ProjectV2

Katherine Penney

8/20/2021

Install Necessary Packages

```
library("gdata")
```

```
## Warning in system(cmd, intern = intern, wait = wait | intern,
## show.output.on.console = wait, : running command 'C:\WINDOWS\system32\cmd.exe /c
## ftype perl' had status 2
```

```
## Warning in system(cmd, intern = intern, wait = wait | intern,
## show.output.on.console = wait, : running command 'C:\WINDOWS\system32\cmd.exe /c
## ftype perl' had status 2
```

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.
```

```
##
```

```
## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLSX' (Excel 2007+) files.
```

```
##
```

```
## gdata: Run the function 'installXLSXsupport()'
## gdata: to automatically download and install the perl
## gdata: libaries needed to support Excel XLS and XLSX formats.
```

```
##
## Attaching package: 'gdata'
```

```
## The following object is masked from 'package:stats':
##
##     nobs
```

```
## The following object is masked from 'package:utils':
##
##     object.size
```

```
## The following object is masked from 'package:base':
##
##     startsWith
```

```
library("boot")
library("plyr")
library("readxl")
library("randomForest")
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:gdata':
##
##     combine
library("ggplot2")
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
library("pROC")
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
library("reshape2")
library("dplyr")
##
## Attaching package: 'dplyr'
## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## The following objects are masked from 'package:gdata':
##
##     combine, first, last
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library("forecast")
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library("neuralnet")
```

```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

Import DataSet into Data Frame

```
DataSetDataFrame1 <- read_excel("C:/Users/klein/Desktop/IST 687/R Code/Project/SeoulBikeData-123.xlsx")
DF1 <- data.frame(DataSetDataFrame1)
```

Remove Non-Functioning Days

Create DataSet for Neural Network

```
DF2 <- DF1[DF1$Functioning_Day=="Yes",]
DF2 <- DF2[,(1:13)]
DF3 <- DF2[,-2]
```

Random Forest w Full Dataset

```
RF1 <- randomForest(DF1[,5:13], (DF1[,1]))
RF1
```

```
##
## Call:
##  randomForest(x = DF1[, 5:13], y = (DF1[, 1]))
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.8810261
##                     % Var explained: 74.4
```

Confusion Matrix w Full Dataset

```
RF2 <- randomForest(DF1[,5:13], as.factor(DF1[,1]))
RF2
```

```
##
## Call:
##  randomForest(x = DF1[, 5:13], y = as.factor(DF1[, 1]))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 34.95%
```

```
## Confusion matrix:
##     0    1   2   3   4   5   6   7  8 9 10 class.error
## 0  36   55  46  54  47  26  17  11  2 0  1  0.87796610
## 1   8 3023 247  15  14   2   1   1  0 0  0  0.08698278
## 2  10  426 899 219  38  16   8   5  0 0  0  0.44540407
## 3  13   46 265 749 178  22   7  11  0 0  1  0.42027864
## 4  14   12  52 249 492  77  27   5  1 0  0  0.47039828
## 5   6    9  32  20 159 193  83  24  4 0  0  0.63584906
## 6   5    2  13  14  47  88 170  52  4 2  3  0.57500000
## 7   2    0  11   8  14  27  76 102 13 3  6  0.61068702
## 8   0    0   0   1   1   8  12  27 21 1  0  0.70422535
## 9   0    0   0   0   0   3   3  11  1 6  3  0.77777778
## 10  0    0   0   0   0   0   3   8  2 2  7  0.68181818
```

Random Forest w Non-Functioning Days Removed

```
RF3 <- randomForest(DF2[,5:13], (DF2[,1]))
RF3
```

```
##
## Call:
##  randomForest(x = DF2[, 5:13], y = (DF2[, 1]))
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.5311519
##                     % Var explained: 84.07
```

