Introduction

"The road to machine learning starts with Regression. Are you ready?"

If you are aspiring to become a data scientist, regression is the first algorithm you need to ~~learn~~ master. Not just to clear job interviews, but to solve real world problems. Till today, a lot of consultancy firms continue to use regression techniques at a larger scale to help their clients. No doubt, it's one of the easiest algorithms to learn, but it requires persistent effort to get to the master level.

Running a regression model is a no-brainer. A simple model <- y~x does the job. But optimizing this model for higher accuracy is a real challenge. Let's say your model gives adjusted $R^2 = 0.678$; how will you improve it?

In this article, I'll introduce you to crucial concepts of regression analysis with practice in R. Data is given for download below. Once you are finished reading this article, you'll able to build, improve, and optimize regression models on your own. Regression has several types; however, in this article I'll focus on linear and multiple regression.

Note: This article is best suited for people new to machine learning with requisite knowledge of statistics. You should have R installed in your laptops.

Table of Contents

What is Regression ? How does it work ?

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. Don't worry. There are several tricks (we'll learn shortly) we can use to obtain convincing results.

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
$\beta 1$ - Slope
$\in$ - Error

βo and $\beta 1$ 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
$\beta 1$ - This is the slope term. It explains the change in Y when X changes by 1 unit. $\in$ - 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 ($\in$) 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 ($\in$) 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 $\sum[\text{Actual}(y) - \text{Predicted}(y')]^2$ by finding the best possible value of regression coefficients ($\beta 0$, $\beta 1$, 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.

Let's understand OLS in detail using an example:

We are given a data set with 100 observations and 2 variables, namely Heightand Weight. We need to predict weight(y) given height(x1). The OLS equation can we written as:

$$Y = \beta o + \beta 1(\text{Height}) + \in$$

When using R, Python or any computing language, you don't need to know how these coefficients and errors are calculated. As a matter of fact, most people don't care. But you must know, and that's how you'll get close to becoming a master.

The formula to calculate these coefficients is easy. Let's say you are given the data, and you don't have access to any statistical tool for computation. Can you still make any prediction? Yes!

The most intuitive and closest approximation of Y is **mean of Y,** i.e. even in the worst case scenario our predictive model should at least give higher accuracy than mean prediction. The formula to calculate coefficients goes like this:

$$\beta 1 = \Sigma(xi - xmean)(yi-ymean)/ \Sigma (xi - xmean)^2 \text{ where } i= 1 \text{ to } n \text{ (no. of obs.)}$$

$$\beta o = ymean - \beta 1(xmean)$$

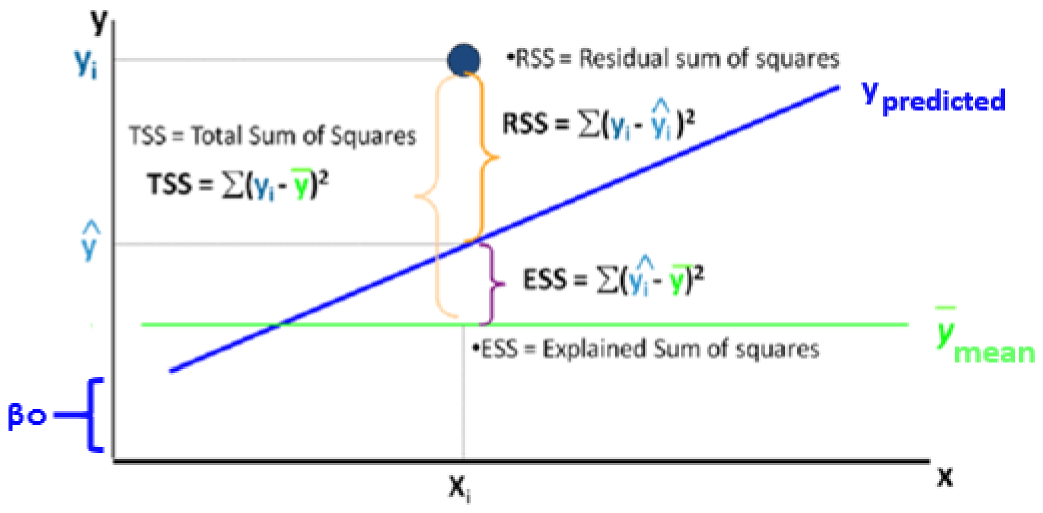
Now you know ymean plays a crucial role in determining regression coefficients and furthermore accuracy. In OLS, the error estimates can be divided into three parts:

