Bike Renting

Sayan Nag

26 December 2018

Contents

1. Introduction

1.1 Problem Statement

The problem deals with the prediction of bike rental counts based on the environmental and seasonal settings.
We would be analyzing the historic data using tools like R and Python and predicting the trends which have been responsible for affecting the bike renting count.

1.2 Data

Our task is to design regression models which will predict the bike renting based on other factors (variables as per given dataset). Given below is a sample of the data set that we are using to predict the bike renting count:

Table 1.1: Bike Renting Sample Data (Columns: 1-2)

instant	dteday
1	01-01-11
2	02-01-11
3	03-01-11
4	04-01-11
5	05-01-11
6	06-01-11

Table 1.2: Bike Renting Sample Data (Columns: 3-7)

season	yr	mnth	holiday	weekday
1	0	1	0	6
1	0	1	0	0
1	0	1	0	1
1	0	1	0	2
1	0	1	0	3
1	0	1	0	4

Table 1.3: Bike Renting Sample Data (Columns: 8-12)

workingday	weathersit	temp	atemp	hum
0	2	0.344167	0.363625	0.805833
0	2	0.363478	0.353739	0.696087
1	1	0.196364	0.189405	0.437273
1	1	0.2	0.212122	0.590435
1	1	0.226957	0.22927	0.436957
1	1	0.204348	0.233209	0.518261

Table 1.4: Bike Renting Sample Data (Columns: 13-14)

windspeed	casual	registered	cnt
0.160446	331	654	985
0.248539	131	670	801
0.248309	120	1229	1349
0.160296	108	1454	1562
0.1869	82	1518	1600
0.089565	88	1518	1606

There are 13 independent variables as per the given dataset as follows:

1. **instant**
2. **dteday**
3. **season**
4. **yr**
5. **mnth**
6. **holiday**
7. **weekday**
8. **workingday**
9. **weathersit**
10. **temp**
11. **atemp**
12. **hum**
13. **windspeed**

The dependent variables are **casual**, **registered** and **cnt**, which are all continuous in nature wherein **cnt** is the sum total of **casual** and **registered**. We need to predict the variable **cnt**, wherein **cnt** is dependent on the values of **casual** and **registered**.

Out of the 13 independent variables, following are the continuous and categorical variables:

Table 1.5: Continuous and Categorical independent variables

Continuous variables	**Categorical variables**
instant	dteday
temp	season
atemp	yr
windspeed	holiday
hum	weekday
	mnth
	workingday
	weathersit

2. Methodology

2.1 Data Pre-Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**.

We will be executing various pre-processing techniques such as **missing value imputation**, **outlier analysis**, **feature selection** and **feature sampling** in order to clean the data and make it suitable for becoming the input to our model. More often than not, data pre-processing can be challenging since this is the most important and time-consuming part of data analysis and modeling – and any small mistake can even get critical, resulting in an underperforming model.

2.1.1 Missing Value Analysis and Imputation

Mostly in real-time data, there are some observations which will be blank or missing. These missing values tend to make the data more difficult to analyze since the users find it difficult to apply statistical techniques like mean, mode, median, correlation, etc. on variables having missing data. Moreover, it is also not advisable to feed an input dataset with missing values to a model, since that may cause high anomalies in the analysis, which may cause wrong predictions altogether.
In our dataset, we **do not have any variable with missing values**. Hence missing value analysis is not necessary in this case.

Code snippet:

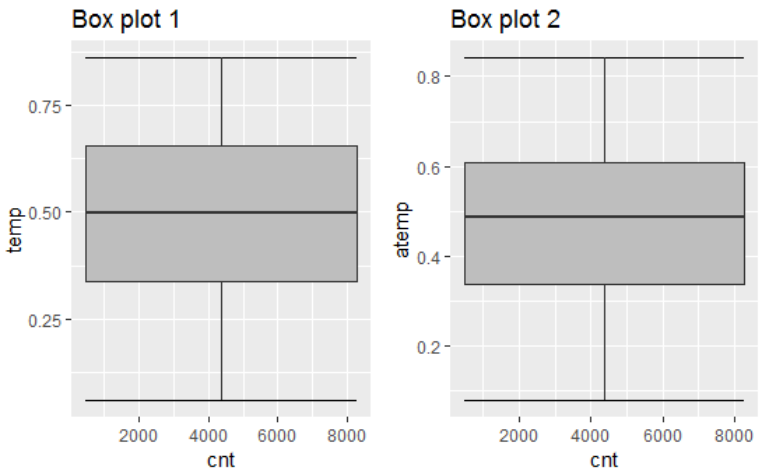
```
#reading CSV data sheet

data = read.csv("Bike_Rental_Count_data.csv", header = T, sep = ",")

#creating reference object with same data

data_1 = data

#Missing value analysis

sum(is.na(data_1))

>0
```