Confusion Matrix w Non-Function Days Removed

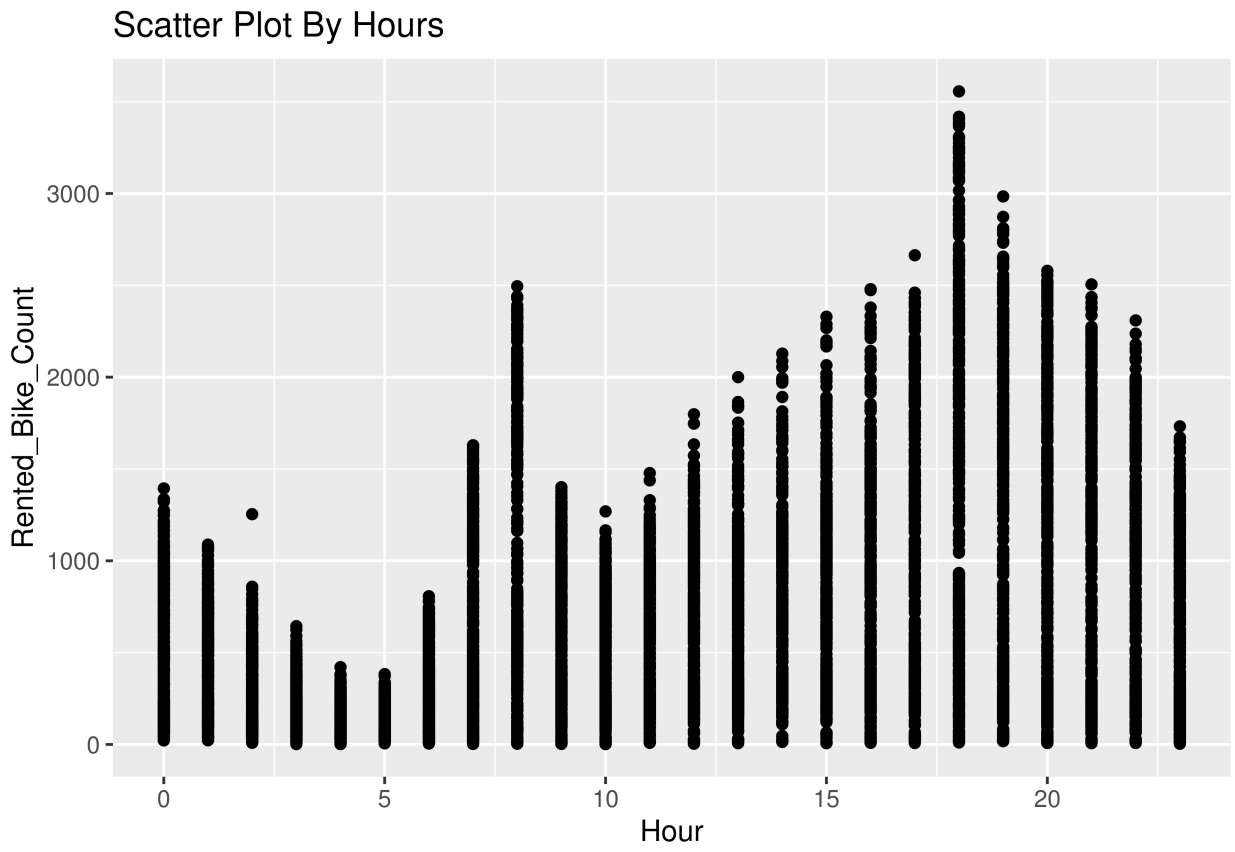
```
RF4 <- randomForest(DF2[,5:13], as.factor(DF2[,1]))
RF4
```

```
##
## Call:
##  randomForest(x = DF2[, 5:13], y = as.factor(DF2[, 1]))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 32.86%
## Confusion matrix:
##       1   2   3   4   5   6   7  8 9 10 class.error
## 1  3024 250  16  16   4   1   0  0 0  0  0.08668076
## 2   430 906 217  36  18  10   3  1 0  0  0.44108575
## 3    43 268 757 183  23   7  10  0 0  1  0.41408669
## 4    17  52 258 486  82  29   4  1 0  0  0.47685684
## 5     9  32  31 146 202  85  20  4 1  0  0.61886792
## 6     3  18  13  43  92 170  50  6 2  3  0.57500000
## 7     0  10  10  11  31  72 108 12 2  6  0.58778626
## 8     0   0   0   1   7  13  30 17 1  2  0.76056338
## 9     0   0   0   1   2   4  12  1 4  3  0.85185185
```

```
## 10    0   0   0   0   0   1   8  3 1  9  0.59090909
```

