Lab 2 - Multiple Regression

1/21/2019

Linear regression is one of the most commonly used predictive modelling techniques. The aim of linear regression is to find a mathematical equation for a continuous response variable Y (can be dichotomous) as a function of one or more explanatory X variable(s).

This mathematical equation can be generalised as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

where, β_0 is the intercept and the β_i's are slope coefficients.

Collectively, they are called regression coefficients and ϵ is the error term, the part of Y the regression model is unable to explain.

For this analysis, we will use the `mtcars` dataset that comes with R by default.

You can access this dataset by typing in `mtcars` in your R console. For this exerciese we will want to load it into our environment and save it as a discrete object. We can get a glimpse of the data before loading it in.

```r
head(mtcars)  # display the first 6 observations
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

We can now proceed to load it in:

```r
raw_df <- data.frame(mtcars)
```

The goal here is to establish a mathematical equation for fuel economy as a function of a car's curb weight and its horsepower rating. Deciding what variables are in our equations should be informed by theory. For this example, we want to estimate `mpg` as a function of `wt` and `hp`. Why? Well we expect that a car's curb weight and its horsepower rating (how much power an engine has) impact its fuel economy.

But before we begin building the regression model, it is a good practice to analyse and understand the variables. The graphical analysis and correlation study below will help with this.

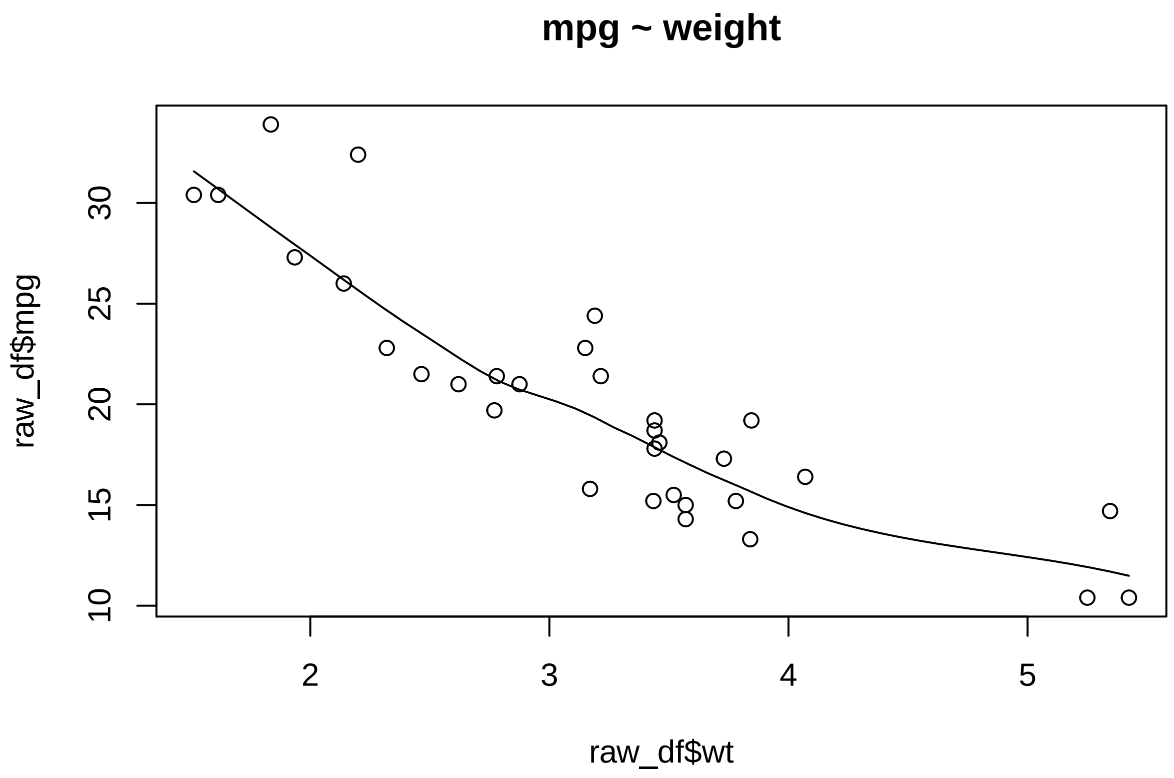
Typically, for each of the predictors, the following plots help visualise the patterns:

- Scatter plot: Visualise the linear relationship (if any) between the predictor and response variable.
- 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.

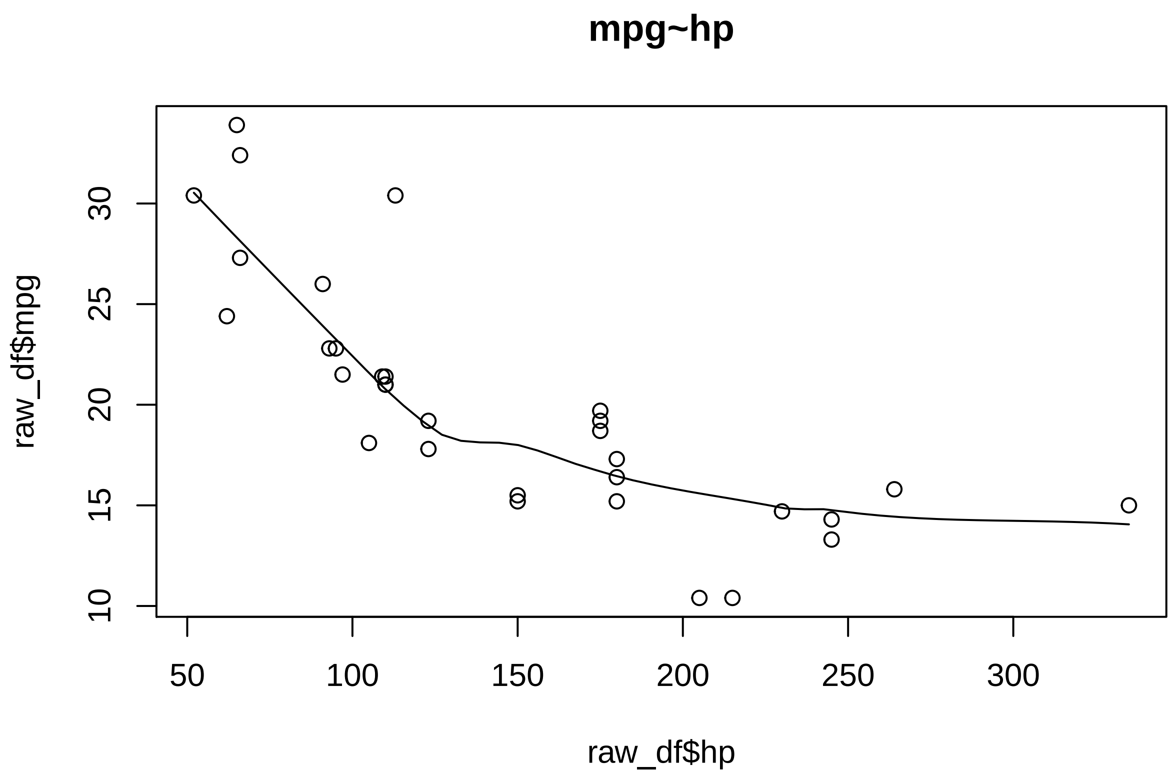
Scatter Plots

Scatter plots can help visualise linear relationships between the response and predictor variables. Ideally, if you have many predictor variables, a scatter plot is drawn for each one of them against the response, along with the line of best fit as seen below.

```r
scatter.smooth(x=raw_df$wt, y=raw_df$mpg, main="mpg ~ weight")  # scatterplot
```



