance of a regression model is conditioned on fulfillment of these assumptions.

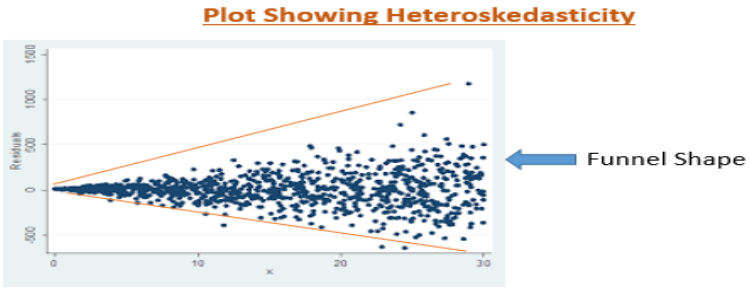
How do I know these assumptions are violated in my data?

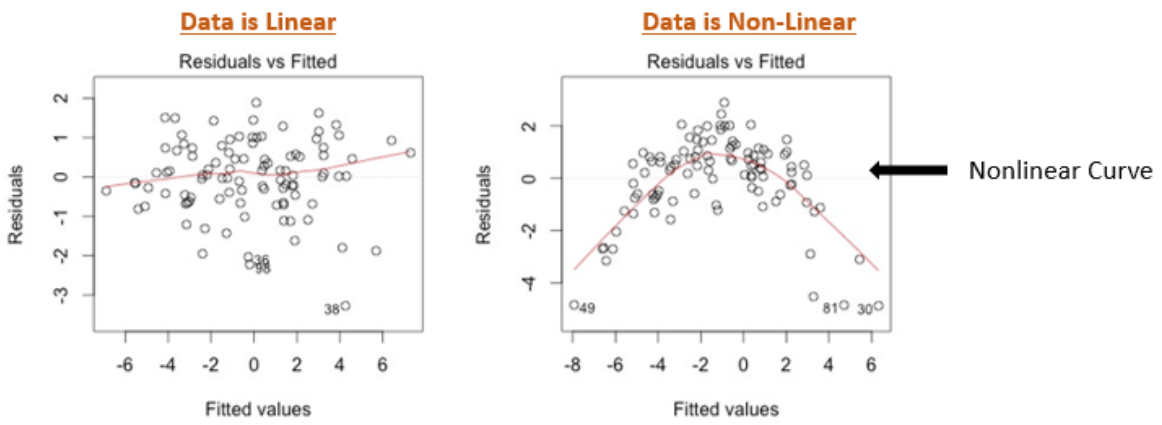
Once these assumptions get violated, regression makes biased, erratic predictions.

you can check performance metrics to estimate violation. But the real treasure is present in the diagnostic a.k.a residual plots. Let's look at the important ones:

1. **Residual vs. Fitted Values Plot**

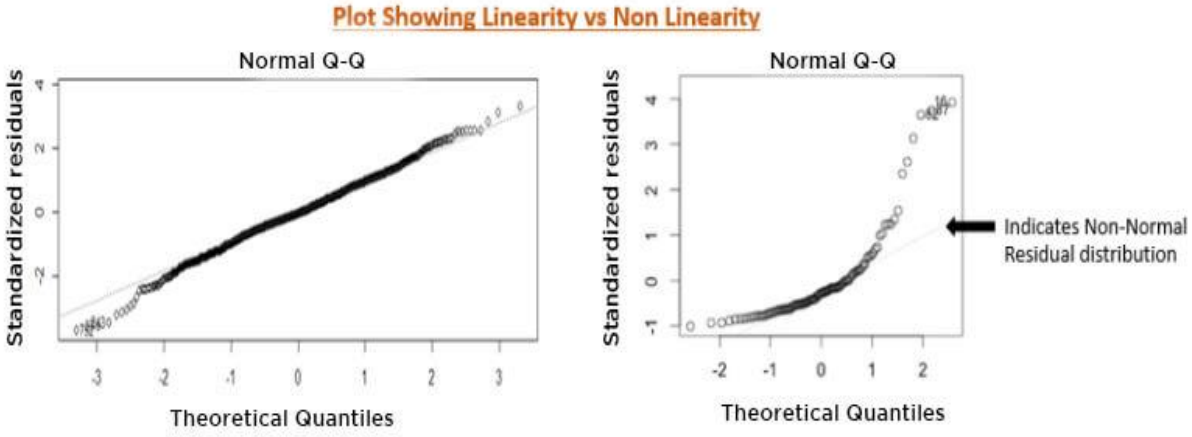
Ideally, this plot shouldn't show any pattern. But if you see any shape (curve, U shape), it suggests non-linearity in the data set. In addition, if you see a funnel shape pattern, it suggests your data is suffering from heteroskedasticity, i.e. the error terms have non-constant variance.