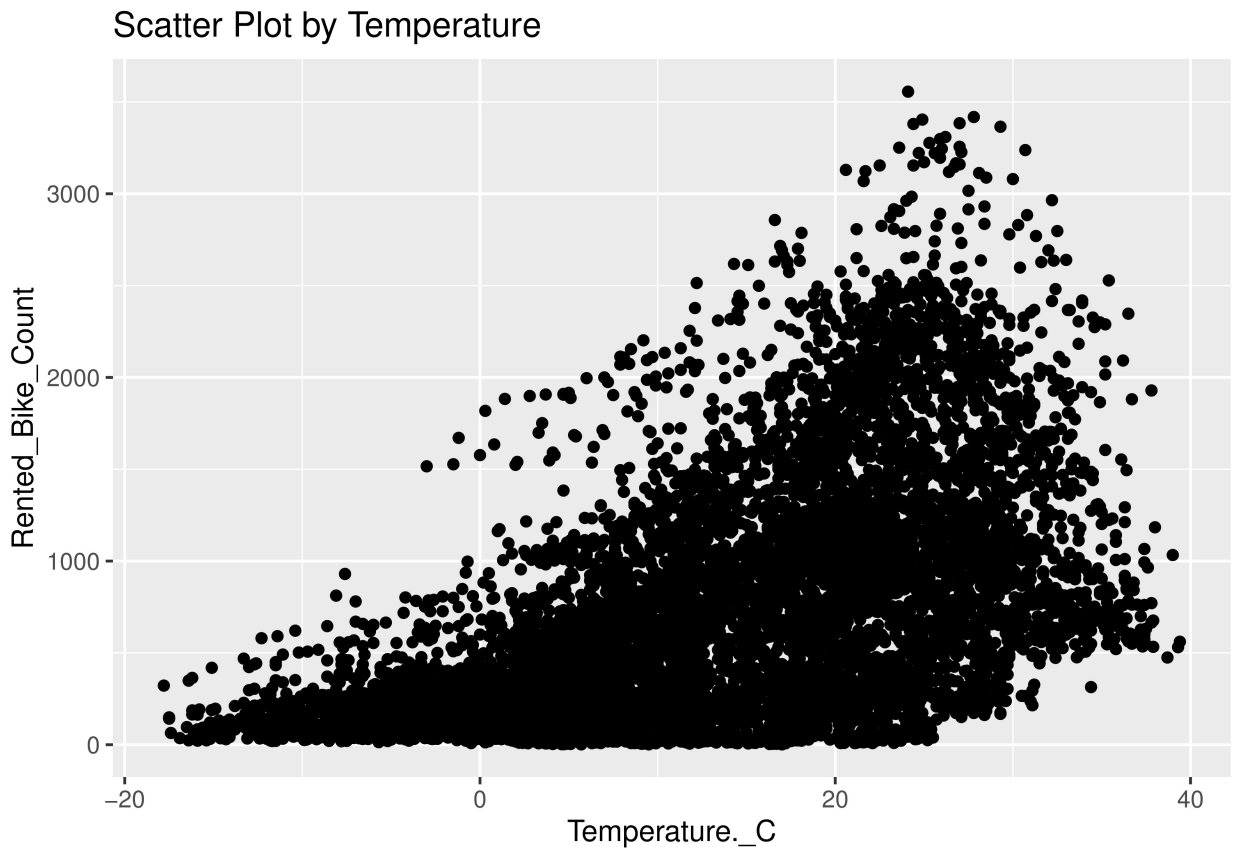
GGPlot Scatter Plot by Hours

```
HourScatter <- ggplot(DF2, aes(x=Hour, y=Rented_Bike_Count)) + geom_point()
HourScatter <- HourScatter + ggtitle("Scatter Plot By Hours")
HourScatter
```



