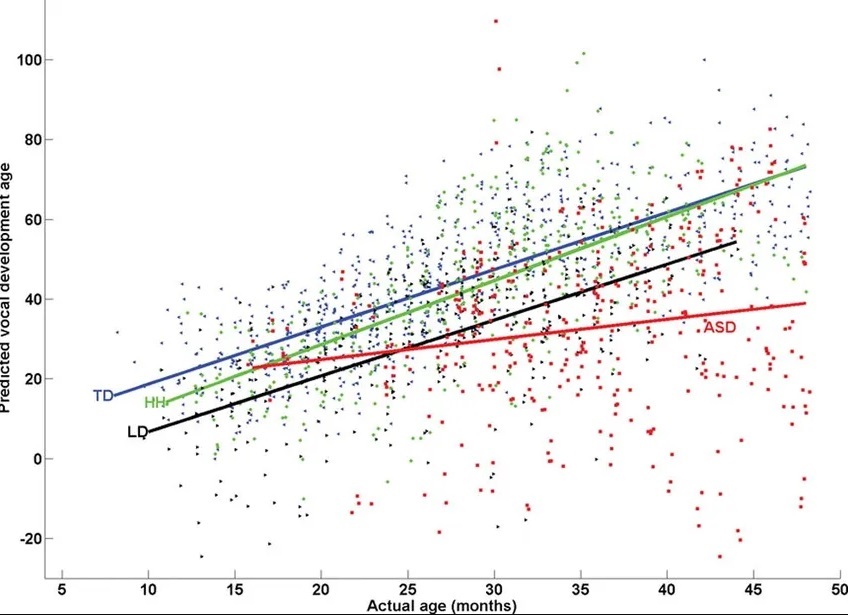
**Unit-III Prediction Linear Regression**

## **What is a Linear Regression?**

Linear regression models are used to show or predict the relationship between adependent and an independent variable. When there are two or more independent variables used in the regression analysis, the model is not simply linear but a multiple regression model.

Simple linear regression is used for predicting the value of one variable by using another variable. A straight line represents the relationship between the two variables with linear regression.

There is a linear relationship between a dependent variable with two or more independent variables in multiple regression. The relationship can also be non-linear, and the dependent and independent variables will not follow a straight line.



Linear and non-linear regression are used to track a response using two or more variables. The non-linear regression is created from assumptions from trial and error and is comparatively difficult to execute.

## Loading required R packages

**Load required packages:**

* tidyverse for data manipulation and visualization
* ggpubr: creates easily a publication ready-plot

library(tidyverse)

library(ggpubr)

theme\_set(theme\_pubr())

## Examples of data and problem

We’ll use the marketing data set [datarium package]. It contains the impact of three advertising medias (youtube, facebook and newspaper) on sales. Data are the advertising budget in thousands of dollars along with the sales. The advertising experiment has been repeated 200 times with different budgets and the observed sales have been recorded.

First install the datarium package using devtools::install\_github("kassmbara/datarium"), then load and inspect the marketing data as follow:

Inspect the data:

# Load the package

data("marketing", package = "datarium")

head(marketing, 4)

## youtube facebook newspaper sales

## 1 276.1 45.4 83.0 26.5

## 2 53.4 47.2 54.1 12.5

## 3 20.6 55.1 83.2 11.2

## 4 181.8 49.6 70.2 22.2

We want to predict future sales on the basis of advertising budget spent on youtube.

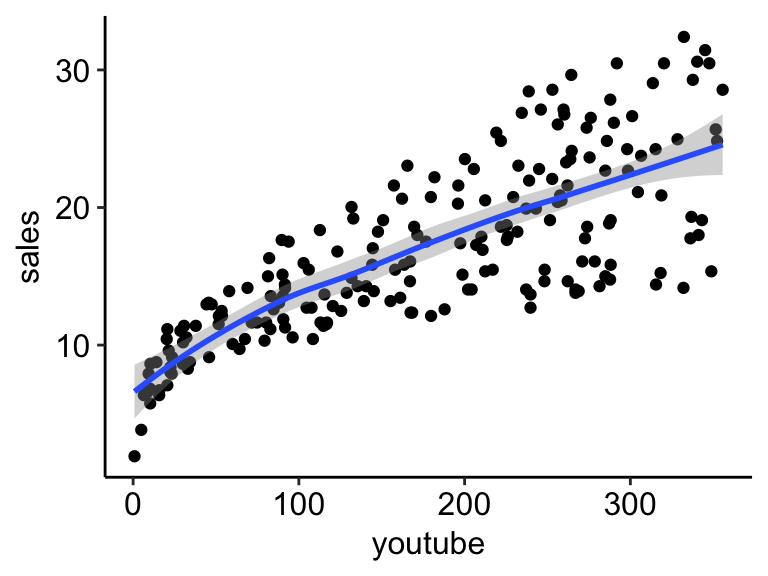
## Visualization

* Create a scatter plot displaying the sales units versus youtube advertising budget.
* Add a smoothed line

ggplot(marketing, aes(x = youtube, y = sales)) +

geom\_point() +

stat\_smooth()



The graph above suggests a linearly increasing relationship between the sales and the youtube variables. This is a good thing, because, one important assumption of the linear regression is that the relationship between the outcome and predictor variables is linear and additive.

It’s also possible to compute the correlation coefficient between the two variables using the R function cor():

cor(marketing$sales, marketing$youtube)

## [1] 0.782

The correlation coefficient measures the level of the association between two variables x and y. Its value ranges between -1 (perfect negative correlation: when x increases, y decreases) and +1 (perfect positive correlation: when x increases, y increases).

A value closer to 0 suggests a weak relationship between the variables. A low correlation (-0.2 < x < 0.2) probably suggests that much of variation of the outcome variable (y) is not explained by the predictor (x). In such case, we should probably look for better predictor variables.

In our example, the correlation coefficient is large enough, so we can continue by building a linear model of y as a function of x.

## **Computation**

The simple linear regression tries to find the best line to predict sales on the basis of youtube advertising budget.

The linear model equation can be written as follow: sales = b0 + b1 \* youtube

The R function lm() can be used to determine the beta coefficients of the linear model:

model <- lm(sales ~ youtube, data = marketing)

model

##

## Call:

## lm(formula = sales ~ youtube, data = marketing)

##

## Coefficients:

## (Intercept) youtube

## 8.4391 0.0475

The results show the intercept and the beta coefficient for the youtube variable.

