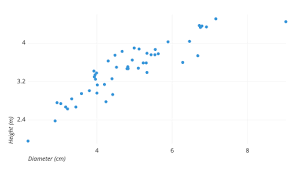
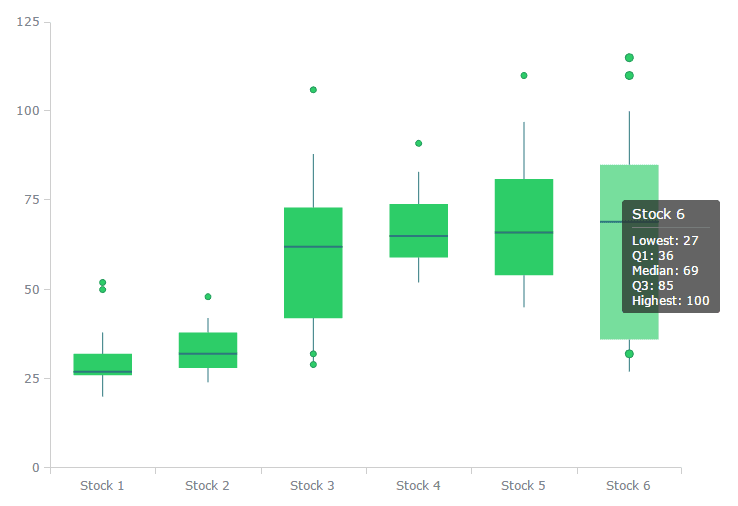
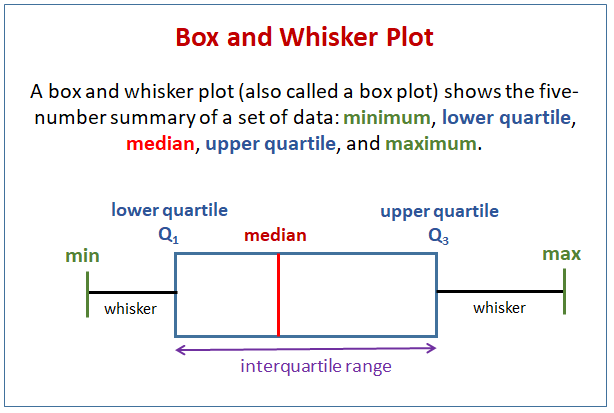
**Graphical Analysis**