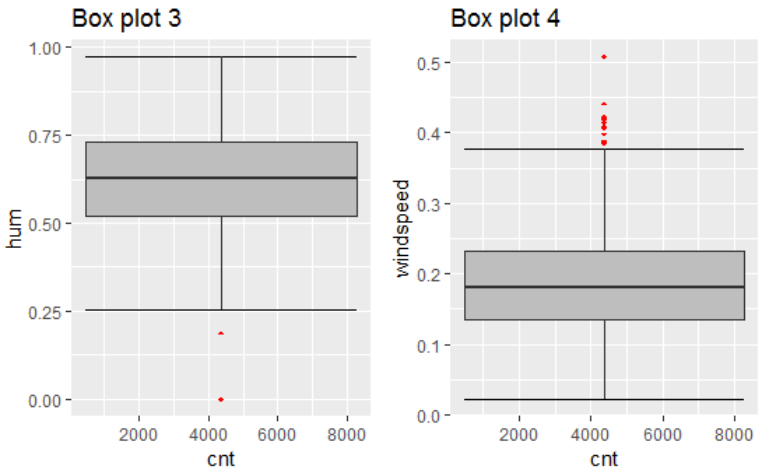
2.1.2 Outlier analysis

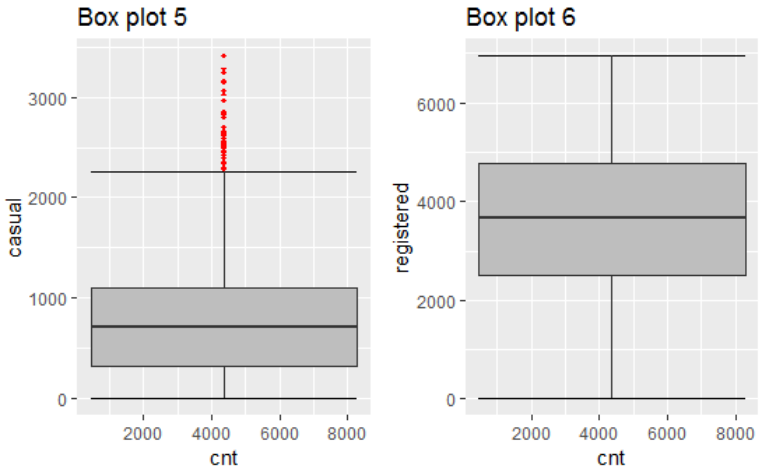
This is a technique for ensuring that the **continuous** variables in the data do not contain values which vary largely from the other sets of values. Whenever we are attempting to design a model, the input data should be devoid of outliers so that the output predicted by the model does not get negatively affected by the outliers.
Following are the code-snippets for outlier analysis:

```
#Outlier analysis in un-normalized independent numeric variables
library(ggplot2)
numerics = sapply(data_1, is.numeric)
numeric_cols = data_1[,numerics]
cnames = c("temp","atemp","hum","windspeed","casual","registered")

for (i in 1:length(cnames))
{
  assign(paste0("oa",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"),
                                data = subset(data))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=cnames[i],x="cnt")+
           ggtitle(paste("Box plot",i)))
}

# Plotting box plots to show outlier values
gridExtra::grid.arrange(oa1,oa2,ncol=2)
gridExtra::grid.arrange(oa3,oa4,ncol=2)
gridExtra::grid.arrange(oa5,oa6,ncol=2)
```

In our case, we have performed a graphical analysis using **box-plots** to check the outliers for all the continuous variables, shown as below:







After performing graphical analysis of outliers, we have replaced the outliers with zeroes (0s) in the dataset and then replaced those zeroes with median (for continuous variables) and mode (for categorical variables), exactly in the same way that we have done for missing value imputation.
Following is the code snippet for replacing the outliers with zeroes in the dataset:

Code-snippet:

```
#Replace all outliers with 0 and impute
for(i in cnames){
  #print(i)
  val = data_1[,i][data_1[,i] %in% boxplot.stats(data_1[,i])$out]
  #print(length(val))
  data_1[,i][data_1[,i] %in% val] = NA
}

#replacing NA values in variable "casual" with median
data_1$casual[is.na(data_1$casual)] = median(data_1$casual, na.rm = TRUE)

for (i in 1:length(data_1$instant))
{
  data_1$cnt[i] = data_1$casual[i] + data_1$registered[i]
}
```

#replacing NA values of "hum" and "windspeed" with median
data_1$hum[is.na(data_1$hum)] = median(data_1$hum, na.rm = TRUE)
data_1$windspeed[is.na(data_1$windspeed)] = median(data_1$windspeed, na.rm = TRUE)

2.1.3 Feature Selection

Post performing missing value and outlier analyses, we need to perform feature selection. Often, the dataset provided by the client/company contains many unnecessary or unimportant variables which do not impact the target/dependent variable. If we keep these variables and pass the input to our model, it causes unnecessary overheads with respect to memory consumption and also model performance. That is why it is absolutely necessary to perform feature selection and determine which independent variables actually impact the target variable and accordingly, we may design the input for the model.

In this project, the following feature selection techniques have been utilized:

a) **Correlation analysis**: This is a test performed essentially only on **continuous numeric** variables and is used to visualize the correlation co-efficients of each independent numeric variable among themselves and also with respect to the dependent or target variable. This test helps one to determine which variables to keep in the dataset and which ones to drop. It is suggested that numeric variables with zero correlation co-efficients with respect to target variable need to be dropped as there is no dependency between those variables and the target variable.

Following is the code snippet:

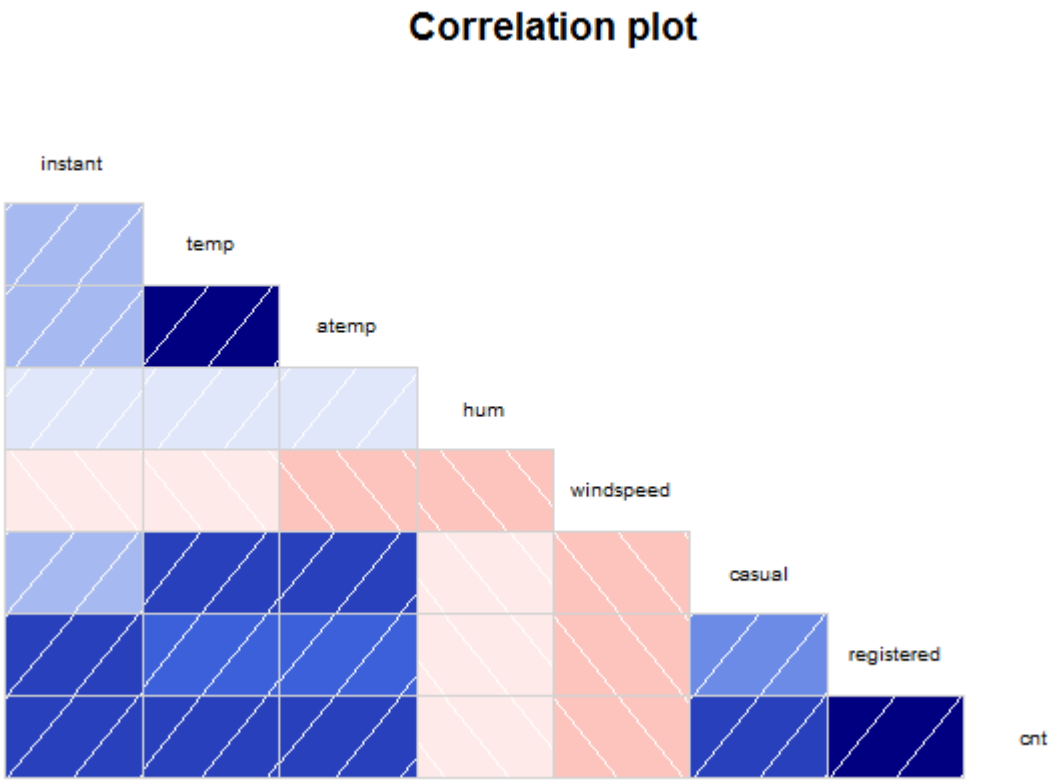
#correlation test on numeric variables
value = round(cor(as.matrix(data_1[,numerics])),2)

library(corrgram)
corrgram(data_1[,numerics], order=FALSE, lower.panel=panel.shade,
upper.panel=NULL, text.panel=panel.txt,
main="Correlation plot"

