Linear Regression

Oliver Imhans

2022-12-21

For convenience, I will add all the necessary libraries at the very beginning.

```
library(tidyverse)
library(dplyr)
library(corrgram)
library(DataExplorer)
library(ppcor)
library(caTools)
library(ggplot2)
library(corrplot)
library(data.table)
library(plotly)
library(lm.beta)
```

In mathematics, regression is a statistical technique that is employed when the relationship between dependent variables and independent variables is considered. This process is used to determine if the changes in the dependent variables are connected with any of the independent variables.

Linear regression

This is the most commonly used type of predictive analysis. In simple terms, this is a linear (arranged along a straight line) approach for relationship modeling between two variables. The variables are always **dependent** and **independent**. It is important to note that the order of the variables matters. The independent variable belongs on the x-axis, while the dependent variable belongs on the y-axis. There are two types of linear regression:

i. Simple Linear Regression

ii. Multiple Linear Regression

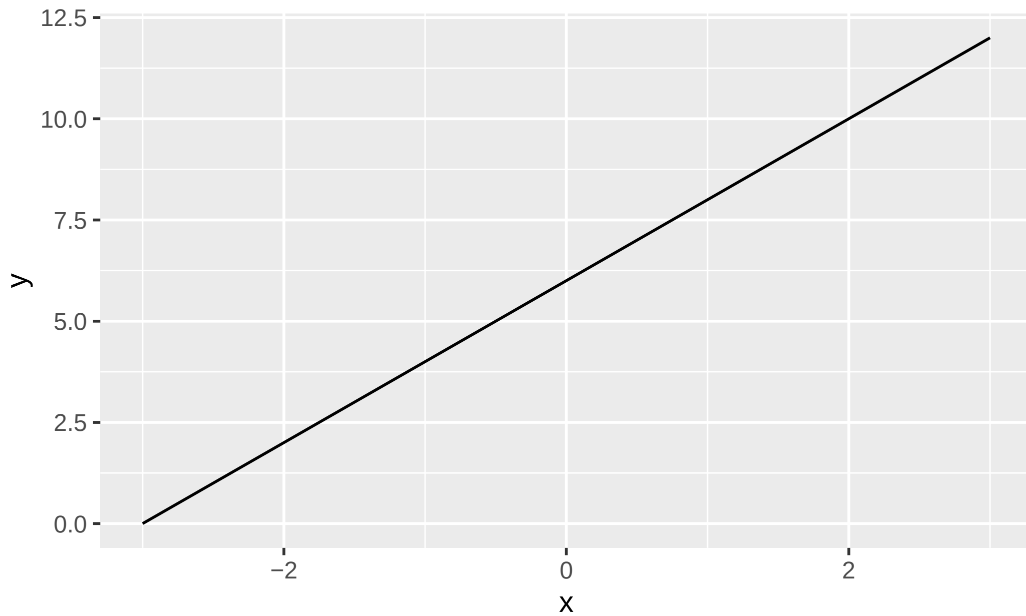
The linear regression for two variables is based on the linear equation $y = mx + c$ where m and c are constants.

The graph of a linear equation of the form above is a *straight line*.

Example Plot the graph of $y = 2x + 6$ using the range -3 to 3.

```
# First define the equation
lin_equ <- function(x){2*x+6}

# Then plot the equation
ggplot(data.frame(x=c(-3, 3)), aes(x=x)) +
  stat_function(fun=lin_equ)
```



How to perform Simple Linear Regression

The formula for linear regression is $y = b_1 X + b_0$. But in a more standard form, the complete linear regression model is:

$$y = b_1 X + b_0 + \epsilon$$

where:

y is the predicted value

b_0 is the intercept.

b_1 is the regression coefficient

X is the independent variable

ϵ is the error of the estimate.

The aim of linear regression is to find the line of best fit that goes through the data set. This is achieved by searching for b_1 the regression coefficient that will minimize the ϵ the error of the model.

In the world of Data Science, *linear regression is an algorithm* that predicts the outcome from the linear relationship between the independent variables and dependent variables. From the foregoing, linear regression is classified as a supervised learning algorithm. There are some benefits to using linear regression
1. It is easily scalable.
2. It is easily implemented.
3. It is relatively straightforward.