**Scatter plot:** Visualise the linear relationship between the predictor and response

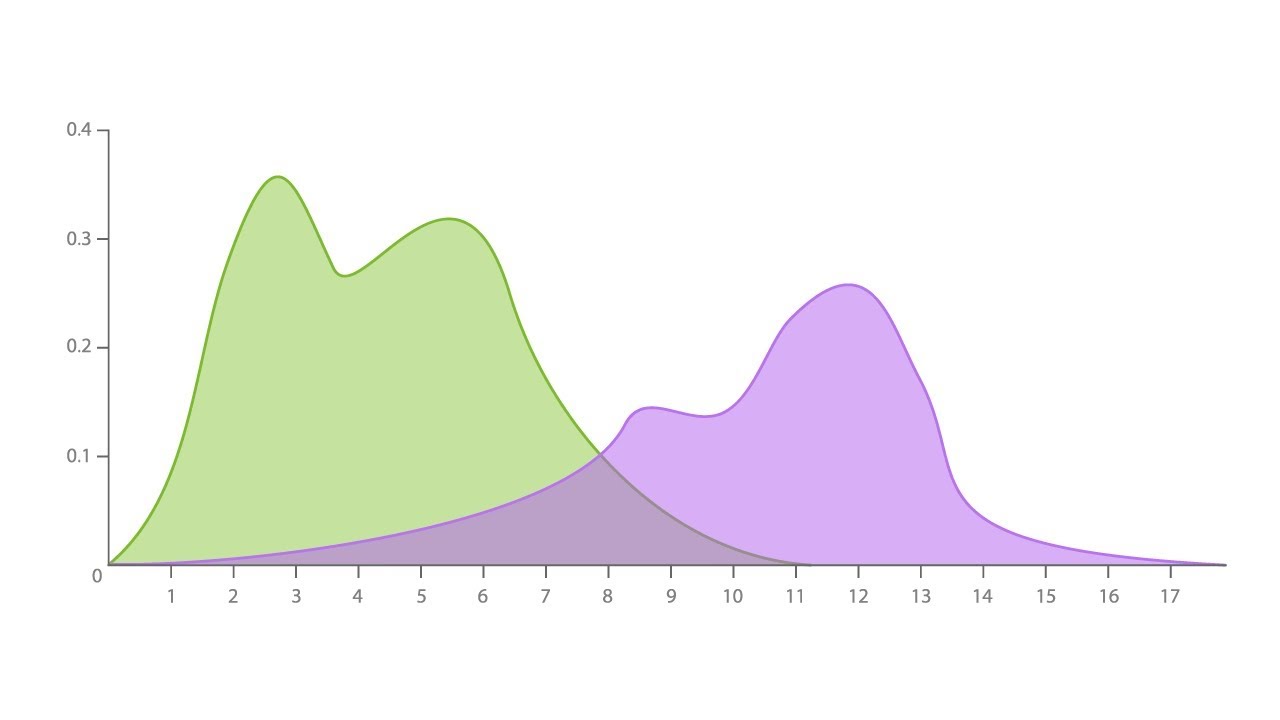
****

**Box plot:** To spot any outlier observations in the variable. Having outliers in your predictor can drastically affect the predictions as they can affect the direction/slope of the line of best fit.





**Density plot:** To see the distribution of the predictor variable. Ideally, a close to normal distribution (a bell shaped curve), without being skewed to the left or right is preferred.



The peaks of a Density Plot help display where values are concentrated over the interval. An advantage Density Plots have over Histograms is that they're better at determining the distribution shape because they're not affected by the number of bins used (each bar used in a typical histogram).

**Linear Regression**

Linear regression is used to predict the value of a continuous variable Y based on one or more input predictor variables X. The aim is to establish a mathematical formula between the the response variable (Y) and the predictor variables (Xs). You can use this formula to predict Y, when only X values are known.

head(cars)

**Using Scatter Plot To Visualise The Relationship**

scatter.smooth(x=cars$speed,y=cars$dist,main="Dist ~ Speed")

## Using BoxPlot To Check For Outliers

## par(mfrow=c(1,2))

## boxplot(cars$speed,main="Speed",sub=paste("OutlierRows",boxplot.stats(cars$speed)$out))

## boxplot(cars$dist,main="Distance",sub=paste("OutlierRows",boxplot.stats(cars$dist)))

## Using Density Plot To Check If Response Variable Is Close To Normal

library(e1071) # for skewness function

par(mfrow=c(1, 2)) # divide graph area in 2 columns

plot(density(cars$speed), main="Density Plot: Speed", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(cars$speed), 2))) # density plot for 'speed'

polygon(density(cars$speed), col="red")

plot(density(cars$dist), main="Density Plot: Distance", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(cars$dist), 2))) # density plot for 'dist'

polygon(density(cars$dist), col="red")

## What is Correlation Analysis?

Correlation analysis studies the strength of relationship between two continuous variables. It involves computing the correlation coefficient between the the two variables.

So what is correlation? And how is it helpful in linear regression?

Correlation is a statistical measure that shows the degree of linear dependence between two variables. Correlation can take values between -1 to +1.

If one variables consistently increases with increasing value of the other, then they have a strong positive correlation (value close to +1).

Similarly, if one consistently decreases when the other increase, they have a strong negative correlation (value close to -1).

A value closer to 0 suggests a weak relationship between the variables.

In other words, if two variables have high correlation, it does not mean one variable ’causes’ the value of the other variable to increase.

Correlation is only an aid to understand the relationship. You can only rely on logic and business reasoning to make that judgement.

cor(cars$speed, cars$dist) # calculate correlation between speed and distance

## Building the Linear Regression Model

## linearMod <- lm(dist ~ speed, data=cars) # build linear regression model on full data

## print(linearMod)

## Linear Regression Diagnostics

## Because, before using a regression model to make predictions, you need to ensure that it is statistically significant.

summary(linearMod)

## Predicting Linear Models

### Create the training and test data

# Create Training and Test data -

set.seed(100) # setting seed to reproduce results of random sampling

trainingRowIndex <- sample(1:nrow(cars), 0.8\*nrow(cars)) # row indices for training data

trainingData <- cars[trainingRowIndex, ] # model training data

testData <- cars[-trainingRowIndex, ]

### **Fit the model on training data and predict dist on test data**

# Build the model on training data

lmMod <- lm(dist ~ speed, data=trainingData) # build the model

distPred <- predict(lmMod, testData) # predict distance

summary (lmMod)

### **Calculate prediction accuracy and error rates**