Following is the summary of the correlation test on the given dataset:

```
> value
            instant  temp atemp   hum windspeed casual registered   cnt
instant        1.00  0.15  0.15  0.00     -0.11   0.22       0.66  0.62
temp           0.15  1.00  0.99  0.12     -0.14   0.58       0.54  0.62
atemp          0.15  0.99  1.00  0.14     -0.16   0.58       0.54  0.62
hum            0.00  0.12  0.14  1.00     -0.20  -0.08      -0.11 -0.12
windspeed     -0.11 -0.14 -0.16 -0.20      1.00  -0.18      -0.20 -0.22
casual         0.22  0.58  0.58 -0.08     -0.18   1.00       0.42  0.64
registered     0.66  0.54  0.54 -0.11     -0.20   0.42       1.00  0.97
cnt            0.62  0.62  0.62 -0.12     -0.22   0.64       0.97  1.00
```

Following is the visualization plot of the correlation test:



In the above correlation plot, deep red signifies high negative correlation and deep blue signifies high positive correlation.

From the given dataset, after performing correlation test, we find that the dependent variable (**cnt**) is having high positive correlation with **temp**, **atemp**, **casual** and **registered**. Moreover, it is having negative correlation with **hum** and **windspeed**. Also a careful investigation reveals that the level of correlation among **instant**, **temp** and **atemp** with **cnt** is exactly the same (correlation co-efficient is 0.62). So, to avoid multicollinearity, we have dropped the variables **instant** and **atemp**,

b) **ANOVA (Analysis of Variance) test**: This test is useful to determine the dependence of categorical independent variables and continuous dependent variable. It uses two hypothesis principles, viz., **null** hypothesis and **alternate** hypothesis.

The null hypothesis assumes that the selected variables are independent of each other whereas the alternate hypothesis assumes that the selected variables under consideration are not independent of each other. It basically calculates the mean of each variable in the entire population with the assumption that if the individual means are equal then the null hypothesis is true (variables are independent of each other). On the other hand, a difference of mean of any variable implies failure of the null hypothesis and acceptance of the alternate hypothesis (variables have at least some dependence with each other). It is guided based on a parameter known as the probability factor or more popularly, as the p-value.

If the p-value is lesser than 0.05, then the null hypothesis is rejected with the assumption that the independent variable under consideration is having dependency with the target variable, i.e., the variable carries information to describe the target variable.