## **Interpretation**

From the output above:

* the estimated regression line equation can be written as follow: sales = 8.44 + 0.048\*youtube
* the intercept (b0) is 8.44. It can be interpreted as the predicted sales unit for a zero youtube advertising budget. Recall that, we are operating in units of thousand dollars. This means that, for a youtube advertising budget equal zero, we can expect a sale of 8.44 \*1000 = 8440 dollars.
* the regression beta coefficient for the variable youtube (b1), also known as the slope, is 0.048. This means that, for a youtube advertising budget equal to 1000 dollars, we can expect an increase of 48 units (0.048\*1000) in sales. That is, sales = 8.44 + 0.048\*1000 = 56.44 units. As we are operating in units of thousand dollars, this represents a sale of 56440 dollars.

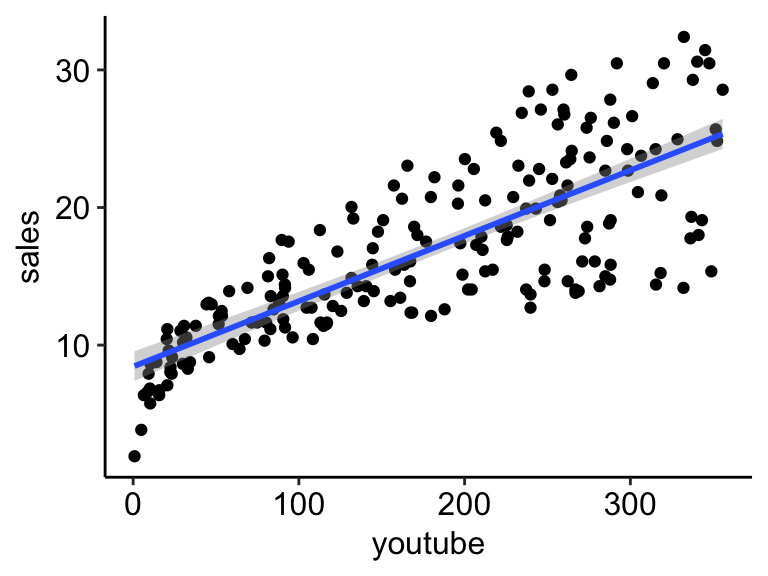
## **Regression line**

To add the regression line onto the scatter plot, you can use the function stat\_smooth() [ggplot2]. By default, the fitted line is presented with confidence interval around it. The confidence bands reflect the uncertainty about the line. If you don’t want to display it, specify the option se = FALSE in the function stat\_smooth().

ggplot(marketing, aes(youtube, sales)) +

geom\_point() +

stat\_smooth(method = lm)



## **Model assessment**

In the previous section, we built a linear model of sales as a function of youtube advertising budget: sales = 8.44 + 0.048\*youtube.

Before using this formula to predict future sales, you should make sure that this model is statistically significant, that is:

* there is a statistically significant relationship between the predictor and the outcome variables
* the model that we built fits very well the data in our hand.

In this section, we’ll describe how to check the quality of a linear regression model.

### **Model summary**

We start by displaying the statistical summary of the model using the R function summary():

summary(model)

##

## Call:

## lm(formula = sales ~ youtube, data = marketing)

##

## Residuals:

## Min 1Q Median 3Q Max

## -10.06 -2.35 -0.23 2.48 8.65

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 8.43911 0.54941 15.4 <2e-16 \*\*\*

## youtube 0.04754 0.00269 17.7 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 3.91 on 198 degrees of freedom

## Multiple R-squared: 0.612, Adjusted R-squared: 0.61

## F-statistic: 312 on 1 and 198 DF, p-value: <2e-16

The summary outputs shows 6 components, including:

* **Call**. Shows the function call used to compute the regression model.
* **Residuals**. Provide a quick view of the distribution of the residuals, which by definition have a mean zero. Therefore, the median should not be far from zero, and the minimum and maximum should be roughly equal in absolute value.
* **Coefficients**. Shows the regression beta coefficients and their statistical significance. Predictor variables, that are significantly associated to the outcome variable, are marked by stars.
* **Residual standard error** (RSE), **R-squared** (R2) and the **F-statistic** are metrics that are used to check how well the model fits to our data.

### **Coefficients significance**

The coefficients table, in the model statistical summary, shows:

* the estimates of the **beta coefficients**
* the **standard errors** (SE), which defines the accuracy of beta coefficients. For a given beta coefficient, the SE reflects how the coefficient varies under repeated sampling. It can be used to compute the confidence intervals and the t-statistic.
* the **t-statistic** and the associated **p-value**, which defines the statistical significance of the beta coefficients.

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 8.4391 0.54941 15.4 1.41e-35

## youtube 0.0475 0.00269 17.7 1.47e-42

**t-statistic and p-values**:

For a given predictor, the t-statistic (and its associated p-value) tests whether or not there is a statistically significant relationship between a given predictor and the outcome variable, that is whether or not the beta coefficient of the predictor is significantly different from zero.

The statistical hypotheses are as follow:

* Null hypothesis (H0): the coefficients are equal to zero (i.e., no relationship between x and y)
* Alternative Hypothesis (Ha): the coefficients are not equal to zero (i.e., there is some relationship between x and y)

Mathematically, for a given beta coefficient (b), the t-test is computed as t = (b - 0)/SE(b), where SE(b) is the standard error of the coefficient b. The t-statistic measures the number of standard deviations that b is away from 0. Thus a large t-statistic will produce a small p-value.

The higher the t-statistic (and the lower the p-value), the more significant the predictor. The symbols to the right visually specifies the level of significance. The line below the table shows the definition of these symbols; one star means 0.01 < p < 0.05. The more the stars beside the variable’s p-value, the more significant the variable.

A statistically significant coefficient indicates that there is an association between the predictor (x) and the outcome (y) variable.

In our example, both the p-values for the intercept and the predictor variable are highly significant, so we can reject the null hypothesis and accept the alternative hypothesis, which means that there is a significant association between the predictor and the outcome variables.

The t-statistic is a very useful guide for whether or not to include a predictor in a model. High t-statistics (which go with low p-values near 0) indicate that a predictor should be retained in a model, while very low t-statistics indicate a predictor could be dropped (P. Bruce and Bruce 2017).

**Standard errors and confidence intervals**:

The standard error measures the variability/accuracy of the beta coefficients. It can be used to compute the confidence intervals of the coefficients.

For example, the 95% confidence interval for the coefficient b1 is defined as b1 +/- 2\*SE(b1), where:

* the lower limits of b1 = b1 - 2\*SE(b1) = 0.047 - 2\*0.00269 = 0.042
* the upper limits of b1 = b1 + 2\*SE(b1) = 0.047 + 2\*0.00269 = 0.052

That is, there is approximately a 95% chance that the interval [0.042, 0.052] will contain the true value of b1. Similarly the 95% confidence interval for b0 can be computed as b0 +/- 2\*SE(b0).