```r
scatter.smooth(x=raw_df$hp, y=raw_df$mpg, main="mpg~hp")
```

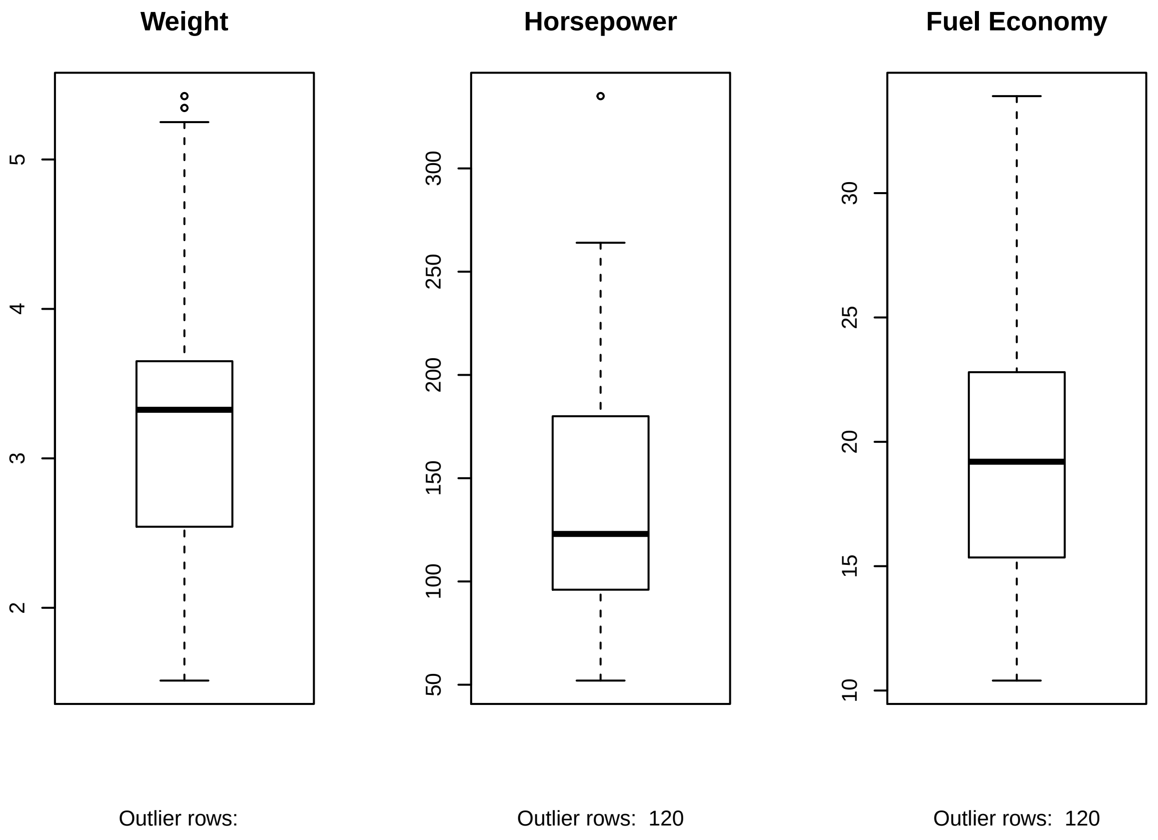


The scatter plot along with the smoothing line above suggests a (mostly) linear and negative relationship between fuel economy and a car's weight, on the one hand, and fuel economy and horsepower, on the other. This is a good thing because one of the underlying assumptions