Residual Sum of Squares (RSS) - $\sum[\text{Actual}(y) - \text{Predicted}(y)]^2$

Explained Sum of Squares (ESS) - $\sum[\text{Predicted}(y) - \text{Mean}(ymean)]^2$

Total Sum of Squares (TSS) - $\sum[\text{Actual}(y) - \text{Mean}(ymean)]^2$

Anatomy of Regression Errors



y
y_i
•RSS = Residual sum of squares
$y_{predicted}$
$RSS = \sum(y_i - \hat{y}_i)^2$
TSS = Total Sum of Squares
$TSS = \sum(y_i - \bar{y})^2$
$\hat{y}$
$ESS = \sum(\hat{y}_i - \bar{y})^2$
$\bar{y}_{mean}$
•ESS = Explained Sum of squares
βo
X_i
x

The most important use of these error terms is used in the calculation of the Coefficient of Determination (R^2).

`R² = 1 - (SSE/TSS)`

R^2 metric tells us the amount of variance explained by the independent variables in the model. In the upcoming section, we'll learn and see the importance of this coefficient and more metrics to compute the model's accuracy.

What are the assumptions made in regression ?

As we discussed above, 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 ∈t must not indicate the at error at ∈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?

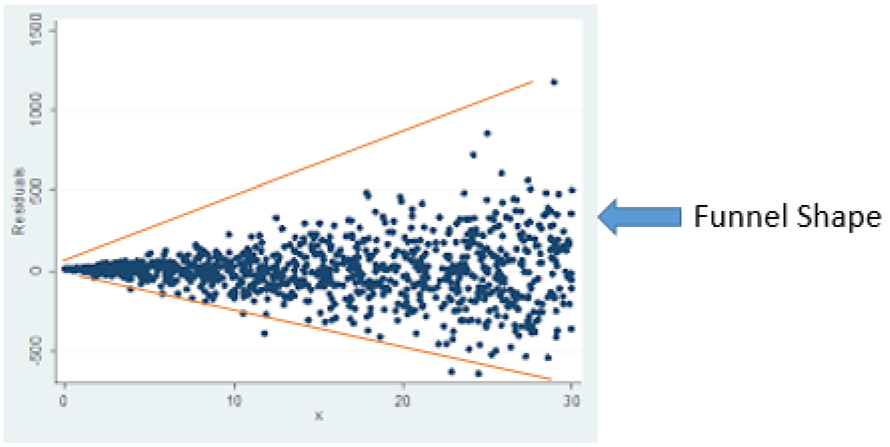
Once these assumptions get violated, regression makes biased, erratic predictions. I'm sure you are tempted to ask me, "How do I know these assumptions are getting violated?"

Of course, 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:

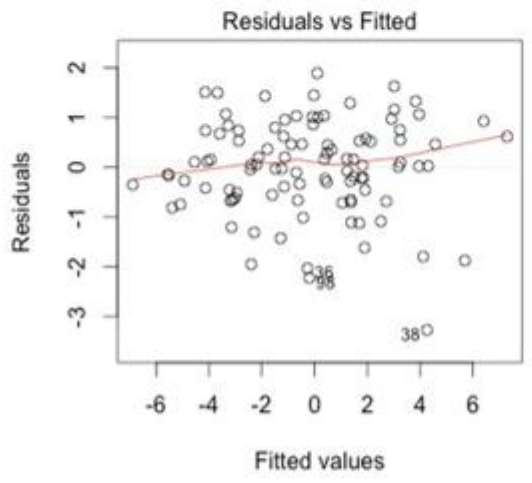
1. **Residual vs. Fitted Values Plot**

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.