To get these information, simply type:

confint(model)

## 2.5 % 97.5 %

## (Intercept) 7.3557 9.5226

## youtube 0.0422 0.0528

### Model accuracy

Once you identified that, at least, one predictor variable is significantly associated the outcome, you should continue the diagnostic by checking how well the model fits the data. This process is also referred to as the goodness-of-fit

The overall quality of the linear regression fit can be assessed using the following three quantities, displayed in the model summary:

1. The Residual Standard Error (RSE).
2. The R-squared (R2)
3. F-statistic

## rse r.squared f.statistic p.value

## 1 3.91 0.612 312 1.47e-42

1. **Residual standard error** (RSE).

The RSE (also known as the model sigma) is the residual variation, representing the average variation of the observations points around the fitted regression line. This is the standard deviation of residual errors.

RSE provides an absolute measure of patterns in the data that can’t be explained by the model. When comparing two models, the model with the small RSE is a good indication that this model fits the best the data.

Dividing the RSE by the average value of the outcome variable will give you the prediction error rate, which should be as small as possible.

In our example, RSE = 3.91, meaning that the observed sales values deviate from the true regression line by approximately 3.9 units in average.

Whether or not an RSE of 3.9 units is an acceptable prediction error is subjective and depends on the problem context. However, we can calculate the percentage error. In our data set, the mean value of sales is 16.827, and so the percentage error is 3.9/16.827 = 23%.

sigma(model)\*100/mean(marketing$sales)

## [1] 23.2

1. **R-squared and Adjusted R-squared**:

The R-squared (R2) ranges from 0 to 1 and represents the proportion of information (i.e. variation) in the data that can be explained by the model. The adjusted R-squared adjusts for the degrees of freedom.

The R2 measures, how well the model fits the data. For a simple linear regression, R2 is the square of the Pearson correlation coefficient.

A high value of R2 is a good indication. However, as the value of R2 tends to increase when more predictors are added in the model, such as in multiple linear regression model, you should mainly consider the adjusted R-squared, which is a penalized R2 for a higher number of predictors.

* An (adjusted) R2 that is close to 1 indicates that a large proportion of the variability in the outcome has been explained by the regression model.
* A number near 0 indicates that the regression model did not explain much of the variability in the outcome.

1. **F-Statistic**:

The F-statistic gives the overall significance of the model. It assess whether at least one predictor variable has a non-zero coefficient.

In a simple linear regression, this test is not really interesting since it just duplicates the information in given by the t-test, available in the coefficient table. In fact, the F test is identical to the square of the t test: 312.1 = (17.67)^2. This is true in any model with 1 degree of freedom.

The F-statistic becomes more important once we start using multiple predictors as in multiple linear regression.

A large F-statistic will corresponds to a statistically significant p-value (p < 0.05). In our example, the F-statistic equal 312.14 producing a p-value of 1.46e-42, which is highly significant.

### Summary

After computing a regression model, a first step is to check whether, at least, one predictor is significantly associated with outcome variables.

If one or more predictors are significant, the second step is to assess how well the model fits the data by inspecting the Residuals Standard Error (RSE), the R2 value and the F-statistics. These metrics give the overall quality of the model.

**Linear Regression:**   
It is the basic and commonly used type for predictive analysis. It is a statistical approach for modeling the relationship between a dependent variable and a given set of independent variables.  
**These are of two types:**

1. Simple linear Regression
2. Multiple Linear Regression

**Multiple Linear Regression :**   
It is the most common form of Linear Regression. Multiple Linear Regression basically describes how a single response variable Y depends linearly on a number of predictor variables.  
The basic examples where Multiple Regression can be used are as follows: 

1. The selling price of a house can depend on the desirability of the location, the number of bedrooms, the number of bathrooms, the year the house was built, the square footage of the lot, and a number of other factors.
2. The height of a child can depend on the height of the mother, the height of the father, nutrition, and environmental factors.

Multiple linear regression is the extension of the simple linear regression, which is used to predict the outcome variable (y) based on multiple distinct predictor variables (x). With the help of three predictor variables (x1, x2, x3), the prediction of y is expressed using the following equation:

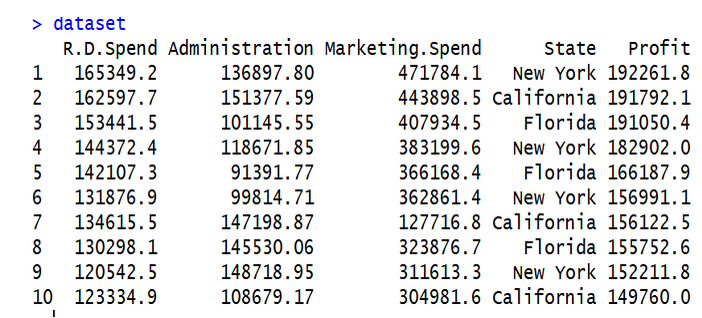
y=b0+b1\*x1+b2\*x2+b3\*x3

The "b" values represent the regression weights. They measure the association between the outcome and the predictor variables. "

**Steps to Perform Multiple Regression in R**

1. ***Data Collection:***The data to be used in the prediction is collected.
2. ***Data Capturing in R:*** Capturing the data using the code and importing a CSV file
3. ***Checking Data Linearity with R:***It is important to make sure that a linear relationship exists between the dependent and the independent variable. It can be done using scatter plots or the code in R
4. ***Applying Multiple Linear Regression in R:***Using code to apply **multiple linear regression in R** to obtain a set of coefficients.
5. ***Making Prediction with R:***A predicted value is determined at the end.

Example:



**# Multiple Linear Regression**

**# Importing the dataset**

**dataset = read.csv('data2.csv')**

**# Encoding categorical data**

**dataset$State = factor(dataset$State,**

**levels = c('New York', 'California', 'Florida'),**

**labels = c(1, 2, 3))**

**dataset$State**

**# Splitting the dataset into the Training set and Test set**

**# install.packages('caTools')**

**library(caTools)**

**set.seed(123)**

**split = sample.split(dataset$Profit, SplitRatio = 0.8)**

**training\_set = subset(dataset, split == TRUE)**

**test\_set = subset(dataset, split == FALSE)**

**# Feature Scaling**

**# training\_set = scale(training\_set)**

**# test\_set = scale(test\_set)**

**# Fitting Multiple Linear Regression to the Training set**

**regressor = lm(formula = Profit ~ .,**

**data = training\_set)**

**# Predicting the Test set results**

**y\_pred = predict(regressor, newdata = test\_set)**

Difference between simple and multiple regression in R

Here’s the data we will use, one year of marketing spend and company sales by month. Download: [CSV](https://www.learnbymarketing.com/wp-content/uploads/2014/12/data-marketing-budget-12mo.csv)

| **Month** | **Spend** | **Sales** |
| --- | --- | --- |
| 1 | 1000 | 9914 |
| 2 | 4000 | 40487 |
| 3 | 5000 | 54324 |
| 4 | 4500 | 50044 |
| 5 | 3000 | 34719 |
| 6 | 4000 | 42551 |
| 7 | 9000 | 94871 |
| 8 | 11000 | 118914 |
| 9 | 15000 | 158484 |
| 10 | 12000 | 131348 |
| 11 | 7000 | 78504 |
| 12 | 3000 | 36284 |