In the given dataset, ANOVA test has been performed to select the categorical variables and following is the result:

```
> summary(anova_result)
             Df    Sum Sq   Mean Sq F value   Pr(>F)
season        3 849038103 283012701  428.97  < 2e-16 ***
yr            1 745881709 745881709 1130.56  < 2e-16 ***
mnth         11 159221858  14474714   21.94  < 2e-16 ***
holiday       1   7264021   7264021   11.01 0.000953 ***
weekday       6  47807030   7967838   12.08 6.04e-13 ***
weathersit    2 174459453  87229726  132.22  < 2e-16 ***
Residuals   706 465780872    659746
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Following is the code snippet of the ANOVA test:

#anova test on continuous target/dependent variable and categorical independent variables
anova_result = aov(cnt~season+yr+mnth+holiday+weekday+workingday+weathersit, data=data_1)
summary(anova_result)

Since the p-value for all the variables in the ANOVA test are way lesser than 0.05, we reject the null hypothesis with the expectancy that variables are dependent of each other. Thus, **none of the categorical variables have been dropped**.

Also, date handling on the variable **dteday** has been done as per the below code snippet:

#extracting only days from "dteday variable
data_1$dteday = weekdays(as.Date(data_1$dteday))

After converting **dteday** into weekdays, we see that **dteday** and **weekday** are redundant in nature, both carrying the same information. So, we have dropped one of those variables (**dteday**).

Thus, post feature selection, the following 3 independent variables are dropped from the original dataset:
1) instant
2) atemp
3) dteday

Thus, the dataset now contains (16-3) = 13 variables of which, 10 are independent variables and the 11th, 12th and 13th variables are the target/dependent variables. As previously mentioned, we need to predict the 13th variable (**cnt**).

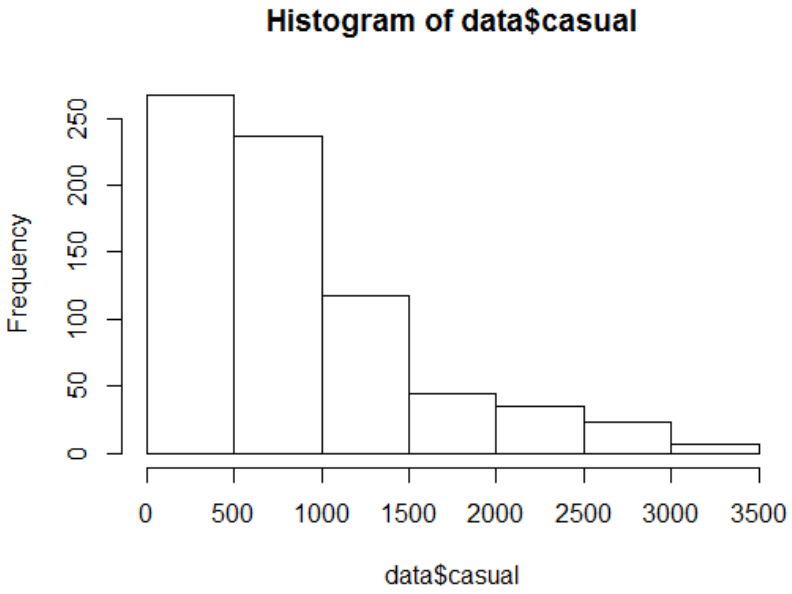
2.1.4 Feature Sampling

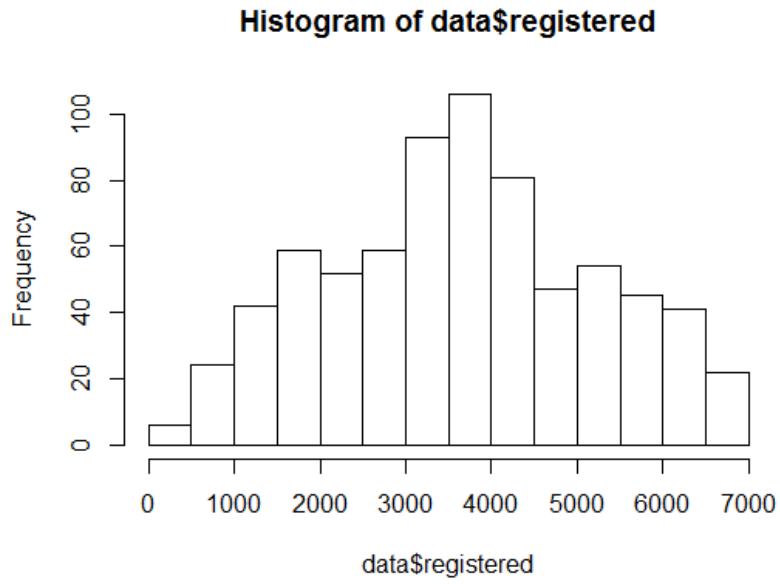
It is a technique which modifies the range of the variables in the dataset as per some pre-defined limits. Many a times one or more variables may vary between a huge range. If those variables are not sampled, it may cause biasing effects in the model wherein, the model performance will be more inclined towards the high-ranged variable and therefore, the effect of the other variable/s may be reduced or shielded.

In order to avoid this situation, the data is being sampled. In this project, **normalization** has been used as the sampling technique, wherein the continuous variables have been normalized in a range of minimum 0 to maximum 1.

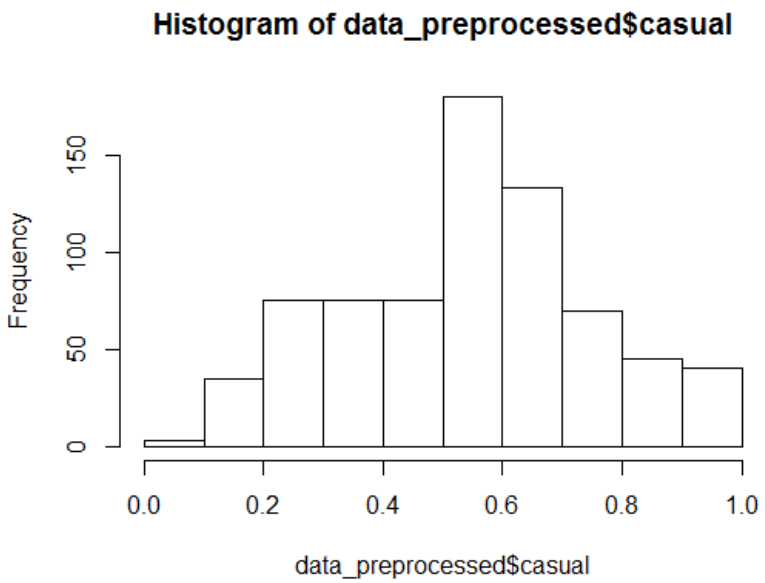
In the given dataset, we see that the variables **temp**, **hum** and **windspeed** are already provided in normalized format. Following are the distribution plots (histogram) of the continuous variables **casual** and **registered** before normalization:

Distributions before normalization:





We see that the variable **casual** is visibly right-skewed and hence, we need to process it in order to make it as much close to normal as possible. We have taken the square-root of the **casual** variable and after that, following is the distribution, which is much close to a normal distribution:



Following is the **summary** of the data post applying the three data pre-processing techniques, as explained above:

```
> summary(data_preprocessed)
     season          yr            mnth          weekday        holiday
 Min.   :1.000   Min.   :1.000   Min.   : 1.00   Min.   :1.000   Min.   :1.000
 1st Qu.:2.000   1st Qu.:1.000   1st Qu.: 4.00   1st Qu.:2.000   1st Qu.:1.000
 Median :3.000   Median :2.000   Median : 7.00   Median :4.000   Median :1.000
 Mean   :2.497   Mean   :1.501   Mean   : 6.52   Mean   :3.997   Mean   :1.029
 3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:10.00   3rd Qu.:6.000   3rd Qu.:1.000
 Max.   :4.000   Max.   :2.000   Max.   :12.00   Max.   :7.000   Max.   :2.000
   workingday      weathersit         temp              hum          windspeed
 Min.   :1.000   Min.   :1.000   Min.   :0.05913   Min.   :0.2542   Min.   :0.02239
 1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.33708   1st Qu.:0.5223   1st Qu.:0.13495
 Median :2.000   Median :1.000   Median :0.49833   Median :0.6275   Median :0.17880
 Mean   :1.684   Mean   :1.395   Mean   :0.49538   Mean   :0.6294   Mean   :0.18626
 3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:0.65542   3rd Qu.:0.7302   3rd Qu.:0.22979
 Max.   :2.000   Max.   :3.000   Max.   :0.86167   Max.   :0.9725   Max.   :0.37811
     casual        registered          cnt
 Min.   :0.0309   Min.   :0.003293   Min.   :0.0342
 1st Qu.:0.3911   1st Qu.:0.367760   1st Qu.:0.8230
 Median :0.5527   Median :0.534010   Median :1.1299
 Mean   :0.5473   Mean   :0.542800   Mean   :1.0901
 3rd Qu.:0.6812   3rd Qu.:0.710127   3rd Qu.:1.3964
 Max.   :1.0000   Max.   :1.000000   Max.   :2.0000
>
```

2.2 Modeling

2.2.1 Model Selection

In this case, we see that the dependent variable **cnt** is continuous in nature. Thus, it falls under a **regression** problem wherein we need to select a model suitable for regression.
We will be checking out with the Random Forest regression, K Nearest Neighbor and Linear Regression models.

2.2.2 Cross Validation

It is a technique used to predict the efficiency of a model under various training datasets. It is performed to ensure that the model designed is not overfitting and can be acceptable in production. There are mainly two types of cross-validation techniques:

1) **K-Fold Cross-Validation** – In this technique, the entire dataset is divided into "K" equal or almost equal sets. Then, one of the sets is kept aside as the testing dataset and the remaining (K-1) sets are considered as the training datasets. The modeling technique is then applied to each of the (K-1) datasets together and then, the model fit is checked on the training sets. The observation which gives the least error metrics becomes the acceptable training dataset for the test dataset. After that, we predict the test dataset observations using the best fit training model, as determined from the cross-validation technique. The resultant accuracy obtained becomes the maximum attainable accuracy for the model.
For larger datasets, the value of K should be ideally around 5 to 10. Higher the value of K, more is the computation time involved due to higher number of iterations and a low value of K is supposed to make the model vulnerable to noise.

2) **LOOCV (Leave One Out Cross Validation)** – In this technique, a single observation is taken from the dataset and that becomes the testing data whereas the remaining observations are the training sets. This process is repeated such that each observation in the dataset population/sample is used once as the test dataset. It is similar to the K-Fold Cross-Validation wherein the value of K is equal to the number of observations in the population/sample.
It is computationally expensive due to the amount of time required in training the data.

We have performed K-Fold Cross-Validation in our dataset and the code for the same is as follows:

```
#perform k-fold cross-validation to handle overfitting
library(caret)
library(caTools)
index_cv = createDataPartition(data_preprocessed$cnt, p=0.80, list = FALSE)
train_cv = data_preprocessed[index_cv,]
test_cv = data_preprocessed[-index_cv,]
control_parameters = trainControl(method = "cv", number = 10,
                          savePredictions = TRUE, classProbs = TRUE)

#cross-validated for KNN
model_cv_knn = train(cnt~., data = train_cv, method = "knn",
                trControl = control_parameters)       # optimal k = 5, MAE=12.84%,
RMSE=18.56%, Rsquared=76.96%

#cross-validated model for Linear Regression
model_cv_lm = train(cnt~., data = train_cv, method = "lm",
                trControl = control_parameters)       # MAE=0%, RMSE=0%, Rsquared=100%
```

#cross-validated model for Random Forest
model_cv_rf = train(cnt~., data = train_cv, method = "rf",
trControl = control_parameters) # ***optimal mtry = 12, MAE=1.61%, RMSE=2.43%, Rsquared=99.63%***

#predict values for KNN, for example
predictions_cv = predict(model_cv_knn,test_cv)

#create confusion matrix for continuous target variable by using table function
conf_matrix = table(predictions = predictions_cv,actual = test_cv$cnt)

2.2.3 Random Forest Regression Model

Following is the model implementation:

```
#Random Forest Regression Modeling
library(randomForest)
model_rf = randomForest(cnt~., data=train_dataset, mtry=12, importance=TRUE, type="anova")

#No. of variables tried at each split = 12
#No. of trees = 500 (by default)

#prediction on train data
pred_rf_train = predict(model_rf, train_dataset)
table_pred_rf_train = data.frame(pred_rf_train,train_dataset$cnt)

#prediction on test data
pred_rf_test = predict(model_rf, test_dataset)
table_pred_rf_test = data.frame(pred_rf_test,test_dataset$cnt)

#calculate model accuracy for Random Forest

#MAE
mean(abs(pred_rf_test - test_dataset$cnt))  #1.37%

#RMSE
sqrt(mean((pred_rf_test - test_dataset$cnt)^2))  #2.14%

#Accuracy = 100 - 1.37 = 98.63%
```

We have fed 80% of the observations in the training dataset and 20% observations in the testing dataset.

2.2.4 Random Forest Regression Model Performance

We see that the Mean Absolute Error (MAE) is 1.37% and Root Mean Square Error (2.14%) result in an effective model accuracy of approximately **98.63**%. Since this accuracy has been attained post following K-Fold cross validation (**considering K = 10**), we can assume that the model won't overfit in real-time.

2.2.5 K Nearest Neighbor Regression Model

This modeling technique requires all the variables under consideration to be of numeric data type. Based on the selected value of “K”, it uses a distance technique (Euclidean distance) in order to predict the variable. A small value of K” can make the model vulnerable to noise whereas a high value of “K” can result in the model bypassing important insights about the observations while predicting. Also, a high value of “K” is computationally expensive. The KNN technique is not advisable to be adopted for large datasets. Post cross-validation, we have found the optimal value of “K” for the dataset to be 5.

Following is the implementation of the model:

```
#K Nearest Neighbor regression modeling
library(FNN)
cnt_train = train_dataset$cnt
cnt_test = test_dataset$cnt
predictions_knn = knn.reg(train_dataset,test_dataset,cnt_train,k=5)

#calculate model accuracy for KNN
#Mean Absolute Error
mean(abs(cnt_test-predictions_knn$pred))  #10.65%

#MAPE
mean(abs((cnt_test-predictions_knn$pred)/(cnt_test)))*100  #22.08%

#Mean Squared Error
mean((cnt_test-predictions_knn$pred)^2)  #2.27%

#Accuracy = 100 - 10.65 = 89.35%
```

2.2.6 K Nearest Neighbor Regression Model Performance

We see that the Mean Absolute Error (MAE) is 10.65%, Mean Square Error (2.27%) and MAPE (Mean Absolute Percentage Error) of 22.08% result in an effective model accuracy of approximately **89.35**%. Since this accuracy has been attained post following K-Fold cross validation (**considering K = 10**), we can assume that the model won’t overfit in real-time.
However, this model does not perform as good as the Random Forest Regression model.

2.2.7 Linear Regression Model

It also intakes all numerical variables and then predicts the fitness of the target/dependent variable based on the variations of the respective input variables.
Following is the implementation of the model:

```
#linear regression modeling
library(usdm)
vif(data_preprocessed[,-13])
vifcor(data_preprocessed[,-13], th=0.9)