of linear regression is that the relationship between the response and predictor variables is linear and additive.

Box Plot

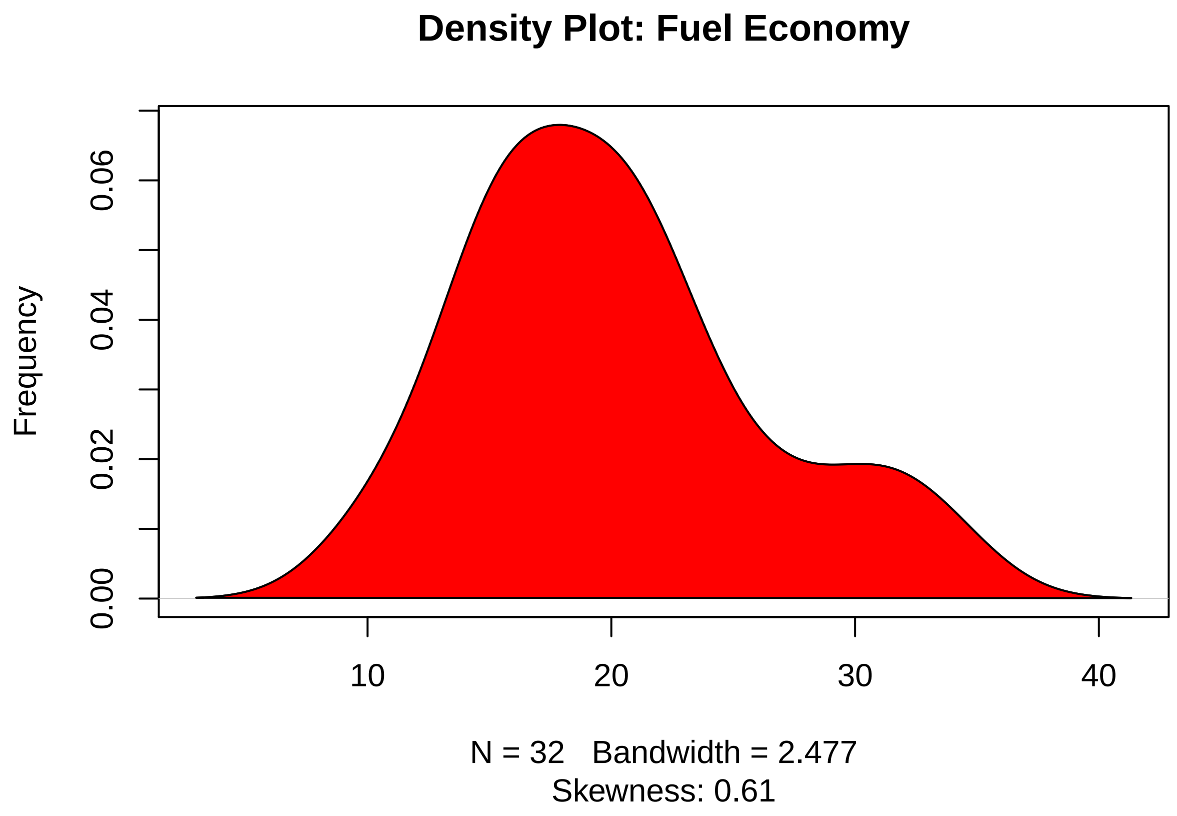
Generally, an outlier is any datapoint that lies outside the 1.5 * inter quartile range (IQR). The IQR is calculated as the distance between the 25th percentile and 75th percentile values for that variable.