Assuming you’ve downloaded the CSV, we’ll read the data in to R and call it the **dataset** variable

|  |  |
| --- | --- |
| 1  2  3  4  5 | #You may need to use the setwd(directory-name) command to  #change your working directory to wherever you saved the csv.  #Use getwd() to see what your current directory is.  dataset = read.csv("data-marketing-budget-12mo.csv", header=T,  colClasses = c("numeric", "numeric", "numeric")) |

## Simple (One Variable) and Multiple Linear Regression Using lm()

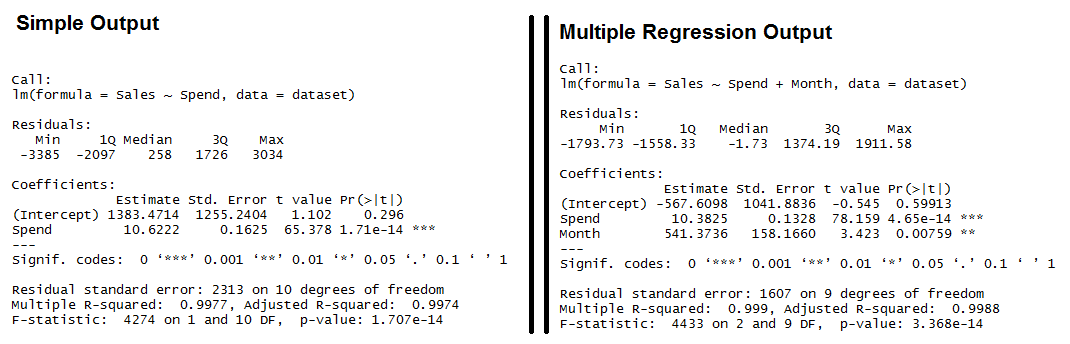
The predictor (or independent) variable for our linear regression will be Spend (notice the capitalized S) and the dependent variable (the one we’re trying to predict) will be Sales (again, capital S).

The lm function really just needs a formula (Y~X) and then a data source.  We’ll use Sales~Spend, data=dataset and we’ll call the resulting linear model “fit”.

|  |  |
| --- | --- |
| 1  2  3  4 | simple.fit = lm(Sales~Spend, data=dataset)  summary(simple.fit)  multi.fit = lm(Sales~Spend+Month, data=dataset)  summary(multi.fit) |

Notices on the multi.fit line the Spend variables is accompanied by the Month variable and a plus sign (+).  The plus sign includes the Month variable in the model as a predictor (independent) variable.

The summary function outputs the results of the linear regression model.

[](https://www.learnbymarketing.com/wp-content/uploads/2014/12/lm-r-regression-summary-output.png)

Output for R’s lm Function showing the formula used, the summary statistics for the residuals, the coefficients (or weights) of the predictor variable, and finally the performance measures including RMSE, R-squared, and the F-Statistic.

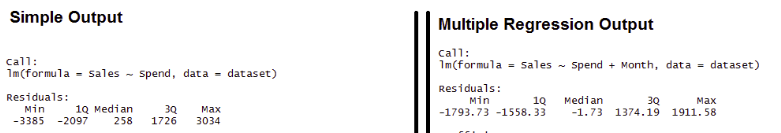
Both models have significant models (see the F-Statistic for Regression) and the Multiple R-squared and Adjusted R-squared are both exceptionally high (keep in mind, this is a simplified example).  We also see that all of the variables are significant (as indicated by the “\*\*”)

## Interpreting R’s Regression Output

* **Residuals**: The section summarizes the residuals, the error between the prediction of the model and the actual results.  Smaller residuals are better.
* **Coefficients**: For each variable and the intercept, a weight is produced and that weight has other attributes like the standard error, a t-test value and significance.
  + **Estimate**: This is the weight given to the variable.  In the simple regression case (one variable plus the intercept), for every one dollar increase in Spend, the model predicts an increase of $10.6222.
  + **Std. Error**: Tells you how precisely was the estimate measured.  It’s really only useful for calculating the t-value.
  + **t-value** and **Pr(>[t])**: The t-value is calculated by taking the coefficient divided by the Std. Error.  It is then used to test whether or not the coefficient is significantly different from zero.  If it isn’t significant, then the coefficient really isn’t adding anything to the model and could be dropped or investigated further.  Pr(>|t|) is the significance level.
* **Performance Measures**: Three sets of measurements are provided.
  + **Residual Standard Error**: This is the standard deviation of the residuals.  Smaller is better.
  + **Multiple / Adjusted R-Square**: For one variable, the distinction doesn’t really matter.  [R-squared](https://www.learnbymarketing.com/definitions/r-squared/) shows the amount of variance explained by the model.  Adjusted R-Square takes into account the number of variables and is most useful for multiple-regression.
  + **F-Statistic**: The F-test checks if at least one variable’s weight is significantly different than zero.  This is a global test to help asses a model.  If the p-value is not significant (e.g. greater than 0.05) than your model is essentially not doing anything.

Need more concrete explanations?  I explain summary output on [this page](https://www.learnbymarketing.com/tutorials/explaining-the-lm-summary-in-r/).

With the descriptions out of the way, let’s start interpreting.



**Residuals**: We can see that the multiple regression model has a smaller range for the residuals: -3385 to 3034 vs. -1793 to 1911.  Secondly the median of the multiple regression is much closer to 0 than the simple regression model.

* **Coefficients**:
  + (Intercept): The intercept is the left over when you average the independent and dependent variable.  In the simple regression we see that the intercept is much larger meaning there’s a fair amount left over.  Multiple regression shows a negative intercept but it’s closer to zero than the simple regression output.
  + Spend: Both simple and multiple regression shows that for every dollar you spend, you should expect to get around 10 dollars in sales.
  + Month: When we add in the Month variable it’s multiplying this variable times the numeric (ordinal) value of the month.  So for every month you are in the year, you add an additional 541 in sales.  So February adds in $1,082 while December adds $6,492 in Sales.
* **Performance Measures**:
  + **Residual Standard Error**: The simple regression model has a much higher standard error, meaning the residuals have a greater variance.  A 2,313 standard error is pretty high considering the average sales is $70,870.
  + **Multiple / Adjusted R-Square**: The R-squared is very high in both cases.  The Adjusted R-square takes in to account the number of variables and so it’s more useful for the multiple regression analysis.
  + **F-Statistic**: The F-test is statistically significant.  This means that both models have at least one variable that is significantly different than zero.