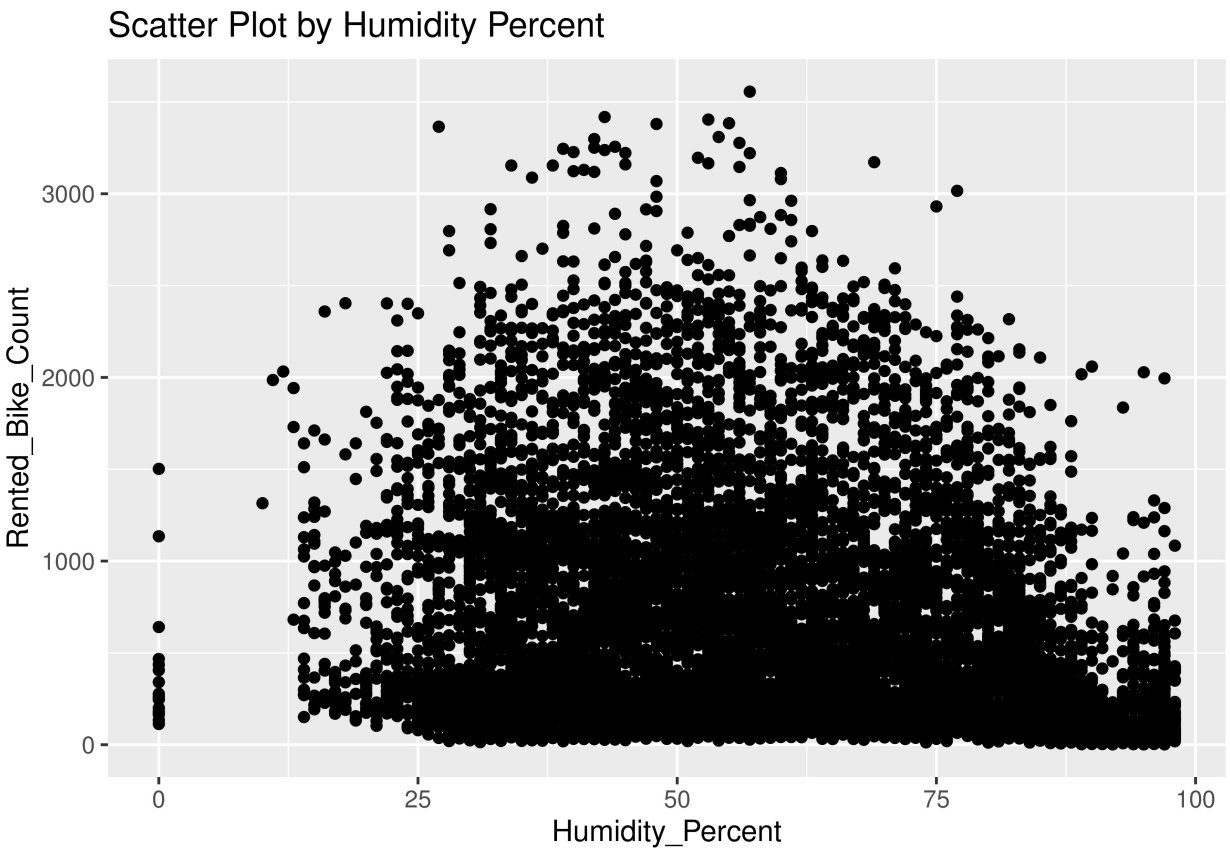
GGplot Scatter Plot by Temp

```
TempScatter <- ggplot(DF2, aes(x=Temperature._C, y=Rented_Bike_Count)) + geom_point()
TempScatter <- TempScatter + ggtitle("Scatter Plot by Temperature")
TempScatter
```