```
par(mfrow = c(1, 3))

boxplot(raw_df$wt, main="Weight", sub=paste("Outlier rows: ", boxplot.stats(cars$speed)$out))  # box pl

boxplot(raw_df$hp, main="Horsepower", sub=paste("Outlier rows: ", boxplot.stats(cars$dist)$out))  # box

boxplot(raw_df$mpg, main="Fuel Economy", sub=paste("Outlier rows: ", boxplot.stats(cars$dist)$out))  #
```

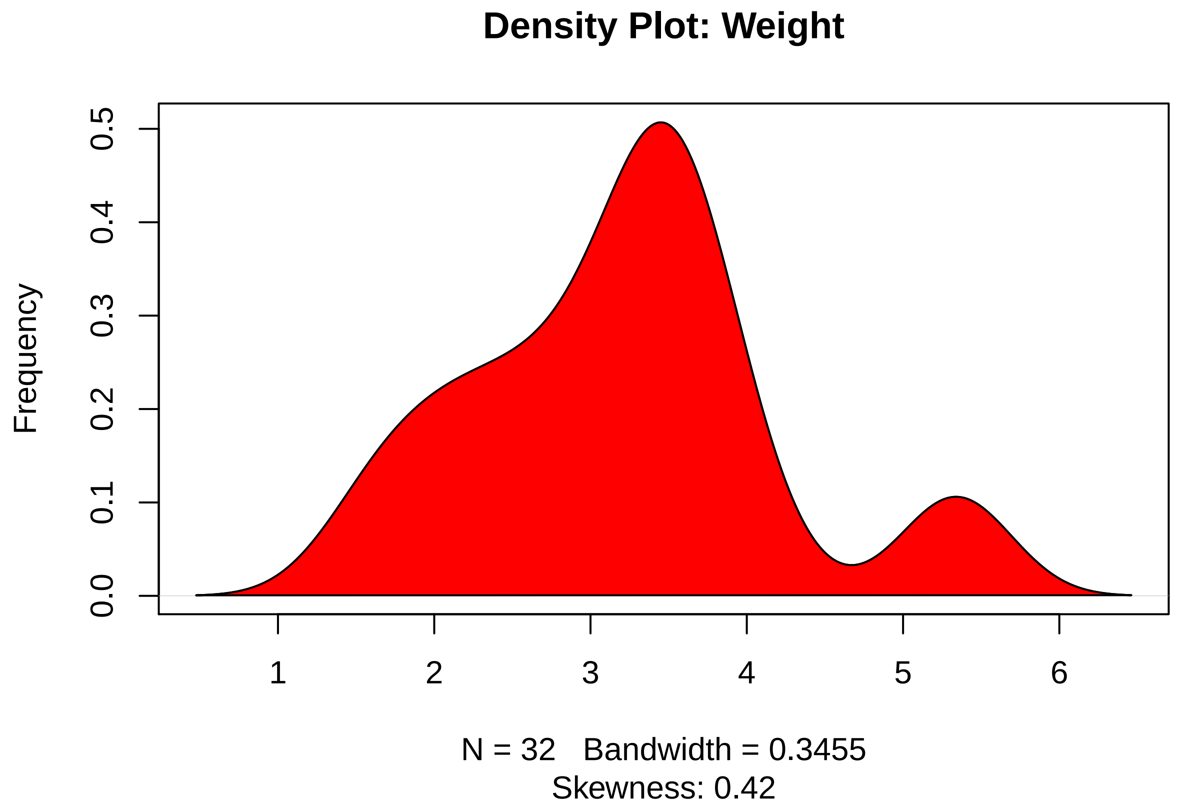


Density Plot

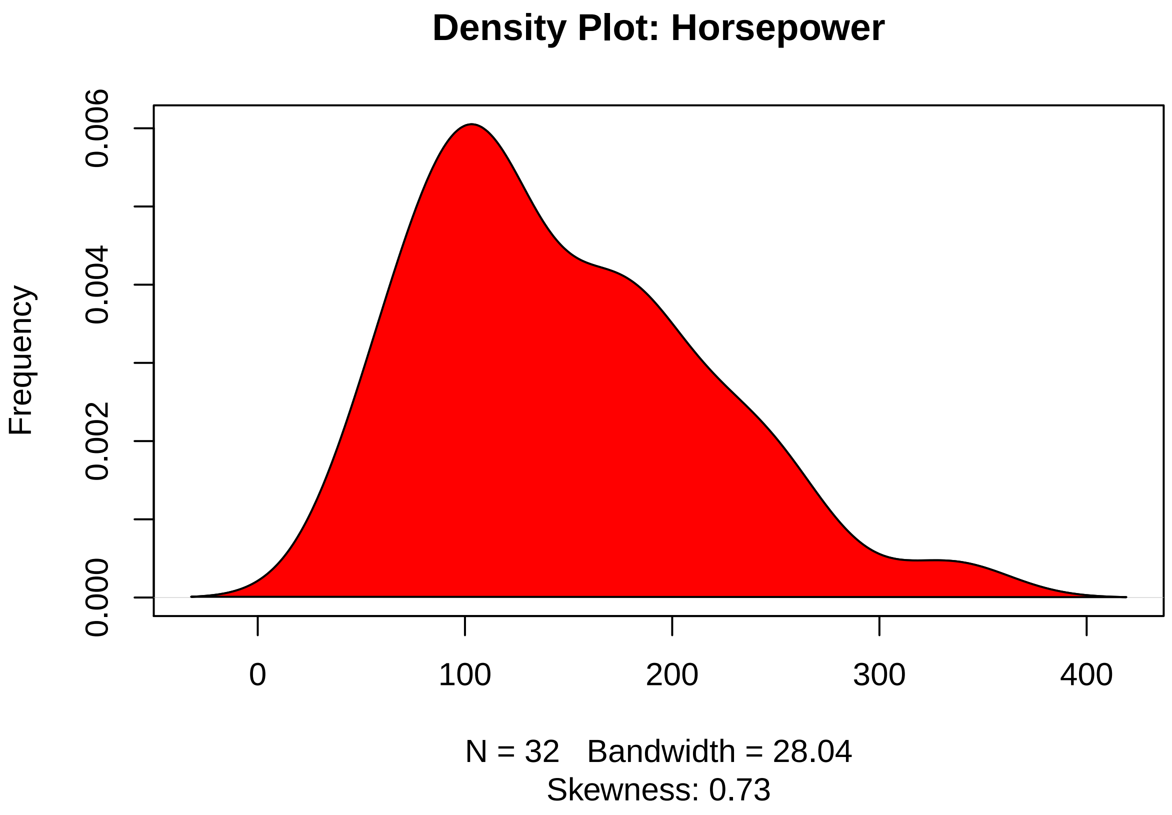
```
library(e1071)  # for skewness function