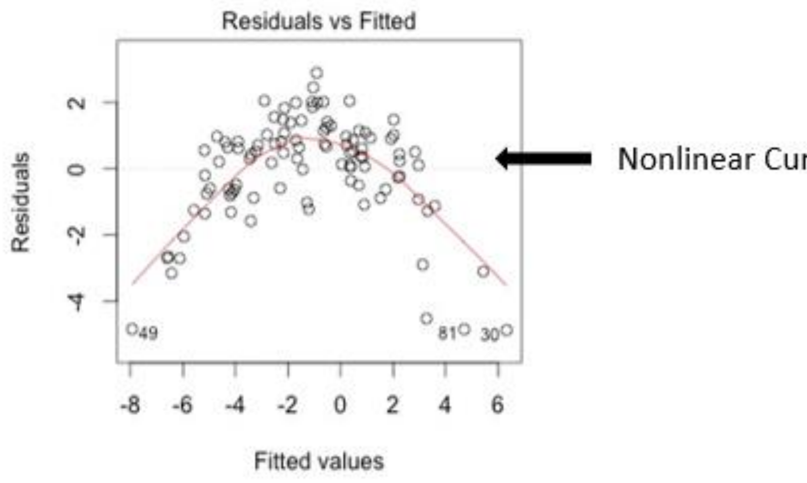
Plot Showing Heteroskedasticity



Data is Linear

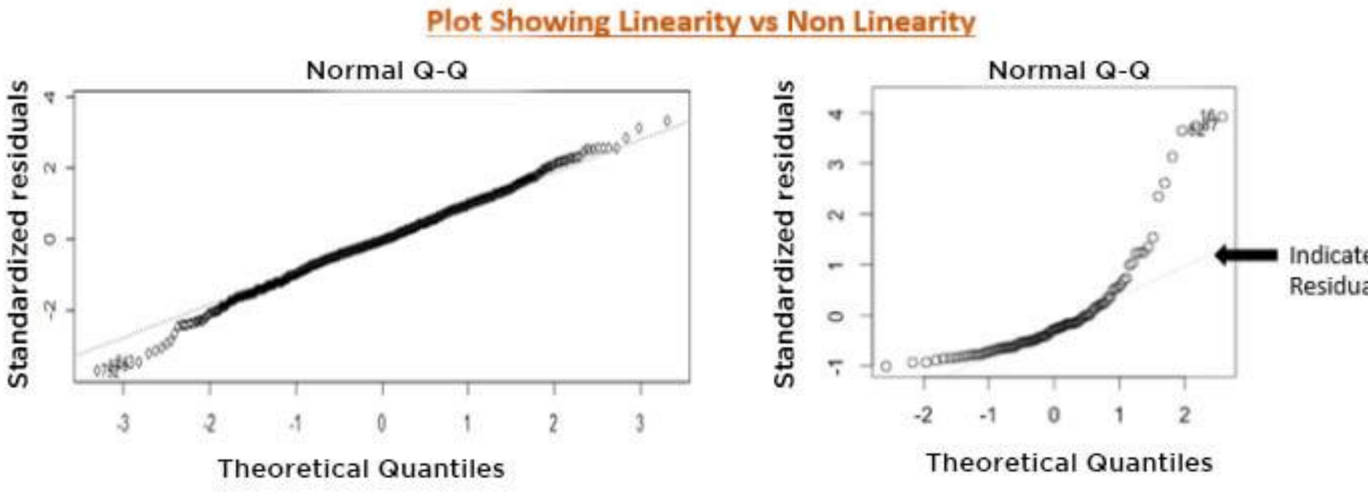


Data is Non-Linear



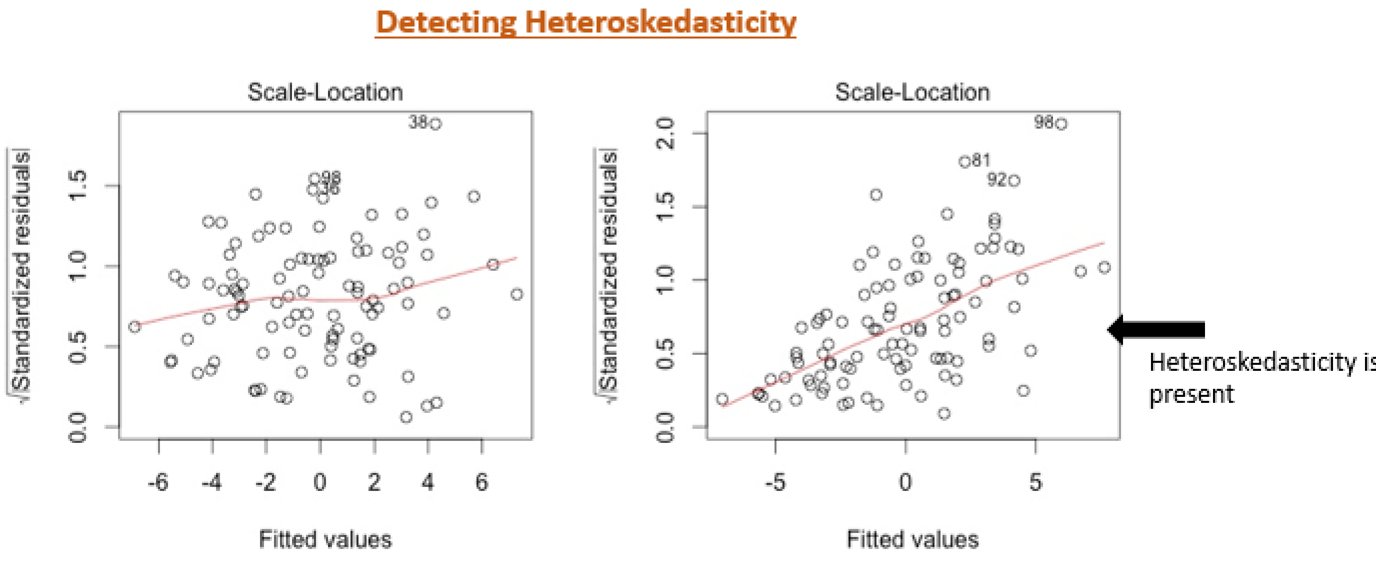
2. **Normality Q-Q Plot**

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 you 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. But if you are adamant at using regression, following are some tips you can 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 you 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 $(30-10)^2 = 400$.
   - **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 $\sqrt{(30-10)^2} = 20$. 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.

Solving a Regression Problem

Let's use our theoretical knowledge and create a model practically. As mentioned above, you should install R in your laptops. I've taken the data set from UCI Machine Learning repository. Originally, the data set is available in .txt file. To save you some time, I've converted it into .csv, and you can download it **here**.

Let's load the data set and do initial data analysis:

```
#set working directory
> path <- "C:/Users/Data/UCI"
> setwd(path)