#checking linear relation for "casual", "registered" and "cnt" to check impact on
#"casual" and "registered" variables, which in turn determine the total count "cnt".
lm_model_cas = lm(casual~.,data = train_dataset)
summary(lm_model_cas)

lm_model_reg = lm(registered~.,data = train_dataset)
summary(lm_model_reg)

lm_model = lm(cnt~.,data = train_dataset)
summary(lm_model)
predictions_lm = predict(lm_model, test_dataset[1:12])

#calculate model accuracy for linear regression
library(DMwR)
regr.eval(test_dataset[,13], predictions_lm, stats = c("mae","rmse","mape","mse"))

#Residual Standard Error
sigma(lm_model)/mean(test_dataset$cnt)  #0%

#Confidence Interval
confint(lm_model)

#Accuracy = 100%
```

2.2.8 Linear Regression Model Performance

We see that after cross-validation, this model perfectly fits the testing/validation dataset in the training dataset. It virtually results in a model accuracy of 100% with an R-squared value of 1(100%). This may seem like the model is going to overfit in real-time however, we have cross validated the dataset beforehand to handle overfitting and hence we should be good to go with the 100% accuracy in this case.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have three models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Bike Renting data, the latter two, *Interpretability* and *Computation Efficiency*, do not hold much significance. Therefore we will use *Predictive performance* as the criteria to compare and evaluate models. Predictive performance can be measured by comparing the predictions of the models with real values of the target variable and calculating some average error measure.

3.1.1 Mean Absolute Error (MAE), Mean Square Error (MSE), Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE)

MAE, MSE, MAPE and RMSE are the important error measures used to calculate the predictive performance of the model. We will apply these measures to our models that we have generated in the previous section.

1) **Random Forest Regression**:

```
#calculate model accuracy for Random Forest
#MAE
mean(abs(pred_rf_test - test_dataset$cnt))  #1.37%
#RMSE
sqrt(mean((pred_rf_test - test_dataset$cnt)^2))  #2.14%
#Accuracy = 100 - 1.37 = 98.63%
```

2) **K Nearest Neighbor Regression**:

```
#calculate model accuracy for KNN
#Mean Absolute Error
mean(abs(cnt_test-predictions_knn$pred))  #10.65%
#MAPE
mean(abs((cnt_test-predictions_knn$pred)/(cnt_test)))*100  #22.08%
#Mean Squared Error
mean((cnt_test-predictions_knn$pred)^2)  #2.27%
#Accuracy = 100 - 10.65 = 89.35%
```

3) **Linear Regression**:

```
#calculate model accuracy for linear regression
library(DMwR)
regr.eval(test_dataset[,13], predictions_lm, stats = c("mae","rmse","mape","mse"))
#Residual Standard Error
sigma(lm_model)/mean(test_dataset$cnt)  #0%
#Confidence Interval
confint(lm_model)
#Accuracy = 100%
```

3.2 Model Selection

After cross-validation, out of Linear Regression, K Nearest Neighbor and Random Forest, Linear Regression performs the best on the given dataset (100% accuracy), followed by Random Forest (98.63% accuracy) and K Nearest Neighbor (89.35% accuracy) respectively in descending order of accuracy. We therefore freeze the **Linear Regression model as the most effective model for prediction in this case**.

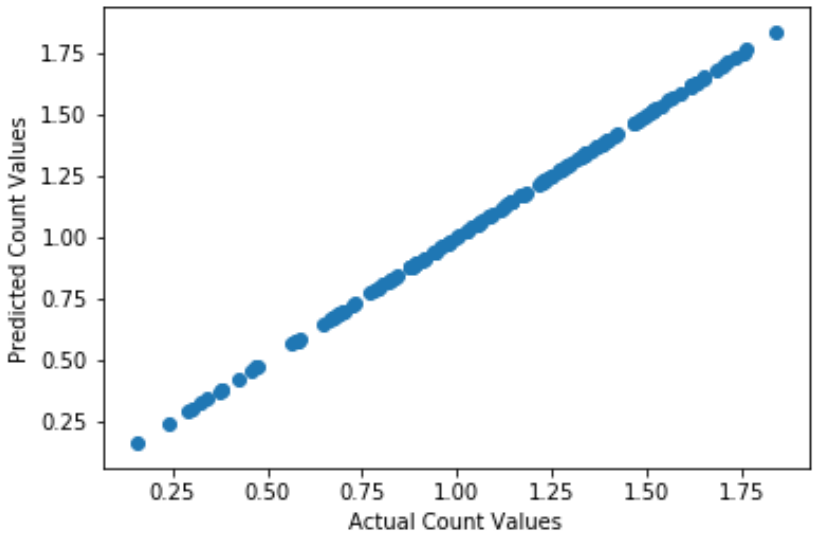
3.3 Answers to Questions

1. Example of output with a sample input.

Following is the code-snippet for finding the sample output:

```
#After cross-validation, out of Decision Tree, Linear Regression, K Nearest Neighbor
#and Random Forest, Linear Regression performs the best on the given dataset,
#followed by Random Forest, K Nearest Neighbor and Decision Tree respectively in
#descending order of accuracy. We therefore freeze the Linear Regression model as
#the most effective model for prediction in this case.

#Linear Regression Equation
wd = data_preprocessed$workingday
tm = data_preprocessed$temp
ca = data_preprocessed$casual
rg = data_preprocessed$registered
cnt = -9.082e-17*wd + 1.392e-16*tm + 1.000e+00*ca + 1.000e+00*rg

#sample input output verification
#Taking 1st observation from dataset
wd_sample = 0
tm_sample = 0.344167
ca_sample = 331
rg_sample = 654
cnt_sample = -9.082e-17*0 + 1.392e-16*0.344167 + 1.000e+00*331 + 1.000e+00*654
print(cnt_sample)
```

Linear Regression Curve



Appendix A - R Code

```
#remove all the objects stored
rm(list=ls())

#set current working directory
setwd("C:/Users/SAYAN/Desktop/Data Science/Projects/Bike Rental Count/R Folder")

#Current working directory
getwd()

#instaling packages
install.packages(c("corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
              "MASS", "rpart", "gbm", "ROSE", "sampling", "DataCombine",
"inTrees","Hmisc","corrplot","usdm","ggplot2"))

#reading CSV data sheet
data = read.csv("Bike_Rental_Count_data.csv", header = T, sep = ",")

#checking data summary
summary(data)

#checking data dimensions
dim(data)

#checking column names of data
colnames(data)

