1. Task Description

Hello,

Although we're generally happy with using RapidMiner as an analytics tool, many people in the industry seem to be moving to R, an open-source statistical programming language and analytics environment that is supported by a huge community of developers. Based on the excellent analysis work you have done thus far I have decided to ask you to explore introducing R into our current processes. To help you get started, I have obtained a walkthrough tutorial in R for you and a script of code to practice with.

Here are the things I would like you to do:

- Learn what R and RStudio are.
- Install R and RStudio.
- Learn how to work within RStudio.
- Upload a data set into RStudio.
- Install packages into RStudio.
- Call a package in RStudio.
- Perform basic exploratory data analysis.
- Preprocess data.
- Create test and training sets using your data.
- Develop a linear regression model
- Evaluate your model.
- Use your model to make to make predictions.

I have attached the data sets that you'll be using for this task. I'll be expecting a report on your experience in a few days.

Thanks,
Danielle

Danielle Sherman
Chief Technology Officer
Blackwell Electronics
www.blackwellelectronics.com

2. Task Solution

Two data sets (cars.csv and iris.csv) were provided by the CTO, to conduct exploratory analysis and also to execute some predictions. A project named "Task1" was created in R, both data sets were loaded to run two different scripts (Iris_Petal.Width_prediction and cars_predictions) and get some data insights.

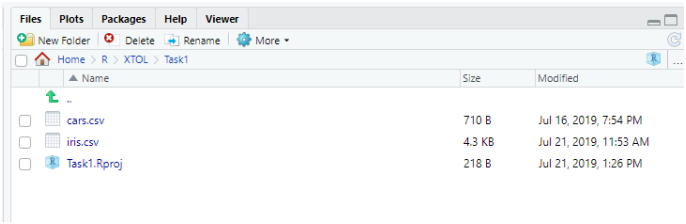

Figure 1 – Datasets loaded into the R project folder (Task1)

2.1 – **Iris Dataset:** For the data Set 1 – Iris.csv, the code below was developed to conduct a prediction of the petal width using petal length data.

```
library(readr)
IrisDataset<- read.csv("iris.csv")
View(IrisDataset)
attributes(IrisDataset)
summary(IrisDataset)
str(IrisDataset)
names(IrisDataset)
hist(IrisDataset$Sepal.Length)
plot(IrisDataset$Sepal.Length, IrisDataset$Sepal.Width)
qqnorm(IrisDataset$Sepal.Lengt)
plot(IrisDataset)
IrisDataset$Species<- as.numeric(IrisDataset$Species)
set.seed(1234)
trainSize <- round(nrow(IrisDataset)*0.8)
trainSize
testSize <- nrow(IrisDataset)-trainSize
testSize
training_indices<-sample(seq_len(nrow(IrisDataset)),size =trainSize)
trainSet <- IrisDataset[training_indices,]
testSet <- IrisDataset[-training_indices,]
LinearModel<- lm(Petal.Width~Petal.Length, trainSet)
summary(LinearModel)
prediction<-predict(LinearModel,testSet)
prediction
plot(prediction)
hist(prediction)
plot(LinearModel)
```

Figure 2 –Iris_Petal.Width_prediction script

The main purpose of this script is import the dataset (iris.csv), conduct data exploration, train a model and conduct the prediction. The model used to conduct the prediction is called "Linearmodel" and the ouputs are listed in figure 3 – LinearModel Outputs.

```
> summary(LinearModel)

Call:
lm(formula = Petal.Width ~ Petal.Length, data = trainSet)

Residuals:
     Min       1Q   Median       3Q      Max
-0.56075 -0.11985 -0.01881  0.13972  0.64663

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.36184    0.04727  -7.654 5.86e-12 ***
Petal.Length  0.41475    0.01117  37.125  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2137 on 118 degrees of freedom
Multiple R-squared:  0.9211,    Adjusted R-squared:  0.9205
F-statistic:  1378 on 1 and 118 DF,  p-value: < 2.2e-16
```

Figure 3 – LinearModel outputs

In the figure above, we draft some conclusions of the model with the following information:

Residuals: The residuals are the difference between the actual observed response values and the response values that the model predicted. The Residuals section of the model output breaks it down into 5 summary points. When assessing how well the model fit the data, you should look for a symmetrical distribution across these points on the mean value zero (0).