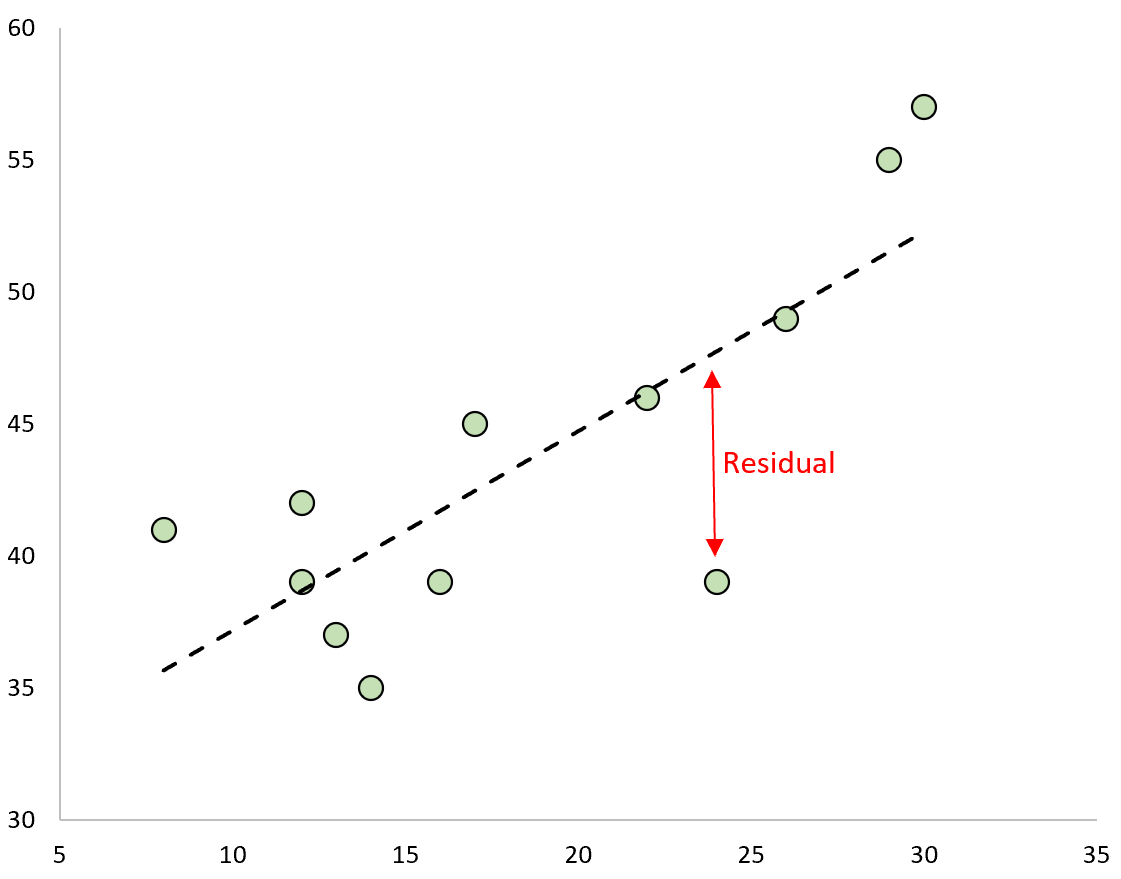
**Calculate Standardized Residuals in R**

A **residual** is the difference between an observed value and a predicted value in a [regression model](https://www.statology.org/linear-regression/).

It is calculated as:

**Residual = Observed value – Predicted value**

If we plot the observed values and overlay the fitted regression line, the residuals for each [observation](https://www.statology.org/observation-in-statistics/) would be the vertical distance between the observation and the regression line:



One type of residual we often use to identify outliers in a regression model is known as a **standardized residual**.

It is calculated as:

**ri  =  ei / s(ei)**  =  **ei / RSE√1-hii**

where:

* **ei:** The ith residual
* **RSE:** The residual standard error of the model
* **hii**: The leverage of the ith observation

step-by-step example of how to calculate standardized residuals in R.

### **Step 1: Enter the Data**

First, we’ll create a small dataset to work with in R:

**#create data**

**data <- data.frame(x=c(8, 12, 12, 13, 14, 16, 17, 22, 24, 26, 29, 30),**

**y=c(41, 42, 39, 37, 35, 39, 45, 46, 39, 49, 55, 57))**

**#view data**

**data**

**x y**

**1 8 41**

**2 12 42**

**3 12 39**

**4 13 37**

**5 14 35**

**6 16 39**

**7 17 45**

**8 22 46**

**9 24 39**

**10 26 49**

**11 29 55**

**12 30 57**

### **Step 2: Fit the Regression Model**

Next, we’ll use the **lm()** function to fit a [simple linear regression model](https://www.statology.org/simple-linear-regression-in-r/):

**#fit model**

**model <- lm(y ~ x, data=data)**

**#view model summary**

**summary(model)**

**Call:**

**lm(formula = y ~ x, data = data)**

**Residuals:**

**Min 1Q Median 3Q Max**

**-8.7578 -2.5161 0.0292 3.3457 5.3268**

**Coefficients:**

**Estimate Std. Error t value Pr(>|t|)**

**(Intercept) 29.6309 3.6189 8.188 9.6e-06 \*\*\***

**x 0.7553 0.1821 4.148 0.00199 \*\***

**---**

**Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1**

**Residual standard error: 4.442 on 10 degrees of freedom**

**Multiple R-squared: 0.6324, Adjusted R-squared: 0.5956**

**F-statistic: 17.2 on 1 and 10 DF, p-value: 0.001988**

### **Step 3: Calculate the Standardized Residuals**

Next, we’ll use the built-in **rstandard()** function to calculate the standardized residuals of the model:

**#calculate the standardized residuals**

**standard\_res <- rstandard(model)**

**#view the standardized residuals**

**standard\_res**

**1 2 3 4 5 6**

**1.40517322 0.81017562 0.07491009 -0.59323342 -1.24820530 -0.64248883**

**7 8 9 10 11 12**

**0.59610905 -0.05876884 -2.11711982 -0.06655600 0.91057211 1.26973888**

We can add the standardized residuals back to the original data frame if we’d like:

**#column bind standardized residuals back to original data frame**

**final\_data <- cbind(data, standard\_res)**

**#view data frame**

**x y standard\_res**

**1 8 41 1.40517322**

**2 12 42 0.81017562**

**3 12 39 0.07491009**

**4 13 37 -0.59323342**

**5 14 35 -1.24820530**

**6 16 39 -0.64248883**

**7 17 45 0.59610905**

**8 22 46 -0.05876884**

**9 24 39 -2.11711982**

**10 26 49 -0.06655600**

**11 29 55 0.91057211**

**12 30 57 1.26973888**

We can then sort each observation from largest to smallest according to its standardized residual to get an idea of which observations are closest to being outliers:

**#sort standardized residuals descending**

**final\_data[order(-standard\_res),]**

**x y standard\_res**

**1 8 41 1.40517322**

**12 30 57 1.26973888**

**11 29 55 0.91057211**

**2 12 42 0.81017562**

**7 17 45 0.59610905**

**3 12 39 0.07491009**

**8 22 46 -0.05876884**

**10 26 49 -0.06655600**

**4 13 37 -0.59323342**

**6 16 39 -0.64248883**

**5 14 35 -1.24820530**

**9 24 39 -2.11711982**

From the results we can see that none of the standardized residuals exceed an absolute value of 3. Thus, none of the observations appear to be outliers.