#structure of data
str(data)

#creating reference object with same data
data_1 = data
#str(data_1)

#getting model details

#names(getModelInfo())

#help

#help("<datasetname>")

#changing dteday type to date
data_1$dteday = as.factor(data_1$dteday)
#data_1$dteday = as.numeric(as.POSIXct(data_1$dteday))

#changing season, yr, mnth, holiday, weekday, workingday, weathersit types to factor
data_1$season = as.factor(data_1$season)
data_1$yr = as.factor(data_1$yr)
data_1$mnth = as.factor(data_1$mnth)
data_1$holiday = as.factor(data_1$holiday)
```

```
data_1$weekday = as.factor(data_1$weekday)
data_1$workingday = as.factor(data_1$workingday)
data_1$weathersit = as.factor(data_1$weathersit)


#Missing value analysis
sum(is.na(data_1))

# Missing value analysis is hence not required since all the variables contain the same number of observations,
# i.e., 731 without any NA values.

#Converting INT datatype variables to numeric NUM
data_1$casual = as.numeric(data_1$casual)
data_1$registered = as.numeric(data_1$registered)
data_1$cnt = as.numeric(data_1$cnt)

#check structure of data
str(data_1)

#Outlier analysis in un-normalized independent numeric variables
library(ggplot2)
numerics = sapply(data_1, is.numeric)
numeric_cols = data_1[,numerics]
cnames = c("temp","atemp","hum","windspeed","casual","registered")

for (i in 1:length(cnames))
{
  assign(paste0("oa",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"),
                        data = subset(data))+
          stat_boxplot(geom = "errorbar", width = 0.5) +
          geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                       outlier.size=1, notch=FALSE) +
          theme(legend.position="bottom")+
          labs(y=cnames[i],x="cnt")+
          ggtitle(paste("Box plot",i)))
}

# Plotting box plots to show outlier values
gridExtra::grid.arrange(oa1,oa2,ncol=2)
gridExtra::grid.arrange(oa3,oa4,ncol=2)
gridExtra::grid.arrange(oa5,oa6,ncol=2)

#Replace all outliers with 0 and impute
for(i in cnames){
  #print(i)
  val = data_1[,i][data_1[,i] %in% boxplot.stats(data_1[,i])$out]
  #print(length(val))
  data_1[,i][data_1[,i] %in% val] = NA
}

#replacing NA values in variable "casual" with median
data_1$casual[is.na(data_1$casual)] = median(data_1$casual, na.rm = TRUE)
```

```
for (i in 1:length(data_1$instant))
{
  data_1$cnt[i] = data_1$casual[i] + data_1$registered[i]
}

#replacing NA values of "hum" and "windspeed" with median
data_1$hum[is.na(data_1$hum)] = median(data_1$hum, na.rm = TRUE)
data_1$windspeed[is.na(data_1$windspeed)] = median(data_1$windspeed, na.rm = TRUE)

#correlation test on numeric variables
value = round(cor(as.matrix(data_1[,numerics])),2)

library(corrgram)
corrgram(data_1[,numerics], order=FALSE, lower.panel=panel.shade,
        upper.panel=NULL, text.panel=panel.txt,
        main="Correlation plot")

#extracting only days from "dteday variable
data_1$dteday = weekdays(as.Date(data_1$dteday))

#from dataset we can see and understand that variables "dteday" and "weekday" are redundant in
nature. So
#we can drop either of these variables from further analysis.

#anova test on continuous target/dependent variable and categorical independent variables
anova_result = aov(cnt~season+yr+mnth+holiday+weekday+workingday+weathersit, data=data_1)
summary(anova_result)

##dropping the variables "instant", "atemp" and "dteday" from the dataset after correlation and
anova tests

data_preprocessed = subset(data_1, select = -c(instant,atemp,dteday))

#reordering dataset
data_preprocessed = data_preprocessed[,c(1,2,3,5,4,6,7,8,9,10,11,12,13)]

#checking variable distribution
hist(data_preprocessed$casual)
hist(data_preprocessed$registered)
hist(data_preprocessed$temp)
hist(data_preprocessed$hum)
hist(data_preprocessed$windspeed)

#normalizing variables "casual" and "registered"
for(i in 1:length(data_preprocessed$casual)){
  data_preprocessed$casual[i] = (data_preprocessed$casual[i] - min(data_preprocessed$casual))/
                        (max(data_preprocessed$casual)-min(data_preprocessed$casual))
}
for(i in 1:length(data_preprocessed$registered)){
  data_preprocessed$registered[i] = (data_preprocessed$registered[i] -
min(data_preprocessed$registered))/
    (max(data_preprocessed$registered)-min(data_preprocessed$registered))
}
```

```
#modifying "cnt" variable so that it is sum of "casual" and "registered"
for(i in 1:length(data_preprocessed$cnt)){
  data_preprocessed$cnt[i] = data_preprocessed$casual[i] + data_preprocessed$registered[i]
}

#------------------------------------------------------------------------
#head of data

head(data_preprocessed,10)

data_preprocessed$season = as.numeric(data_preprocessed$season)
data_preprocessed$yr = as.numeric(data_preprocessed$yr)
data_preprocessed$mnth = as.numeric(data_preprocessed$mnth)
data_preprocessed$holiday = as.numeric(data_preprocessed$holiday)
data_preprocessed$weekday = as.numeric(data_preprocessed$weekday)
data_preprocessed$workingday = as.numeric(data_preprocessed$workingday)
data_preprocessed$weathersit = as.numeric(data_preprocessed$weathersit)

#removing skewness in the variable "casual"

data_preprocessed$casual = sqrt(data_preprocessed$casual)

for(i in 1:length(data_preprocessed$cnt)){
  data_preprocessed$cnt[i] = data_preprocessed$casual[i] + data_preprocessed$registered[i]
}

# data_preprocessed$season = as.factor(data_preprocessed$season)
# data_preprocessed$yr = as.factor(data_preprocessed$yr)
# data_preprocessed$mnth = as.factor(data_preprocessed$mnth)
# data_preprocessed$holiday = as.factor(data_preprocessed$holiday)
# data_preprocessed$weekday = as.factor(data_preprocessed$weekday)
# data_preprocessed$workingday = as.factor(data_preprocessed$workingday)
# data_preprocessed$weathersit = as.factor(data_preprocessed$weathersit)