plot(density(raw_df$mpg), main="Density Plot: Fuel Economy", ylab="Frequency", sub=paste("Skewness:", ro
polygon(density(raw_df$mpg), col="red")
```



```
plot(density(raw_df$wt), main="Density Plot: Weight", ylab="Frequency", sub=paste("Skewness:", round(e1
polygon(density(raw_df$wt), col="red")
```



```r
plot(density(raw_df$hp), main="Density Plot: Horsepower", ylab="Frequency", sub=paste("Skewness:", roun
polygon(density(raw_df$hp), col="red")
```



Correlation Analysis

Correlation analysis studies the strength of the relationship between two continuous variables. More specifically, it is a statistical measure that shows the degree of linear dependence between two variables. Correlation coefficients can take values anywhere between -1 to $+1$.

If one variable consistently increases with increasing values of the other, then they have a strong positive correlation (value close to $+1$). Similarly, if one consistently decreases when the other increases, they have a strong negative correlation (value close to -1). A value closer to 0 suggests a weak relationship between the variables.

> However, correlation does not imply causation.

In other words, if two variables are highly correlated, it does not mean one variable 'causes' the value of the other variable to increase.

To compute in R we simply use the `cor()` function with the two numeric variables as arguments.

```r
cor(raw_df$mpg, raw_df$wt)
```

```
## [1] -0.8676594
```

```r
cor(raw_df$mpg, raw_df$hp)
```

```
## [1] -0.7761684
```

Building the Linear Regression Model

The lm() function takes in two main arguments: `Formula` and `Data`.

The data is typically a `data.frame` object and the formula is an object of class `formula`.

```
linearMod <- lm(mpg ~ wt + hp, data=raw_df)  # build linear regression model on full data
print(linearMod)
```

```
##
## Call:
## lm(formula = mpg ~ wt + hp, data = raw_df)
##
## Coefficients:
## (Intercept)           wt           hp
##    37.22727     -3.87783     -0.03177
```

By building the linear regression model, we have established the relationship between the predictors and response in the form of a mathematical formula. That is, fuel economy (mpg) as a function of curb weight (wt) and horsepower (hp).

For the above output, you can notice the 'Coefficients' column having three components: Intercept: 37.227, weight: -3.878, horsepower: -0.032 In other words,

$$mpg = \beta_0 + \beta_1 wt + \beta_2 hp => mpg = 37.23 - 3.88wt - 0.03hp$$

Linear Regression Diagnostics

Lets begin by printing the summary statistics for linearMod.

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

```
##
## Call:
## lm(formula = mpg ~ wt + hp, data = raw_df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.941 -1.600 -0.182  1.050  5.854
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
## wt          -3.87783    0.63273  -6.129 1.12e-06 ***
## hp          -0.03177    0.00903  -3.519  0.00145 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.593 on 29 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8148
## F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