### **Step 4: Visualize the Standardized Residuals**

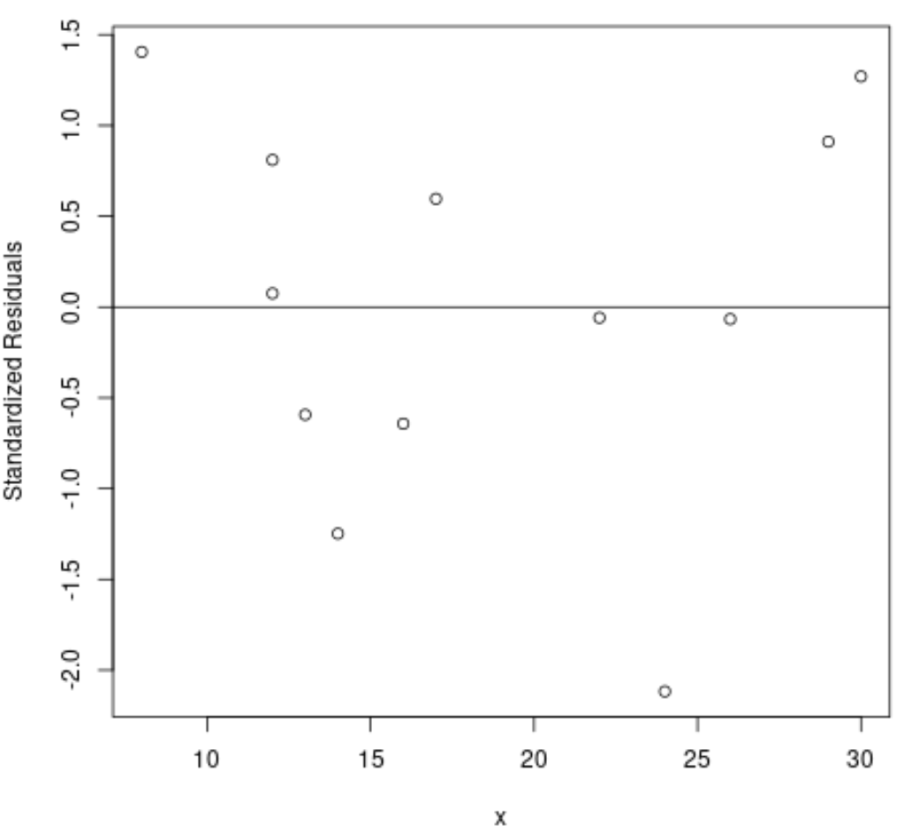
Lastly, we can create a scatterplot to visualize the values for the predictor variable vs. the standardized residuals:

**#plot predictor variable vs. standardized residuals**

**plot(final\_data$x, standard\_res, ylab='Standardized Residuals', xlab='x')**

**#add horizontal line at 0**

**abline(0, 0)**



# How to extract the residuals and predicted values from linear model in R?

The residuals are the difference between actual values and the predicted values and the predicted values are the values predicted for the actual values by the linear model. To extract the residuals and predicted values from linear model, we need to use resid and predict function with the model object.

Consider the below data frame −

## Example

[Live Demo](http://tpcg.io/liMFPPw5)

x1<-rnorm(20,14,3.25)

y1<-rnorm(20,6,0.35)

df1<-data.frame(x1,y1)

df1

## Output

       x1       y1

1  14.565652  6.506233

2  13.350634  6.481486

3  8.636661   5.806754

4  11.495087  6.164963

5  12.159347  6.749101

6  16.642371  6.061237

7  9.137345   6.121711

8  12.616223  5.911341

9  10.109950  5.819494

10 15.953629  6.067601

11 13.579602  6.438686

12 14.708544  5.175576

13 19.234206  6.926994

14 13.539790  5.669169

15 15.101462  6.253202

16 13.812982  6.042699

17 12.680245  6.019822

18 15.292250  6.174533

19 12.759720  5.648624

20 11.371360  5.879896

Creating linear model between x1 and y1 −

Model1<-lm(y1~x1,data=df1)

Finding the residuals and predicted values from Model1 −

resid(Model1)

     1         2            3         4             5       6

0.34576809 0.38483276 -0.04232628 0.16576122 0.71501218 -0.20829562

    7             8          9           10        11        12

0.24633525 -0.14674241 -0.10696233 -0.16575896 0.33000726 -0.99239332

    13         14         15           16           17         18

0.52134132 -0.43741908 0.06459654 -0.07823690 -0.04162379 -0.02409178

   19          20

-0.41699579 -0.11280836

> predict(Model1)

   1       2           3       4        5         6       7       8

6.160465 6.096654 5.849080 5.999202 6.034088 6.269532 5.875376 6.058083

   9         10        11      12       13      14       15       16

5.926456 6.233360 6.108679 6.167970 6.405653 6.106588 6.188605 6.120936

   17       18         19       20

6.061445 6.198625 6.065619 5.992704

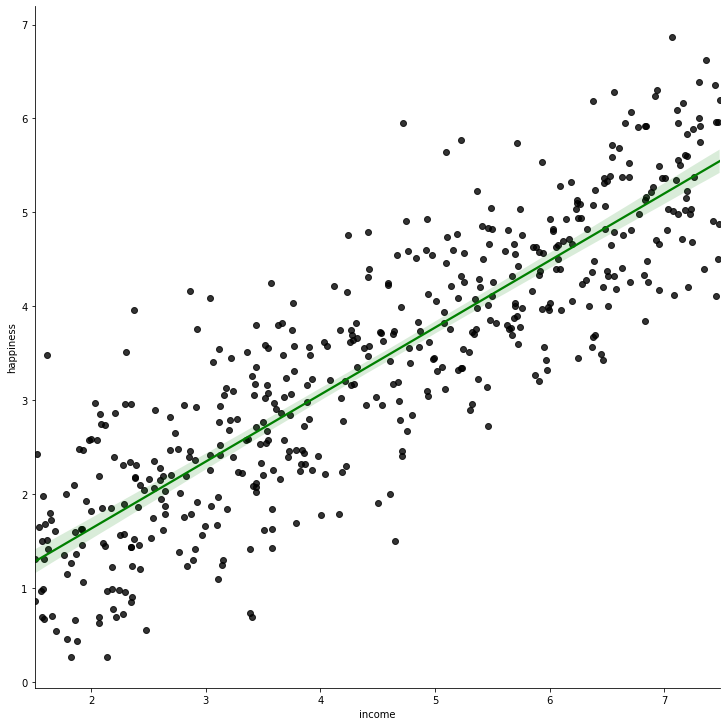
# How to Extract the Intercept from a Linear Regression Model in R