In iris example, we can see that the distribution of the residuals do not appear to be strongly symmetrical. That means that the model predicts certain points that fall far away from the actual observed (-0.018)

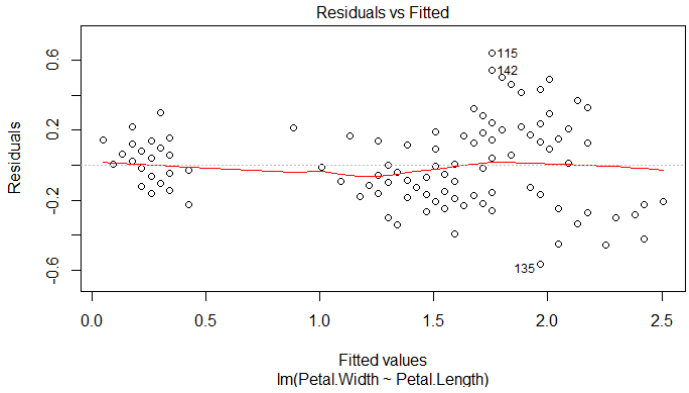

Figure 4 – Residuals vs Fitted

Coefficient - Pr(>t): Relates to the probability of observing any value equal or larger than *t*. A p-value of 5% or less is a good cut-off point. In the iris LinearModel, the p-values are very close to zero. A small p-value for the intercept and the slope indicates that we can reject the null hypothesis which allows us to conclude that there is a relationship between length and width.

Residual Standard Error: It's a measure of the *quality* of a linear regression fit. In the LinearModel example, the actual petal width can deviate from the true regression line by approximately **0.213 units**, on average. In other words, given that the mean width for all petals is **-0.36** and that the Residual Standard Error is **0.213**, we can say that the percentage error is (any prediction would still be off by) **59%**. It's also worth noting that the Residual Standard Error was calculated with 148 degrees of freedom.

Note: Degrees of freedom are the number of data points that went into the estimation of the parameters used after taking into account these parameters (restriction). In our case, we had 118 data points and two parameters (intercept and slope).

Multiple R-squared, Adjusted R-squared: The R-squared statistic provides a measure of how well the model is fitting the actual data. It always lies between 0 and 1 (i.e.: a number near 0 represents a regression that does not explain the variance in the response variable well and a number close to 1 does explain the observed variance in the response variable).

In our case the R-squared is 0.92, which is roughly 92% of the variance found in the response variable (width) can be explained by the predictor variable (Length).

F-Statistic: It's a good indicator of whether there is a relationship between our predictor and the response variables. The further the F-statistic is from 1 the better it is. However, how much larger the F-statistic needs to be depends on both the number of data points and the number of predictors.

Generally, when the number of data points is large, an F-statistic that is only a little bit larger than 1 is already sufficient to reject the null hypothesis (H0 : There is no relationship between length and width). The reverse is true as if the number of data points is small, a large F-statistic is required to be able to ascertain that there may be a relationship between predictor and response variables.

In our example the F-statistic is **1378** which is relatively larger than 1 given the size of our data.

2.2 – **Cars Dataset:** For the dataset 2 – cars.csv, the code below was developed to conduct a prediction of the distances required to stop a car using the speed of the car (Figure 5).

```
library(readr)
carros <- read.csv("cars.csv")
View(carros)
attributes(carros)
summary(carros)
str(carros)
names(carros)
plot(carros)
hist(carros$speed.of.car)
plot(carros$speed.of.car, carros$distance.of.car)
qqnorm(carros$speed.of.car)
qqnorm(carros$distance.of.car)
carros$speed.of.car<- as.numeric(carros$speed.of.car)
names(carros)<- c("model", "speed", "distance")
set.seed(029)
trainSize <- round(nrow(carros)*0.8)
trainSize
testSize <- nrow(carros)-trainSize
testSize
training_indices<-sample(seq_len(nrow(carros)),size =trainSize)
trainSet <- carros[training_indices,]
testSet <- carros[-training_indices,]
Modelolineal<- lm(distance~speed, trainSet)
summary(Modelolineal)
prediction<-predict(Modelolineal,testSet)
prediction
plot(prediction)
hist(prediction)
plot(Modelolineal)
```

Figure 5 – cars_prediction script

The main purpose of this script is import the dataset (cars.csv), conduct data exploration, train a model and conduct the prediction. The model used to conduct the prediction is called “Modelolineal” and the ouputs are listed in figure 6 – Modelolineal Outputs.

```
> summary(Modelolineal)

Call:
lm(formula = distance ~ speed, data = trainSet)

Residuals:
   Min     1Q Median     3Q    Max
-6.672 -3.936 -1.672  4.080 13.017

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.8464     2.3363  -10.63    6e-13 ***
speed         4.3679     0.1507   28.98   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.966 on 38 degrees of freedom
Multiple R-squared:  0.9567,    Adjusted R-squared:  0.9556
F-statistic: 839.9 on 1 and 38 DF,  p-value: < 2.2e-16
```

Figure 6 – Modelolineal output

The results listed in Figure 6, can be interpreted as follows:

Residuals: The residuals are the difference between the actual observed response values and the response values that the model predicted. The Residuals section of the model output breaks it down into 5 summary points. When assessing how well the model fit the data, you should look for a symmetrical distribution across these points on the mean value zero (0).

In iris example, we can see that the distribution of the residuals do not appear to be strongly symmetrical. That means that the model predicts certain points that fall far away from the actual observed (**-1.672**).