Calculating the t Statistic and p-Values

When the model coefficients and standard errors are known, the formula for calculating the t-Statistic is as follows:

$$t - Statistic = \frac{\beta - coefficient}{Std.Error}$$

In case you want to compute some of the statistics by manual code, the below snippet shows how.

```
modelSummary <- summary(linearMod)  # capture model summary as an object

modelCoeffs <- modelSummary$coefficients  # model coefficients

beta_wt <- modelCoeffs["wt", "Estimate"] # get beta estimate for weight
se_wt <- modelCoeffs["wt", "Std. Error"]  # get std.error for weight

beta_hp <- modelCoeffs["hp", "Estimate"] # get beta estimate for horsepower
se_hp <- modelCoeffs["hp", "Std. Error"]  # get std.error for horsepower

t_wt <- beta_wt/se_wt  # calc t statistic
t_hp <- beta_hp/se_hp

p_wt <- 2*pt(-abs(t_wt), df=nrow(raw_df)- 1)  # calc p Value
p_hp <- 2*pt(-abs(t_hp), df=nrow(raw_df)- 1)  # calc p Value

f <- summary(linearMod)$fstatistic
f_statistic <- modelSummary$fstatistic[1]  # fstatistic
```

R-Squared and Adj R-Squared

The R-Squared tells us the proportion of variation in the dependent (response) variable that has been explained by our model.

$$R^2 = 1 - \frac{RSS}{TSS}$$

where, RSS is the Residual Sum of Squares given by

$$RSS = \sum_i^n (y_i - \hat{y_i})^2$$

and the Sum of Squared Total is given by

$$TSS = \sum_i^n (y_i - \bar{y})^2$$

Here, y-hat is the fitted value for observation i and y-bar is the mean of Y.

We don't necessarily discard a model based on a low R-Squared value.

As you add more X variables to your model, the R-Squared value of the more complicated model will always be greater than that of the simpler one. Therefore, whatever new variable you add can only add to the variation that was already explained.

Adjusted R-Squared is formulated such that it penalises the number of terms (read predictors) in your model. Therefore, when comparing nested models, it is a good practice to compare using the adj-R-squared rather than just R-squared.

Predicting Linear Models

So far you have seen how to build a linear regression model using the whole dataset. If you build it that way, there is no way to tell how the model will perform with new data.

One way to test how well our model performs is to split our dataset into a 80:20 sample (training:test), then, build the model on the 80% sample and then use the model thus built to predict the dependent variable on test data.

Doing it this way, we will have the model predicted values for the 20% data (test) as well as the actuals (from the original dataset).

By calculating accuracy measures (like min_max accuracy) and error rates (MAPE or MSE), you can find out the prediction accuracy of the model.

Now, lets see how to actually do this.

- Step 1: Create the training and test data This can be done using the sample() function. Just make sure you set the seed using set.seed() so the samples can be recreated for future use.

```
# Create Training and Test data
set.seed(1)  # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(raw_df), 0.8*nrow(raw_df))  # row indices for training data
trainingData <- raw_df[trainingRowIndex, ]  # model training data
testData  <- raw_df[-trainingRowIndex, ]   # test data
```

- Step 2: Fit the model on training data and predict dist on test data

```
# Build the model on training data
lmMod <- lm(mpg ~ wt + hp, data=trainingData)  # build the model
distPred <- predict(lmMod, testData)  # predict distance
```

- Step 3: Review diagnostic measures.

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

```
##
## Call:
## lm(formula = mpg ~ wt + hp, data = trainingData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3522 -1.9910 -0.1166  0.9746  5.3848
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 38.164421   1.957521  19.496 2.27e-15 ***
## wt          -3.980309   0.699258  -5.692 1.00e-05 ***
## hp          -0.032932   0.009688  -3.399  0.00258 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.694 on 22 degrees of freedom
## Multiple R-squared:  0.8292, Adjusted R-squared:  0.8137
## F-statistic: 53.41 on 2 and 22 DF,  p-value: 3.606e-09
```

- Step 4: Calculate prediction accuracy and error rates A simple correlation between the actuals and predicted values can be used as a form of accuracy measure.

A higher correlation accuracy implies that the actuals and predicted values have similar directional movement, i.e. when the actuals values increase the predicted values also increase and vice-versa.

```r
actuals_preds <- data.frame(cbind(actuals=testData$dist, predicteds=distPred))  # make actuals_predicte
correlation_accuracy <- cor(actuals_preds)  # 82.7%
head(actuals_preds)
```

```
##                   predicteds
## Mazda RX4           24.11344
## Merc 280            20.42147
## Honda Civic         30.02373
## Toyota Corona       25.15851
## Dodge Challenger    19.21387
## Fiat X1-9           28.28898
```