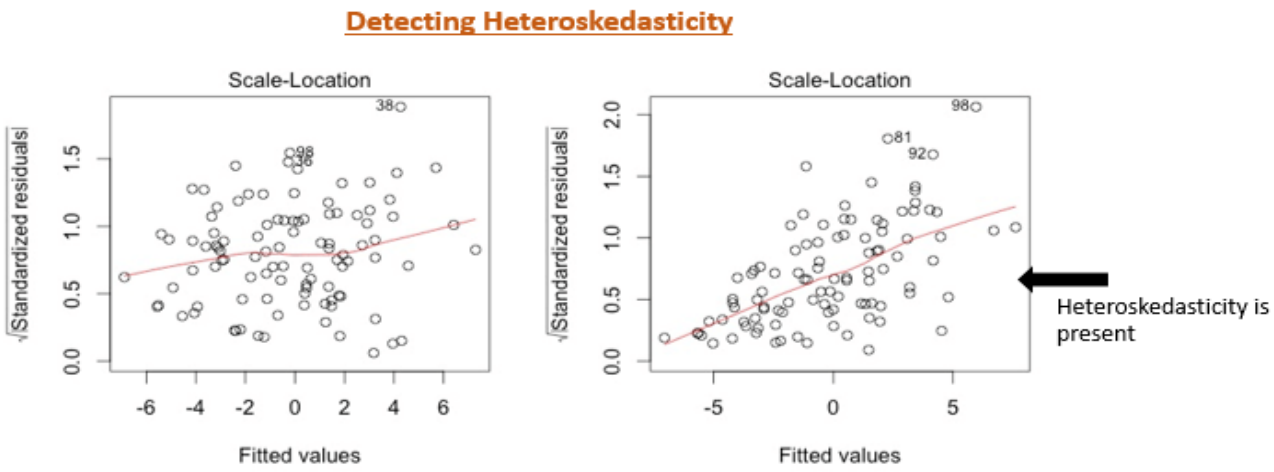
2. **Normality Q-Q Plot**

As the name suggests, this plot is used to determine the normal distribution of errors. It uses standardized values of residuals. Ideally, this plot should show a straight line. If you find a curved, distorted line, then your residuals have a non-normal distribution (problematic situation).



3. **Scale Location Plot**

This plot is also useful to determine heteroskedasticity. Ideally, this plot shouldn't show any pattern. Presence of a pattern determine heteroskedasticity. Don't forget to corroborate the findings of this plot with the funnel shape in residual vs. fitted values.



If you are a non-graphical person, you can also perform quick tests / methods to check assumption violations:

1. **Durbin Watson Statistic (DW)** - This test is used to check autocorrelation. Its value lies between 0 and 4. A DW=2 value shows no autocorrelation. However, a value between 0 < DW < 2 implies positive autocorrelation, while 2 < DW < 4 implies negative autocorrelation.
2. **Variance Inflation Factor** (VIF) - This metric is used to check multicollinearity. VIF <=4 implies no multicollinearity but VIF >=10 suggests high multicollinearity. Alternatively, you can also look at the tolerance (1/VIF) value to determine correlation in IVs. In addition, you can also create a correlation matrix to determine collinear variables.
3. **Breusch-Pagan / Cook Weisberg Test** - This test is used to determine presence of heteroskedasticity. If you find p < 0.05, you reject the null hypothesis and infer that heteroskedasticity is present.