#perform k-fold cross-validation to handle overfitting
library(caret)
library(caTools)
index_cv = createDataPartition(data_preprocessed$cnt, p=0.80, list = FALSE)
train_cv = data_preprocessed[index_cv,]
test_cv = data_preprocessed[-index_cv,]
control_parameters = trainControl(method = "cv", number = 10,
                          savePredictions = TRUE, classProbs = TRUE)

#cross-validated for KNN
model_cv_knn = train(cnt~., data = train_cv, method = "knn",
                 trControl = control_parameters)       # optimal k = 5, MAE=12.84%, RMSE=18.56%,
Rsquared=76.96%

#cross-validated model for Linear Regression
model_cv_lm = train(cnt~., data = train_cv, method = "lm",
                trControl = control_parameters)      # MAE=0%, RMSE=0%, Rsquared=100%

#cross-validated model for Random Forest
model_cv_rf = train(cnt~., data = train_cv, method = "rf",
                trControl = control_parameters)  # optimal mtry = 12, MAE=1.61%, RMSE=2.43%,
Rsquared=99.63%
```

```
#predict values for KNN, for example
predictions_cv = predict(model_cv_knn,test_cv)

#create confusion matrix for continuous target variable by using table function
conf_matrix = table(predictions = predictions_cv,actual = test_cv$cnt)

#------------------------------------------------------------------------------

#decision tree regression model design
library(rpart)
train_index = sample(1:nrow(data_preprocessed), 0.80*nrow(data_preprocessed))
train_dataset = data_preprocessed[train_index,]
test_dataset = data_preprocessed[-train_index,]
fit = rpart(cnt~., data=train_dataset, method = "anova")
predictions_dt = predict(fit,train_dataset[,-13])

#calculate model accuracy for decision tree
library(DMwR)
regr.eval(test_dataset[,13], predictions_dt, stats = c("mae","rmse","mape","mse"))

mape = function(y,yhat){
  mean(abs((y-yhat)/y))*100
}

mape(test_dataset[,13],predictions_dt)

#Accuracy = 100 - 165 = -65%
#------------------------------------------------------------------------------

#linear regression modeling
library(usdm)
vif(data_preprocessed[,-13])
vifcor(data_preprocessed[,-13], th=0.9)

#checking linear relation for "casual", "registered" and "cnt" to check impact on
#"casual" and "registered" variables, which in turn determine the total count "cnt".
lm_model_cas = lm(casual~.,data = train_dataset)
summary(lm_model_cas)

lm_model_reg = lm(registered~.,data = train_dataset)
summary(lm_model_reg)

lm_model = lm(cnt~.,data = train_dataset)
summary(lm_model)
predictions_lm = predict(lm_model, test_dataset[1:12])

#calculate model accuracy for linear regression
library(DMwR)
regr.eval(test_dataset[,13], predictions_lm, stats = c("mae","rmse","mape","mse"))
```

```
#Residual Standard Error
sigma(lm_model)/mean(test_dataset$cnt)  #0%

#Confidence Interval
confint(lm_model)

#Accuracy = 100%

#-----------------------------------------------------------------

#K Nearest Neighbor regression modeling
library(FNN)
cnt_train = train_dataset$cnt
cnt_test = test_dataset$cnt
predictions_knn = knn.reg(train_dataset,test_dataset,cnt_train,k=5)

#calculate model accuracy for KNN
#Mean Absolute Error
mean(abs(cnt_test-predictions_knn$pred))  #10.65%

#MAPE
mean(abs((cnt_test-predictions_knn$pred)/(cnt_test)))*100  #22.08%

#Mean Squared Error
mean((cnt_test-predictions_knn$pred)^2)  #2.27%

#Accuracy = 100 - 10.65 = 89.35%

#elbow function
# install.packages("sjPlot")
# install.packages("tidyselect")
# library(sjPlot)
# install.packages('TMB', type = 'source')
# sjc.elbow(test_dataset, steps = 23)

#-----------------------------------------------------------------

#Random Forest Regression Modeling
library(randomForest)
model_rf = randomForest(cnt~., data=train_dataset, mtry=12, importance=TRUE, type="anova")

#No. of variables tried at each split = 12
#No. of trees = 500 (by default)

#prediction on train data
pred_rf_train = predict(model_rf, train_dataset)
table_pred_rf_train = data.frame(pred_rf_train,train_dataset$cnt)

#prediction on test data
pred_rf_test = predict(model_rf, test_dataset)
table_pred_rf_test = data.frame(pred_rf_test,test_dataset$cnt)
```

```
#calculate model accuracy for Random Forest

#MAE
mean(abs(pred_rf_test - test_dataset$cnt))  #1.37%

#RMSE
sqrt(mean((pred_rf_test - test_dataset$cnt)^2))  #2.14%

#Accuracy = 100 - 1.37 = 98.63%

#---------------------------------------------------------------------

#After cross-validation, out of Decision Tree, Linear Regression, K Nearest Neighbor
#and Random Forest, Linear Regression performs the best on the given dataset,
#followed by Random Forest, K Nearest Neighbor and Decision Tree respectively in
#descending order of accuracy. We therefore freeze the Linear Regression model as
#the most effective model for prediction in this case.

#Linear Regression Equation
wd = data_preprocessed$workingday
tm = data_preprocessed$temp
ca = data_preprocessed$casual
rg = data_preprocessed$registered
cnt = -9.082e-17*wd + 1.392e-16*tm + 1.000e+00*ca + 1.000e+00*rg

#sample input output verification
#Taking 1st observation from dataset
wd_sample = 0
tm_sample = 0.344167
ca_sample = 331
rg_sample = 654
cnt_sample = -9.082e-17*0 + 1.392e-16*0.344167 + 1.000e+00*331 + 1.000e+00*654
print(cnt_sample)

#---------------------------------------------------------------------
```

5. References

- Algina, J., & Olejnik, S. (2003). Conducting power analyses for ANOVA and ANCOVA in between-subjects designs. *Evaluation & the Health Professions*
- https://www.r-bloggers.com/
- https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6
- https://blog.datarobot.com/multiple-regression-using-statsmodels
- https://www.datasciencecentral.com/profiles/blogs/top-20-python-libraries-for-data-science-in-2018