Linear regression is a method of predictive analysis in machine learning. It is basically used to check two things:

1. If a set of predictor variables (independent) does a good job predicting the outcome variable (dependent).
2. Which of the predictor variables are significant in terms of predicting the outcome variable and in what way, which is determined by the magnitude and sign of the estimates respectively.

Linear regression is used with one outcome variable and one or more than one predictor variable. Simple linear regression will work with one outcome and one predictor variable. The simple linear regression model is essentially a linear equation of the form **y = c + b\*x**; where y is the dependent variable (outcome), x is the independent variable (predictor), b is the slope of the line; also known as regression coefficient and c is the intercept; labeled as constant.

A linear regression line is a line that best fits the graph between the predictor variable (independent) and the predicted variable (dependent).



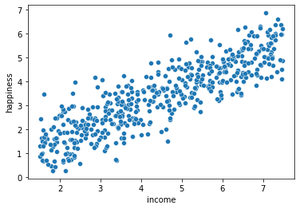
Regression line (solid green) for income vs happiness dataset

In the above diagram, the green line is the best fit line; and it is taken as the regression line for the given dataset.

One of the most popular methods of deciding the regression line is the method of least-squares. This method essentially works to find the best-fit line for the data by minimizing the sum of the squares of the vertical deviations from each data point (the deviation of a point residing on the line is 0). As the deviations are squared, there is no cancellation between the positive and negative values of the deviation.

#### **Approach:**

1. Select a suitable problem statement for linear regression. We will be selecting [income.data\_](https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210518181857/income.data_.csv).
2. Install and load the packages for plotting/visualization. You can visualize the data points to see if the data is suitable for the linear regression.
3. Read the dataset in a data frame. You can also visualize the data frame after reading (example shown in the code below).
4. Create a linear regression model from the data using **lm()** function. Store the created model in a variable.
5. Explore the model.

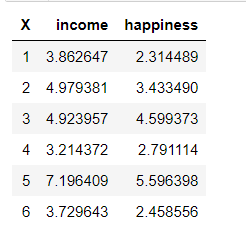


Scatter plot after plotting the dependent and independent variables against each other

**Step 1:** Install and load the required packages. Read and explore the dataset. You can also set the working directory of the notebook using **setwd()** function, passing the path of the directory (where the dataset is stored) as an argument.

|  |
| --- |
| # install the packages and load them  install.packages("ggplot2")  install.packages("tidyverse")  library(ggplot2)  library(tidyverse)    # Read the data into a data frame  dataFrame <- read.csv("income\_data.csv")    # Explore the data frame  head(dataFrame) |

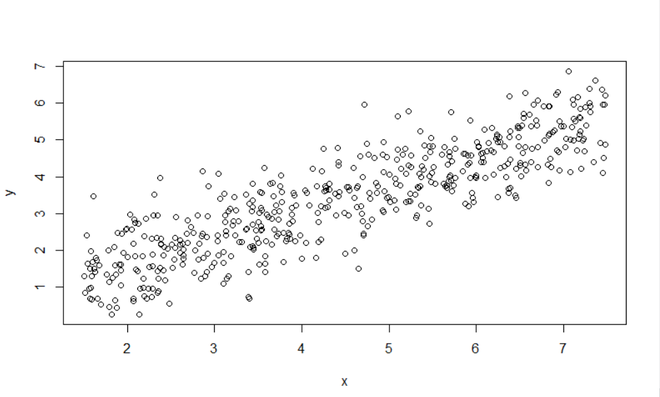
**Output:**



**Step 2:** Separate the variables of the dataset. Visualize the dataset.

|  |
| --- |
| # Allocate the columns to different variables  # x is the independent variable  x <- dataFrame$income    # y is the dependent variable  y <- dataFrame$happiness    # Plot the graph between dependent and independent variable  plot(x, y) |

**Output:**



Graph of X (Income) vs Y (Happiness)

**Step 3:** Clear the linear regression model from the data. Train and see the model.

|  |
| --- |
| # Create the linear model from the data.  # y ~ x denotes y dependent and x is the independent variable  model <- lm(y~x)    # Print the model to check the intercept  model |

**Output:**

Call:

lm(formula = y ~ x)

Coefficients:

(Intercept) x

0.2043 0.7138

As you can see, the value of intercept is 0.2043. But how to obtain this value in a variable?

## Extracting the values of intercept

We can use a summary of the created model to extract the value of the intercept.

**Code:**

|  |
| --- |
| model\_summary <- summary(model)    intercept\_value <- model\_summary$coefficients[1,1]    intercept\_value |

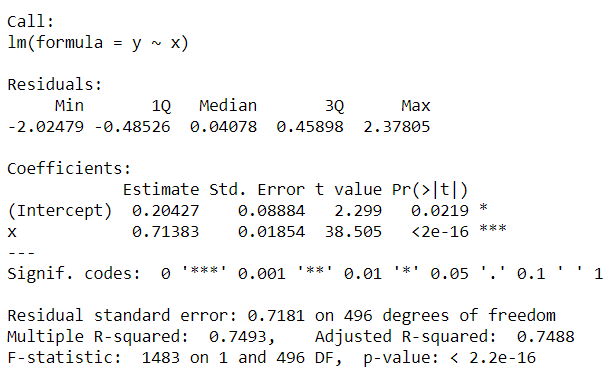
**Output:**

0.204270396204177

If you try to print the summary of the model (model\_summary) variable, you will see the coefficients below. It is a 2D matrix, which stores all the said coefficients. Therefore, [1,1] will correspond to the predicted intercept (of the regression line).

|  |
| --- |
| model\_summary <- summary(model)  model\_summary |

**Output:**



This is how we extract the value of intercept from a linear regression model in R.

**Relationship Between r and R-squared in Linear Regression**

**R-squared** is a measure of how well a linear regression model fits the data. It can be interpreted as the proportion of variance of the outcome Y explained by the linear regression model.