How can improve the accuracy of a regression model ?

There is little you can do when your data violates regression assumptions. An obvious solution is to use tree-based algorithms which capture non-linearity quite well. The following are some tips to implement:

1. If your data is suffering from non-linearity, **transform the IVs** using sqrt, log, square, etc.
2. If your data is suffering from heteroskedasticity, **transform the DV** using sqrt, log, square, etc. Also, you can use weighted least square method to tackle this problem.

3. If your data is suffering from multicollinearity, use a correlation matrix to check correlated variables. Let's say variables A and B are highly correlated. Now, instead of removing one of them, use this approach: Find the **average correlation** of A and B with the rest of the variables. Whichever variable has the higher average in comparison with other variables, remove it. Alternatively, you can use **penalized regression methods** such as lasso, ridge, elastic net, etc.
4. You can do variable selection based on **p values**. If a variable shows p value > 0.05, we can remove that variable from model since at p> 0.05, we'll always fail to reject null hypothesis.

How can access the fit of regression model?

The ability to determine model fit is a tricky process. The metrics used to determine model fit can have different values based on the type of data. Hence, we need to be extremely careful while interpreting regression analysis. Following are some metrics you can use to evaluate your regression model:

1. **R Square (Coefficient of Determination)** - As explained above, this metric explains the percentage of variance explained by covariates in the model. It ranges between 0 and 1. Usually, higher values are desirable but it rests on the data quality and domain. For example, if the data is noisy, you'd be happy to accept a model at low R^2 values. But it's a good practice to consider adjusted R^2 than R^2 to determine model fit.
2. **Adjusted R^2**- The problem with R^2 is that it keeps on increasing as you increase the number of variables, regardless of the fact that the new variable is actually adding new information to the model. To overcome that, we use adjusted R^2 which doesn't increase (stays same or decrease) unless the newly added variable is truly useful.
3. **F Statistics** - It evaluates the overall significance of the model. It is the ratio of explained variance by the model by unexplained variance. It compares the full model with an intercept only (no predictors) model. Its value can range between zero and any arbitrary large number. Naturally, higher the F statistics, better the model.
4. **RMSE / MSE / MAE** - Error metric is the crucial evaluation number we must check. Since all these are errors, lower the number, better the model. Let's look at them one by one:

- **MSE** - This is mean squared error. It tends to amplify the impact of outliers on the model's accuracy. For example, suppose the actual y is 10 and predictive y is 30, the resultant MSE would be $(30\text{-}10)^2 = 400$.
- **MAE** - This is mean absolute error. It is robust against the effect of outliers. Using the previous example, the resultant MAE would be (30-10) = 20
- **RMSE** - This is root mean square error. It is interpreted as how far on an average, the residuals are from zero. It nullifies squared effect of MSE by square root and provides the result in original units as data. Here, the resultant RMSE would be $\sqrt{(30\text{-}10)^2} = 20$. Don't get baffled when you see the same value of MAE and RMSE. Usually, we calculate these numbers after summing overall values (actual - predicted) from the data.

R Program

```
path ="d:/vsm"

setwd(path) #set working directory

mydata <- read.csv("airfoil_self_noise.csv") #load data and check data

str(mydata)

colSums(is.na(mydata)) #check missing values

cor(mydata)

regmodel <- lm(Sound_pressure_level ~ ., data = mydata)

summary(regmodel)

par(mfrow=c(2,2)) #set graphic output

plot (regmodel) #create residual plots

set.seed(1) #sample

d <- sample ( x = nrow(mydata), size = nrow(mydata)*0.7)

train <- mydata[d,] #1052 rows

test <- mydata[-d,] #451 rows

regmodel <- lm (Sound_pressure_level ~.,data = train) #train model

summary(regmodel)

regpred <- predict(regmodel, test) #test model

library(Metrics)
```