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

This data has 5 independent variables and Sound_pressure_level as the dependent variable (to be predicted). In predictive modeling, we should always check missing values in data. If any data is missing, we can use methods like mean, median, and predictive modeling imputation to make up for missing data.
#check missing values
> colSums(is.na(mydata))
This data set has no missing values. Good for us! Now, to avoid multicollinearity, let's check correlation matrix.
> cor(mydata)
After you see carefully, you'd infer that Angle_of_Attack and Displacement show 75% correlation. It's up to us if we should consider this correlation % as a damaging level. Usually, correlation above 80% (subjective) is considered higher. Therefore, we can forego this combination and won't remove any variable.

In R, the base function lm is used for regression. We can run regression on this data by:

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

```
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:

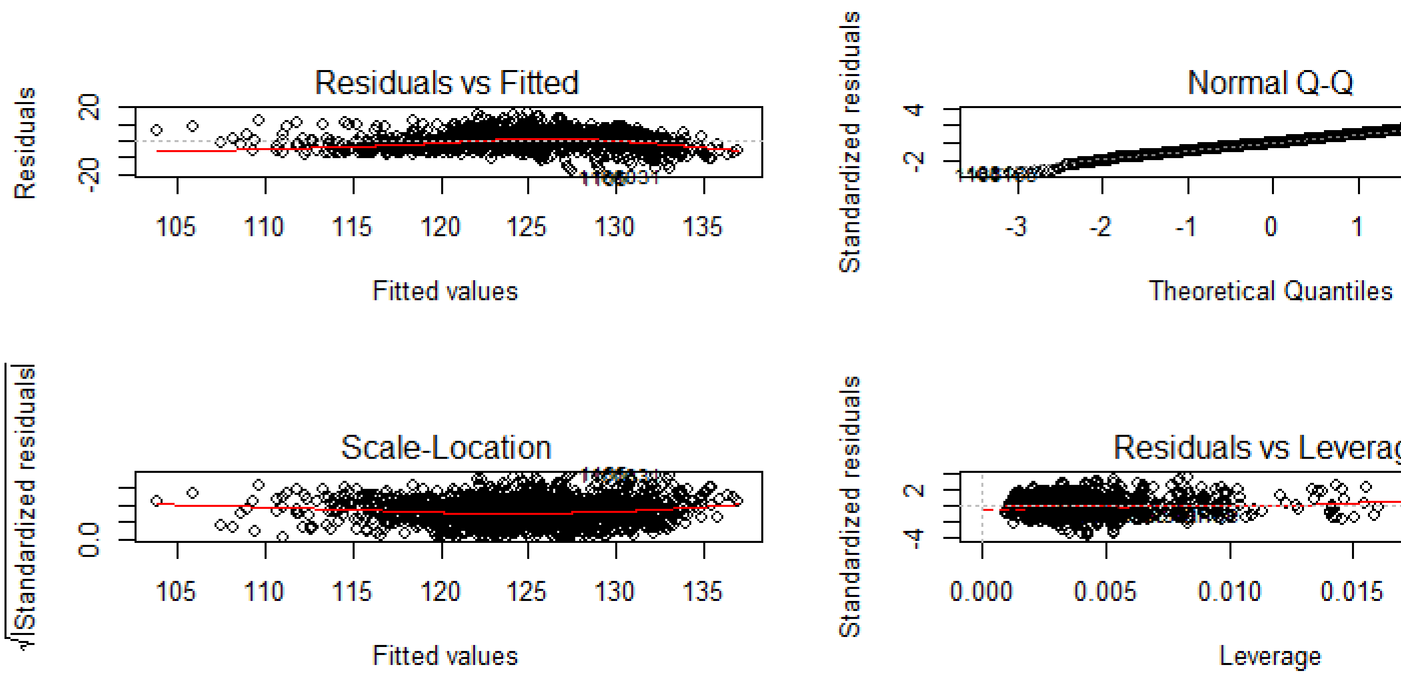
- **Intercept** - This is the βo value. It's the prediction made by model when all the independent variables are set to zero.