The example below will focus on how to use R to create a regression model. **R linear regression uses the lm() function to create this regression model.** To view this model, the summary() function will be used.

The dataset for this example is available at the link: https://www.kaggle.com/datasets/karthickveerakumar/salary-data-simple-linear-regression?resource=download

```
# Importing the dataset
data = read.csv('salary.csv')
```

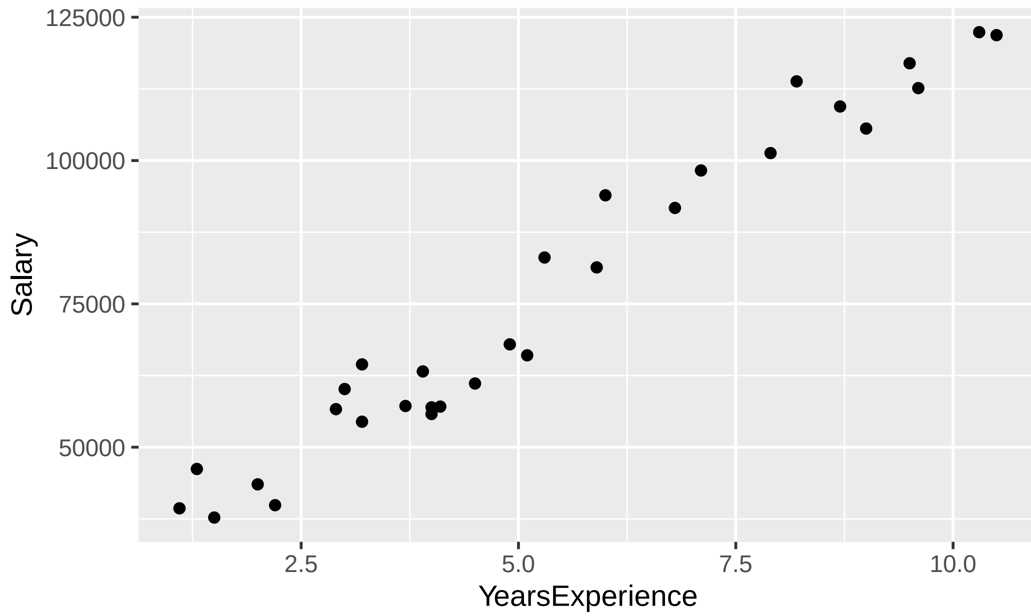
Taking a quick overview of the data

```
summary(data)
```

```
##  YearsExperience      Salary
##  Min.   : 1.100   Min.   : 37731
##  1st Qu.: 3.200   1st Qu.: 56721
##  Median : 4.700   Median : 65237
##  Mean   : 5.313   Mean   : 76003
##  3rd Qu.: 7.700   3rd Qu.:100545
##  Max.   :10.500   Max.   :122391
```

A quick visualization of the data

```
# scatter plot
ggplot(data, aes(x=YearsExperience, y=Salary)) +
    geom_point()
```




```
# Splitting the data into the training data and testing data

split = sample.split(data$Salary, SplitRatio = 0.8)
train_data = subset(data, split == TRUE)
test_data = subset(data, split == FALSE)
```

```
# Fit the Linear Regression model into the training data
model = lm(formula = Salary ~ YearsExperience,
                        data = train_data)
coef(model)
```

```
##     (Intercept) YearsExperience
##        26120.37         9428.08
```

NB The results show the intercept and the beta coefficient for the YearsExperience variable.

Interpretation

From the output above:

i. The estimated regression line equation can be written as follow:

$$Salary = 27266.90 + (9243.10 * YearsExperience)$$

ii. The intercept (b_0) is 27266.90. It can be interpreted as the predicted salary for a zero YearsExperience.