rmse(actual = test$Sound_pressure_level,predicted = regpred)

d <- boxplot(train$Displacement,varwidth = T,outline = T,border = T,plot = T)

d$out #enlist outlier observations

Output of the program and analysis

```
> path ="d:/vsm"
> setwd(path)
> mydata <- read.csv("airfoil_self_noise.csv")
> str(mydata)
'data.frame':  1503 obs. of  6 variables:
 $ Frquency.Hz.        : int  800 1000 1250 1600 2000 2500 3150 4000 5000 6300 ...
 $ Angle_of_Attack     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Chord_Length        : num  0.305 0.305 0.305 0.305 0.305 ...
 $ Free_stream_velocity: num  71.3 71.3 71.3 71.3 71.3 71.3 71.3 71.3 71.3 71.3 ..
.
 $ Displacement        : num  0.00266 0.00266 0.00266 0.00266 0.00266 ...
 $ Sound_pressure_level: num  126 125 126 128 127 ...


> colSums(is.na(mydata))
        Frquency.Hz.      Angle_of_Attack         Chord_Length Free_stream_velocit
y
                   0                    0                    0                    0
        Displacement Sound_pressure_level
                   0                    0
```

This data has no missing values

```
> cor(mydata)
                     Frquency.Hz. Angle_of_Attack Chord_Length Free_stream_velocit
y
Frquency.Hz.          1.000000000     -0.27276454 -0.003660639          0.13366383
1
Angle_of_Attack      -0.272764536      1.00000000 -0.504868150          0.05875956
5
Chord_Length         -0.003660639     -0.50486815  1.000000000          0.00378662
9
Free_stream_velocity  0.133663831      0.05875957  0.003786629          1.00000000
0
Displacement         -0.230107353      0.75339378 -0.220842431         -0.00397401
3
```

```
Sound_pressure_level -0.390711412     -0.15610753 -0.236161512          0.12510280
1
                     Displacement Sound_pressure_level
Frquency.Hz.         -0.230107353           -0.3907114
Angle_of_Attack       0.753393785           -0.1561075
Chord_Length         -0.220842431           -0.2361615
Free_stream_velocity -0.003974013            0.1251028
Displacement          1.000000000           -0.3126695
Sound_pressure_level -0.312669506            1.0000000
```

Usually, correlation above 80% (subjective) is considered higher. Therefore, we ca n forego this combination and won't remove any variable.

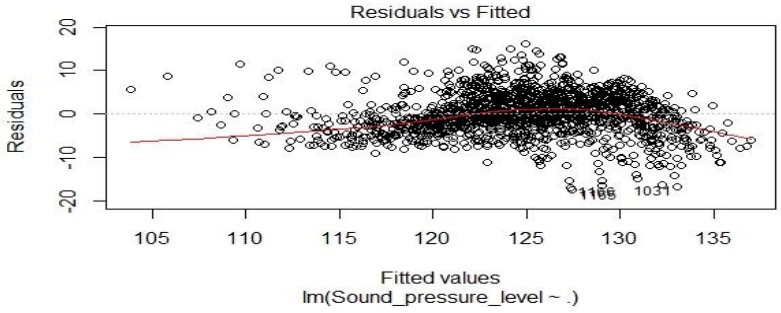
```
> regmodel <- lm(Sound_pressure_level ~ ., data = mydata)
> summary(regmodel)

Call:
lm(formula = Sound_pressure_level ~ ., data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max
-17.480  -2.882  -0.209   3.152  16.064

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.328e+02  5.447e-01  243.87   <2e-16 ***
Frquency.Hz.         -1.282e-03  4.211e-05  -30.45   <2e-16 ***
Angle_of_Attack      -4.219e-01  3.890e-02  -10.85   <2e-16 ***
Chord_Length         -3.569e+01  1.630e+00  -21.89   <2e-16 ***
Free_stream_velocity  9.985e-02  8.132e-03   12.28   <2e-16 ***
Displacement         -1.473e+02  1.501e+01   -9.81   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.809 on 1497 degrees of freedom
Multiple R-squared:  0.5157,   Adjusted R-squared:  0.5141
F-statistic: 318.8 on 5 and 1497 DF,  p-value: < 2.2e-16
```