It is a number between 0 and 1 (0 ≤ R2 ≤ 1). The closer its value is to 1, the more variability the model explains. And R2 = 0 means that the model cannot explain any variability in the outcome Y.

On the other hand, the **correlation coefficient r** is a measure that quantifies the strength of the linear relationship between 2 variables.

r is a number between -1 and 1 (-1 ≤ r ≤ 1):

* **A value of r close to -1**: means that there is negative correlation between the variables (when one increases the other decreases and vice versa)
* **A value of r close to 0**: indicates that the 2 variables are not correlated (no linear relationship exists between them)
* **A value of r close to 1**: indicates a positive linear relationship between the 2 variables (when one increases, the other does)

# Correlation coefficient

head(mtcars, 5)

The variables vs and am are categorical variables, so they are removed for this article:

# remove vs and am variables

library(tidyverse)

dat <- mtcars %>%

select(-vs, -am)

# display 5 first obs. of new dataset

head(dat, 5)

## Between two variables

The correlation between 2 variables is found with the cor() function.

Suppose we want to compute the correlation between horsepower (hp) and miles per gallon (mpg):

# Pearson correlation between 2 variables

cor(dat$hp, dat$mpg)

## [1] -0.7761684

Note that the correlation between variables X and Y is equal to the correlation between variables Y and X so the order of the variables in the cor() function does not matter.

The Pearson correlation is computed by default with the cor() function. If you want to compute the Spearman correlation, add the argument method = "spearman" to the cor() function:

# Spearman correlation between 2 variables

cor(dat$hp, dat$mpg,

method = "spearman"

)

## [1] -0.8946646

There are several correlation methods (Run ?cor for more information about the different methods available in the cor() function):

* **Pearson** correlation is often used for [quantitative continuous](https://statsandr.com/blog/variable-types-and-examples/#continuous) variables that have a linear relationship
* **Spearman** correlation (which is actually similar to Pearson but based on the ranked values for each variable rather than on the raw data) is often used to evaluate relationships involving at least one [qualitative ordinal](https://statsandr.com/blog/variable-types-and-examples/#ordinal) variable or two quantitative variables if the link is partially linear
* **Kendall’s tau-b** which is computed from the number of concordant and discordant pairs is often used for qualitative ordinal variables

Ordinary Least Squares (OLS) method is widely used **to estimate the parameters of a linear regression model**. For the validity of OLS estimates, there are assumptions made while running linear regression models.

Ordinary Least Squares regression (OLS) is a common technique for **estimating coefficients of linear regression equations** which describe the relationship between one or more independent quantitative variables and a dependent variable (simple or multiple linear regression).

**What is the logic behind simple linear regression model?**

As the name suggests, linear regression follows the linear mathematical model for determining the value of one dependent variable from value of one given independent variable. Remember the linear equation from school?

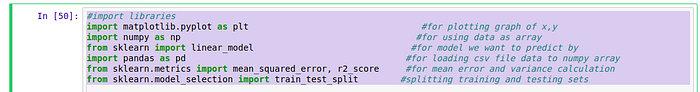
# y=mx+c

where y is the dependent variable, m is slope, x is the independent variable and c is the intercept for a given line.

We also have multiple regression model where multiple independent variables are used to calculate one dependent variable.

I have used Jupyter Notebook for implementation. Any Python IDE can be used of your choice. So let’s get rolling..

Step 1: Importing libraries



Step 1

There are already developed libraries in Python for implementation of Machine Learning models.

First library called matplotlib is used to plot the graph in last step. ”plt” is used as variable name for using this library in code ahead.

sklearn is official machine learning library in python for various model implementation.

numpy is used to convert data into arrays for actual use by sklearn library.

pandas is used to access .csv file of our dataset.

Step 2: Loading dataset



Our dataset is in a .csv file type. Pandas variable pd is used to access the dataset with read\_csv() function.

Step 3: Split to independent and dependent variables



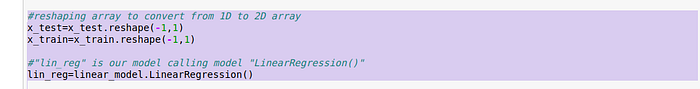
We define x as the independent variable in dataset by iloc(index location) value. [] is used to define array elements. “:” inside [] indicates consider all rows in dataset and separating by using “,” we specify the number of column which we want to use as independent or dependent variable values starting the count from zero in dataset.

Step 4: Splitting data into training and testing data



Now, entire dataset is divided into training and testing set so that prediction does not overfit or underfit and correct values are obtained. train\_test\_split() is inbuilt function from scikit learn for splitting x and y variables data. “test\_size” parameter is used to divide (1/3)rd of entire dataset(30%) into test data and remaining as training data.Setting random\_state as null would not allow random values to be taken from dataset.

Step 5: Choosing the Model



We reshape our independent variable as sklearn expects a 2D array as input.

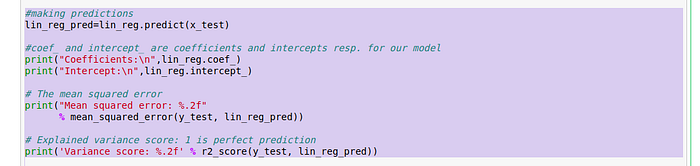
Linear Regression is our model here with variable name of our model as “lin\_reg”. We can try the same dataset with many other models as well. This part varies for any model otherwise all other steps are similar as described here.

Step 6: Fit our model



We now fit our model to the linear regression model by training the model with our independent variable and dependent variables.

Step 7: Predict the output



Finally our model predicts the dependent variable “lin\_reg\_pred” using the test values of independent variable.

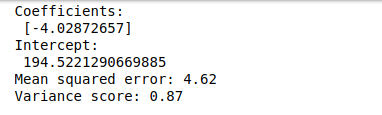
We can see the coefficient,intercept values for our outlier and also the mean squared error and variance for the predicted values(lin\_reg\_pred) and actual test value of dependent variable(y\_test). Inbuilt methods does the math with the predefined formulae for each value.

Step 8: Plot the graph

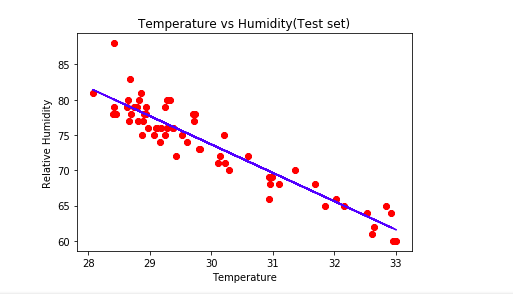


We ultimately want to visualize the actual data values and predicted data values in a graphical format. “plt”, matplotlib variable, is used to plot points using “scatter()” and outlier using “plot()” functions.

Output might vary depending on various system features. Output I got is as follows:



Output



Output Graph