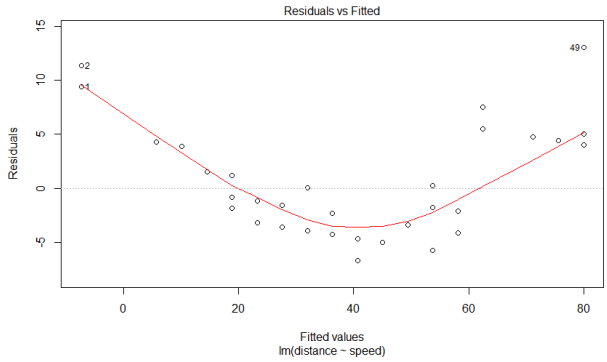

Figure 7 – Residuals vs Fitted

Coefficient - Pr(>t): Relates to the probability of observing any value equal or larger than *t*. A p-value of 5% or less is a good cut-off point. In the iris LinearModel, the p-values are very close to zero. A small p-value for the intercept and the slope indicates that we can reject the null hypothesis which allows us to conclude that there is a relationship between speed and distance.

Residual Standard Error: It's a measure of the *quality* of a linear regression fit. In the LinearModel example, the actual distance required to stop can deviate from the true regression line by approximately **4,96 units**, on average. In other words, given that the mean width for all petals is **24.84** and that the Residual Standard Error is **4.96**, we can say that the percentage error is (any prediction would still be off by) **20%**. It's also worth noting that the Residual Standard Error was calculated with 48 degrees of freedom.

Note: Degrees of freedom are the number of data points that went into the estimation of the parameters used after taking into account these parameters (restriction). In our case, we had 38 data points and two parameters (intercept and slope).

Multiple R-squared, Adjusted R-squared: The R-squared statistic provides a measure of how well the model is fitting the actual data. It always lies between 0 and 1 (i.e.: a number near 0 represents a regression that does not explain the variance in the response variable well and a number close to 1 does explain the observed variance in the response variable).

In our case the R-squared is 0.95, which is roughly 95% of the variance found in the response variable (width) can be explained by the predictor variable (Length).

F-Statistic: It's a good indicator of whether there is a relationship between our predictor and the response variables. The further the F-statistic is from 1 the better it is. However, how much larger the F-statistic needs to be depends on both the number of data points and the number of predictors.

Generally, when the number of data points is large, an F-statistic that is only a little bit larger than 1 is already sufficient to reject the null hypothesis (H0 : There is no relationship between length and width). The reverse is true as if the number of data points is small, a large F-statistic is required to be able to ascertain that there may be a relationship between predictor and response variables.

In our example the F-statistic is **839** which is relatively larger than 1 given the size of our data.

3. Results Discussion

The dataset used in both predictions shows positive results in terms of relationship between petal length and petal width for the iris dataset and speed vs distance for cars dataset. We cannot reject the null hypothesis since p-value is lower than 5% and R-squared is close to 1 in both cases.

4. Recommendations

Both models are not optimized, we can improve the model transforming the response variable using a log function to normalize the data. Compare the difference between the models to increase accuracy.

5. R scripts

Iris Petal Prediction

```
library(readr)
IrisDataset<- read.csv("iris.csv")
View(IrisDataset)
attributes(IrisDataset)
summary(IrisDataset)
str(IrisDataset)
names(IrisDataset)
hist(IrisDataset$Sepal.Length)
plot(IrisDataset$Sepal.Length, IrisDataset$Sepal.Width)
qqnorm(IrisDataset$Sepal.Lengt)
plot(IrisDataset)
IrisDataset$Species<- as.numeric(IrisDataset$Species)
set.seed(1234)
trainSize <- round(nrow(IrisDataset)*0.8)
trainSize
testSize <- nrow(IrisDataset)-trainSize
testSize
training_indices<-sample(seq_len(nrow(IrisDataset)),size =trainSize)
trainSet <- IrisDataset[training_indices,]
testSet <- IrisDataset[-training_indices,]
LinearModel<- lm(Petal.Width~Petal.Length, trainSet)
summary(LinearModel)
prediction<-predict(LinearModel,testSet)
prediction
plot(prediction)
hist(prediction)
plot(LinearModel)
```

Cars Prediction

```
library(readr)
carros <- read.csv("cars.csv")
View(carros)
attributes(carros)
summary(carros)
str(carros)
names(carros)
plot(carros)
hist(carros$speed.of.car)
plot(carros$speed.of.car, carros$distance.of.car)
qqnorm(carros$speed.of.car)
```

```
qqnorm(carros$distance.of.car)
carros$speed.of.car<- as.numeric(carros$speed.of.car)
names(carros)<- c("model", "speed", "distance")
set.seed(029)
trainSize <- round(nrow(carros)*0.8)
trainSize
testSize <- nrow(carros)-trainSize
testSize
training_indices<-sample(seq_len(nrow(carros)),size =trainSize)
trainSet <- carros[training_indices,]
testSet <- carros[-training_indices,]
Modelolineal<- lm(distance~speed, trainSet)
summary(Modelolineal)
prediction<-predict(Modelolineal,testSet)
prediction
plot(prediction)
hist(prediction)
plot(Modelolineal)
```