~ . tells lm to use all the independent variables. Let's understand the regression output in detail:

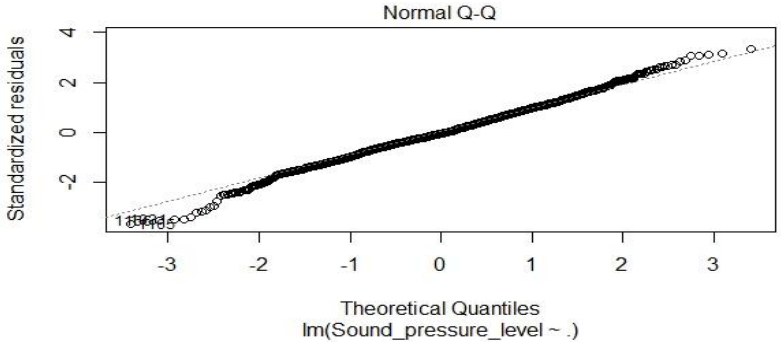
- **Intercept** - This is the βo value. It's the prediction made by model when all the independent variables are set to zero.
- **Estimate** - This represents regression coefficients for respective variables. It's the value of slope. Let's interpret it for Chord_Length. We can say, when Chord_Length is increased by 1 unit, holding other variables constant, Sound_pressure_level decreases by a value of -35.69.
- **Std. Error** - This determines the level of variability associated with the estimates. Smaller the standard error of an estimate is, more accurate will be the predictions.
- **t value** - t statistic is generally used to determine variable significance, i.e. if a variable is significantly adding information to the model. t value > 2 suggests the variable is significant. I used it as an optional value as the same information can be extracted from the p value.
    - **p value** - It's the probability value of respective variables determining their significance in the model. p value < 0.05 is always desirable.

The adjusted R^2 implies that our model explains ~51% total variance in the data. And, the overall p value of the model is significant