The linear model for this dataset is built and a formula that can be used to make a prediction is also derived. Before using this regression model, I have to ensure that it is statistically significant.

```
# model summary
summary(model)
```

```
##
## Call:
## lm(formula = Salary ~ YearsExperience, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8038.7 -4091.2  -610.8  3688.5 11251.1
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)        26120       2564   10.19 8.57e-10 ***
## YearsExperience     9428        407   23.16  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5842 on 22 degrees of freedom
## Multiple R-squared:  0.9606, Adjusted R-squared:  0.9588
## F-statistic: 536.5 on 1 and 22 DF,  p-value: < 2.2e-16
```

```
# Predicting the Test set results
ypredict = predict(model, newdata = test_data)
```

```
actual_predict <- data.frame(cbind(actual=test_data$Salary, predicted=ypredict))
head(actual_predict)
```

```
##    actual predicted
## 1   39343  36491.26
## 8   54445  56290.23
## 14  57081  64775.50
## 17  66029  74203.58
## 18  83088  76089.19
## 21  91738  90231.31
```

Checking for statistical significance

i. This model can be considered statistically significant only when both the p-Values are less that the pre-determined statistical significance level, which is ideally 0.05. This is visually interpreted by the significant stars at the end of the row. The more stars beside the variable's p-Value, the more significant the variable.

```
reg_p <- function (pvalue)
  {if (class(pvalue) != "lm") stop("Not an object of class 'lm' ")
   f <- summary(pvalue)$fstatistic
   p <- pf(f[1],f[2],f[3],lower.tail=F)
   attributes(p) <- NULL
   return(p)}

reg_p(model)
```

```
```

```
## [1] 6.071725e-17
```

This value corresponds with model summary

ii. R-Squared: the higher the better (> 0.70)

```r
summary(model)$r.squared
```

```
## [1] 0.9606077
```

iii. Adj R-Squared: the higher the better

```r
summary(model)$adj.r.squared
```

```
## [1] 0.9588171
```

iv. F-Statistic: the higher the better

```r
summary(model)$fstatistic
```

```
##    value    numdf    dendf
## 536.4843   1.0000  22.0000
```

v. Std. Error: When it is closer to zero the better.

```r
modelSummary <- summary(model)
modelCoeffs <- modelSummary$coefficients
beta.estimate <- modelCoeffs["YearsExperience", "Estimate"]
std.error <- modelCoeffs["YearsExperience", "Std. Error"]
std.error
```

```
## [1] 407.0472
```

vi. t-statistic: this should be greater than 1.96 for p-value to be less than 0.05.

```r
t_value <- beta.estimate/std.error
t_value
```

```
## [1] 23.16213
```

vii. AIC Lower the better

```r
AIC(model)
```

```
## [1] 488.3143
```

viii. BIC Lower the better

```r
BIC(model)
```

```
## [1] 491.8485
```

ix. MAPE (Mean absolute percentage error): The lower the better

```r
mape <- mean(abs((actual_predict$predicted - actual_predict$actual))/actual_predict$actual)
mape
```

```
## [1] 0.07760593
```

x. MSE (Mean squared error) Lower the better

```r
mean(model$residuals^2)
```

```
## [1] 31281990
```

xi. Min_Max Accuracy => mean(min(actual, predicted)/max(actual, predicted)) The higher the better

```r
min_max_accuracy <- mean(apply(actual_predict, 1, min) / apply(actual_predict, 1, max))
min_max_accuracy
```

```
## [1] 0.9275211
```