- **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. Can we still improve this model ? Let's try to do it. Now, we'll check the residual plots, understand the pattern and derive actionable insights (if any):

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

```
> #create residual plots
> plot (regmodel)
```



Among all, Residual vs. Fitted value catches my attention. Not exactly though, but I see signs of heteroskedasticity in this data. Remember funnel shape? You can see a similar pattern. To overcome this situation, we'll build another model with log(y).

```
> regmodel <- update(regmodel, log(Sound_pressure_level)~.)
> summary(regmodel)
```

```
Call:
lm(formula = log(Sound_pressure_level) ~ Frquency(Hz) + Angle_of_Attack +
Chord_Length + Free_stream_velocity + Displacement, data = mydata)

Residuals:
     Min         1Q       Median       3Q         Max
-0.146939 -0.023272 -0.000701 0.025425 0.122213

Coefficients:
                          Estimate     Std. Error   t value   Pr(>|t|)
(Intercept)                4.891e+00    4.393e-03    1113.31   <2e-16 ***
 Frquency(Hz)          -1.054e-05    3.396e-07    -31.05    <2e-16 ***
Angle_of_Attack          -3.369e-03    3.137e-04    -10.74    <2e-16 ***
Chord_Length             -2.878e-01    1.315e-02    -21.89    <2e-16 ***
Free_stream_velocity      8.071e-04    6.559e-05     12.31    <2e-16 ***
Displacement             -1.244e+00    1.211e-01    -10.28    <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03878 on 1497 degrees of freedom
Multiple R-squared: 0.5235, Adjusted R-squared: 0.5219
F-statistic: 329 on 5 and 1497 DF, p-value: < 2.2e-16
```

Though, the improvement isn't significant, we've increased our adjusted R^2 to 52.19%. Also, it looked like that funnel shape wasn't completely evident, thus implying non-severe effect of non-constant variance.

Let's divide the data set into train and test to check our final evaluation metric. We'll keep 70% data in train and 30% in test file. The reason being that we should keep enough data in train so that the model identifies obvious emerging patterns.

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

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

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

```
#convert back to original value
> regpred <- exp(regpred)

> library(Metrics)
> rmse(actual = test$Sound_pressure_level,predicted = regpred)
[1] 5.03423
```

Interpretation of RMSE error will be more helpful while comparing it with other models. Right now, we can't say if 5.03 error is the optimal value we could expect. I've got you started solving regression problems. Now, you should spend more time and try to obtain a lower error rate than 5.03. For example, you should next check for outlier values using a box plot:

```
#save the output of boxplot
> d <- boxplot(train$Displacement,varwidth = T,outline = T,border = T,plot = T)
> d$out #enlist outlier observations
```

Outliers have a substantial impact on regression's accuracy. Treating outliers is a tricky task. It requires in-depth understanding of data to acknowledge the existence of these high leverage points. I would suggest you read more about it, and if you are unable to find a way let me know in comments. I'm excited to see if you can do it!

Summary

My motive in writing this article is to get you started at solving regression problems, with a greater focus on the theoretical aspects. Running an algorithm isn't rocket science, but knowing how it works will surely give you more control over what you do.

In this article, I've discussed the basics and semi-advanced concepts of regression. In addition, I've also explained best practices which you are advised to follow when facing low model accuracy. We learned about regression assumptions, violations, model fit, and residual plots with practical dealing in R. If you are a python user, you can run regression using linear.fit(x_train, y_train) after loading scikit learn library.