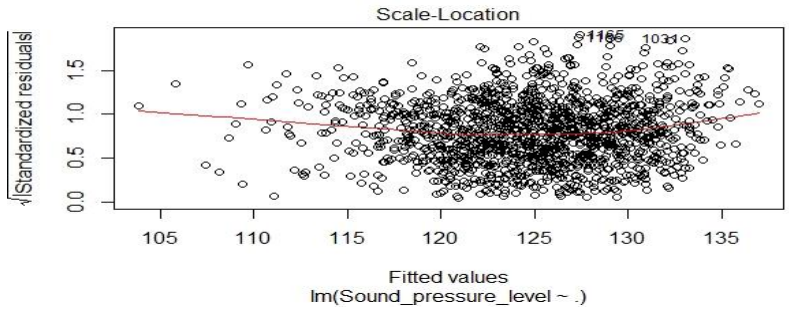
```
> plot (regmodel)
```



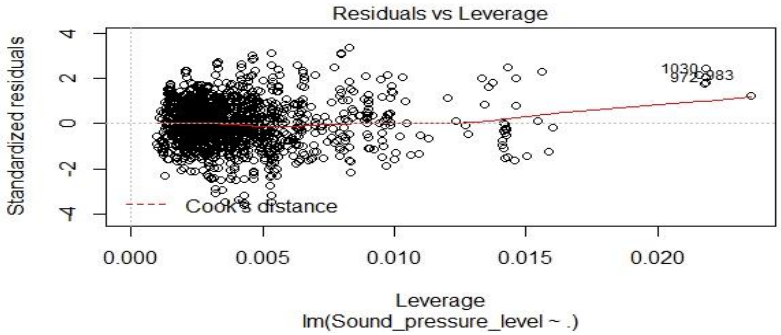

```
Hit <Return> to see next plot:
```



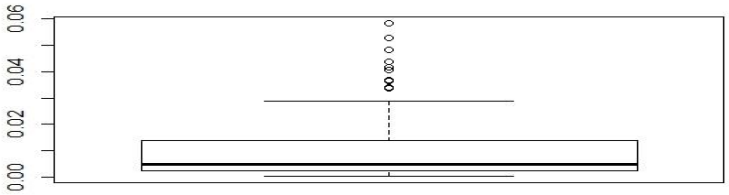
Hit <Return> to see next plot:



Hit <Return> to see next plot:



Hit <Return> to see next plot:




```
> set.seed(1)
> d <- sample ( x = nrow(mydata), size = nrow(mydata)*0.7)
> train <- mydata[d,] #1052 rows
> test <- mydata[-d,] #451 rows
> regmodel <- lm (Sound_pressure_level ~.,data = train)
> summary(regmodel)

Call:
lm(formula = Sound_pressure_level ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-17.3763  -2.8799  -0.2483   3.0800  16.2065
```

```
Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.329e+02  6.423e-01 206.965  < 2e-16 ***
Frquency.Hz.         -1.273e-03  4.751e-05 -26.790  < 2e-16 ***
Angle_of_Attack      -4.378e-01  4.615e-02  -9.486  < 2e-16 ***
Chord_Length         -3.620e+01  1.877e+00 -19.284  < 2e-16 ***
Free_stream_velocity  1.005e-01  9.574e-03  10.494  < 2e-16 ***
Displacement         -1.437e+02  1.776e+01  -8.088 1.66e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.727 on 1046 degrees of freedom
Multiple R-squared:  0.5383,   Adjusted R-squared:  0.5361
F-statistic: 243.9 on 5 and 1046 DF,  p-value: < 2.2e-16

> regpred <- predict(regmodel, test)
> library(Metrics)
> rmse(actual = test$Sound_pressure_level,predicted = regpred)
[1] 4.995263
> d <- boxplot(train$Displacement,varwidth = T,outline = T,border = T,plot = T)
> d$out #enlist outlier observations
  [1] 0.0528487 0.0408268 0.0368233 0.0337792 0.0368233 0.0368233 0.0528487 0.0364
840
  [9] 0.0437259 0.0437259 0.0337792 0.0584113 0.0337792 0.0418756 0.0437259 0.0364
840
 [17] 0.0418756 0.0528487 0.0418756 0.0364840 0.0584113 0.0528487 0.0341183 0.0368
233
 [25] 0.0528487 0.0584113 0.0418756 0.0408268 0.0483159 0.0337792 0.0584113 0.0341
183
 [33] 0.0437259 0.0337792 0.0337792 0.0341183 0.0364840 0.0483159 0.0584113 0.0418
756
 [41] 0.0368233 0.0483159 0.0368233 0.0341183 0.0528487 0.0483159 0.0364840 0.0337
792
 [49] 0.0341183 0.0418756 0.0341183 0.0408268 0.0341183 0.0341183 0.0528487 0.0408
268
 [57] 0.0408268 0.0584113 0.0418756 0.0337792 0.0584113 0.0437259 0.0437259 0.0337
792
 [65] 0.0408268 0.0437259 0.0368233 0.0437259 0.0483159 0.0408268 0.0368233 0.0418
756
 [73] 0.0341183 0.0408268 0.0364840 0.0418756 0.0584113 0.0341183 0.0437259 0.0584
113
 [81] 0.0364840 0.0364840 0.0483159 0.0528487 0.0584113 0.0528487 0.0483159 0.0437
259
 [89] 0.0368233 0.0418756 0.0584113 0.0418756 0.0437259 0.0418756 0.0483159 0.0408
268

 [97]  0.0483159  0.0408268  0.0408268  0.0584113  0.0418756  0.0483159  0.0408268
0.0483159
```