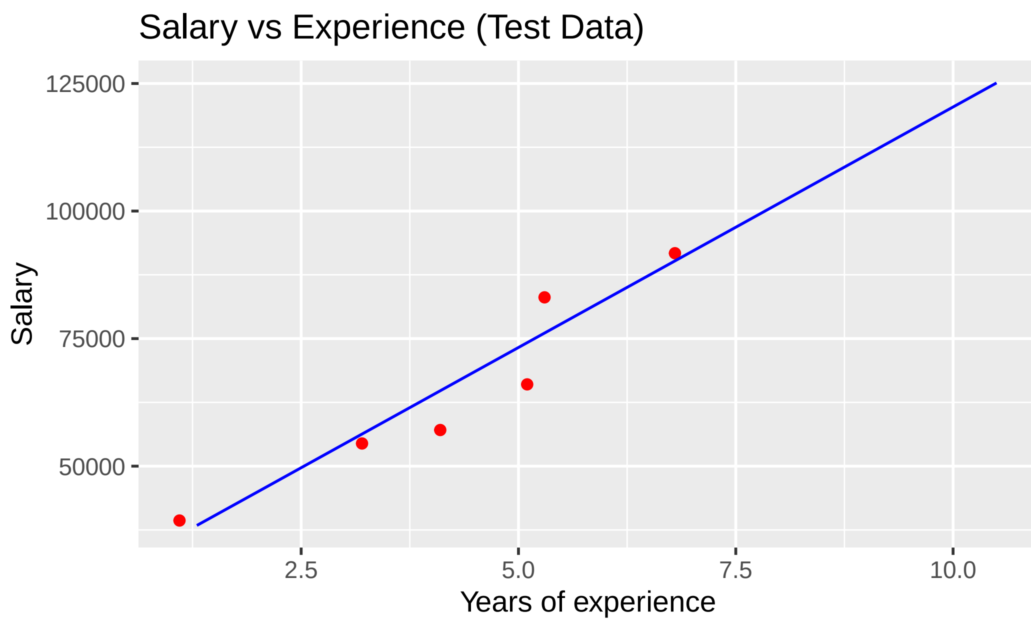
Visualization

```r
# Visualizing the training data outcome
ggplot() + geom_point(aes(x = train_data$YearsExperience,
                 y = train_data$Salary), colour = 'red') +
geom_line(aes(x = train_data$YearsExperience,
y = predict(model, newdata = train_data)), colour = 'blue') +

ggtitle('Salary vs Experience (Train Data)') +
xlab('Years of experience') +
ylab('Salary')
```


```r
# Visualizing the test data outcome
  ggplot() +
  geom_point(aes(x = test_data$YearsExperience, y = test_data$Salary),
             colour = 'red') +
  geom_line(aes(x = train_data$YearsExperience,
             y = predict(model, newdata = train_data)),
```

```
             colour = 'blue') +
  ggtitle('Salary vs Experience (Test Data)') +
  xlab('Years of experience') +
  ylab('Salary')
```



Finding Accuracy

```
rmse<-sqrt(mean(ypredict -data$YearsExperience)^2)
rmse
```

```
## [1] 66341.53
```

Multiple Linear Regression

Multiple linear regression is an extension of simple linear regression used to predict an outcome variable (y) on the basis of multiple distinct predictor variables (x).

With three predictor variables (x), the prediction of y is expressed by the following equation:

$$y = b_0 + b_1 * X_1 + b_2 * X_2 + b_3 * X_3$$

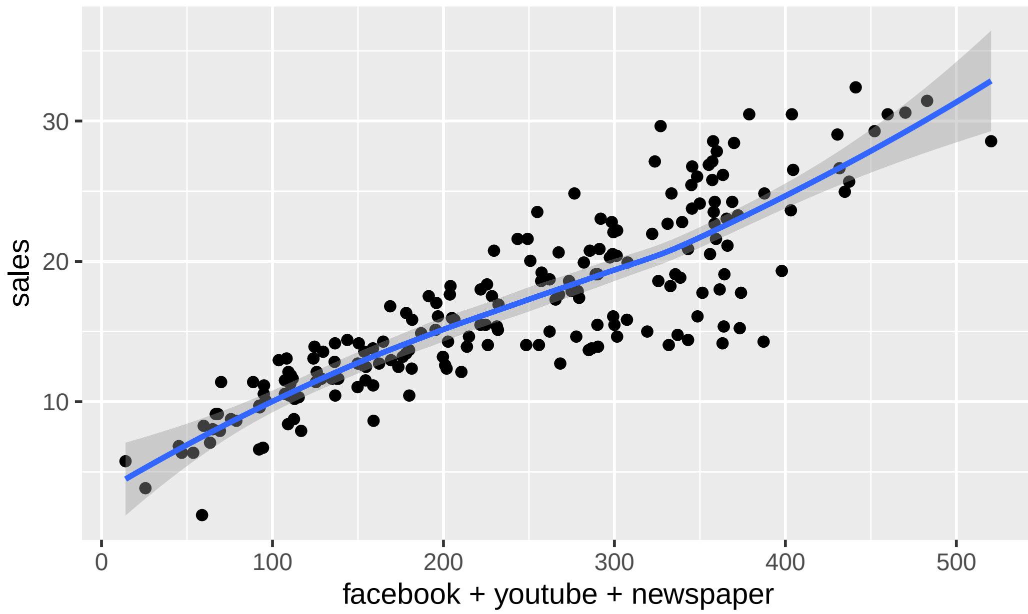
Below is an example

```
multi_data<-data("marketing", package = "datarium")
head(marketing, 4)
```

```
##   youtube facebook newspaper sales
## 1  276.12    45.36     83.04 26.52
## 2   53.40    47.16     54.12 12.48
## 3   20.64    55.08     83.16 11.16
## 4  181.80    49.56     70.20 22.20
```

Visualization

```r
ggplot(marketing, aes(x = facebook+youtube+newspaper, y = sales)) +
  geom_point() +
  stat_smooth()
```




```r
cor(marketing$sales, marketing$facebook)
```

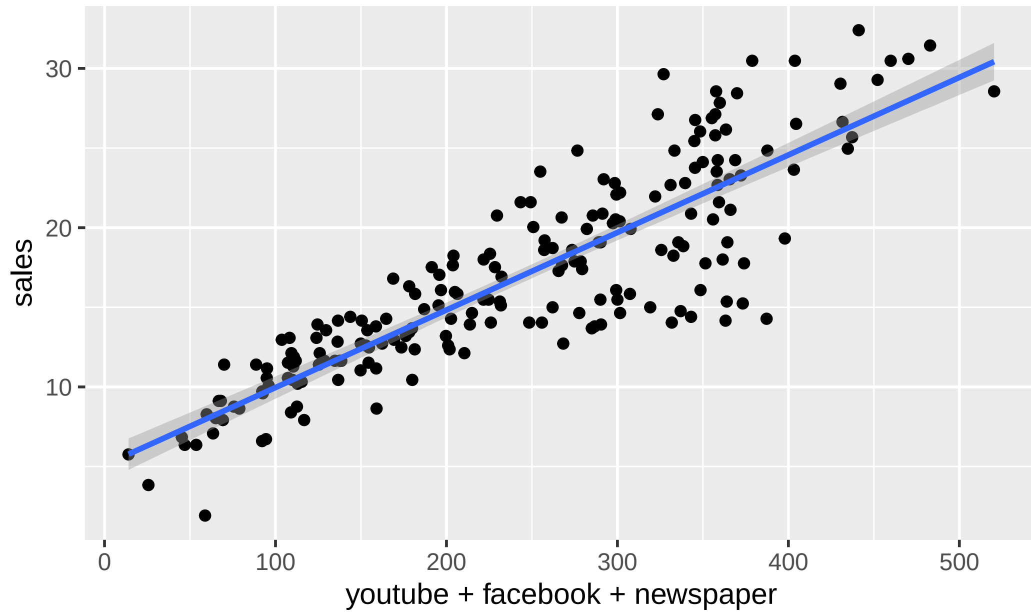
```
## [1] 0.5762226
```

```r
# Building the model
multi_model <- lm(sales ~ youtube + facebook + newspaper, data = marketing)
summary(multi_model)
```

```
##
## Call:
## lm(formula = sales ~ youtube + facebook + newspaper, data = marketing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5932  -1.0690   0.2902   1.4272   3.3951
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.526667   0.374290   9.422   <2e-16 ***
## youtube      0.045765   0.001395  32.809   <2e-16 ***
## facebook     0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.023 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

```r
# Regression line
ggplot(marketing, aes(youtube + facebook + newspaper, sales)) +
```

```
  geom_point() +
  stat_smooth(method = lm)
```



Residual Standard Error (RSE), or sigma

```
sigma(multi_model)/mean(marketing$sales)
```

```
## [1] 0.1202004
```

Conclusion