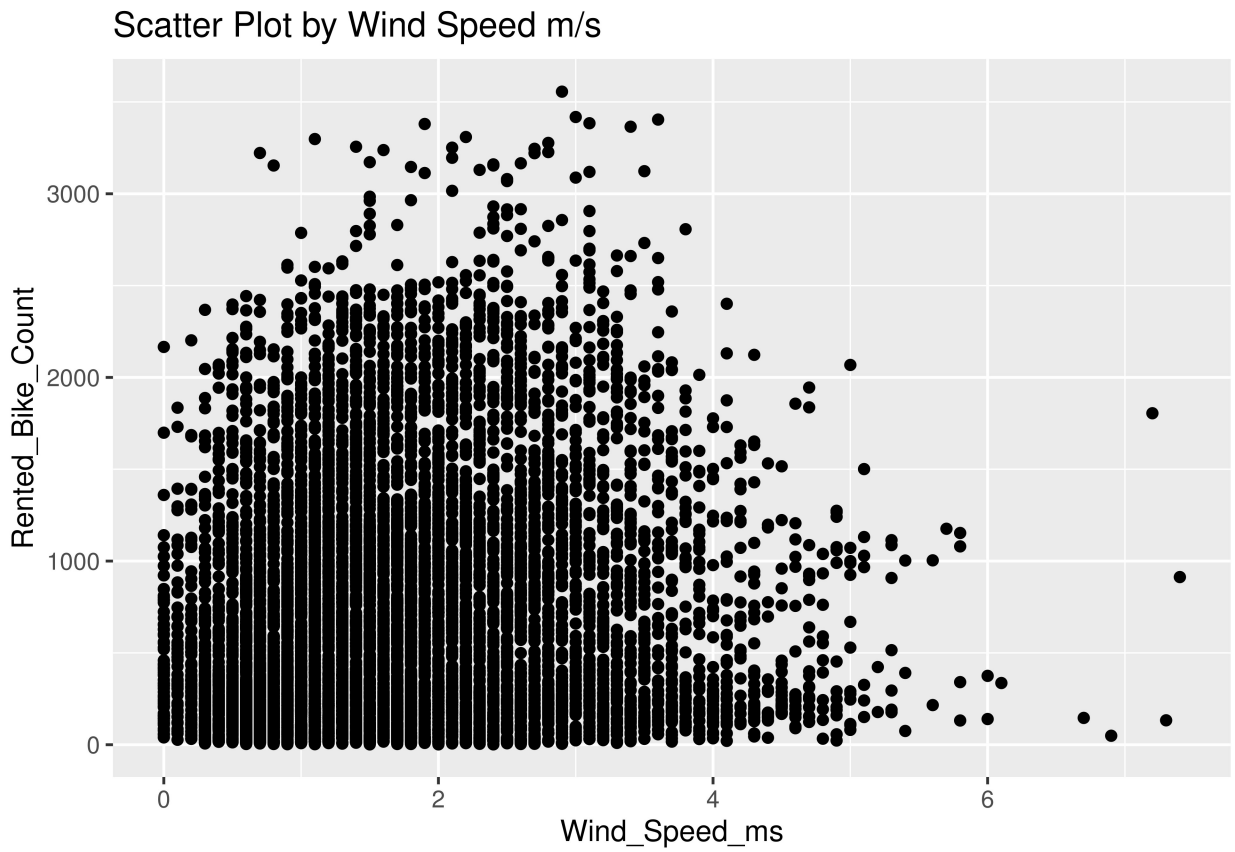
GGplot Scatter Plot by Humidity Percent

```
HumScatter <- ggplot(DF2, aes(x=Humidity_Percent, y=Rented_Bike_Count)) + geom_point()
HumScatter <- HumScatter + ggtitle("Scatter Plot by Humidity Percent")
HumScatter
```



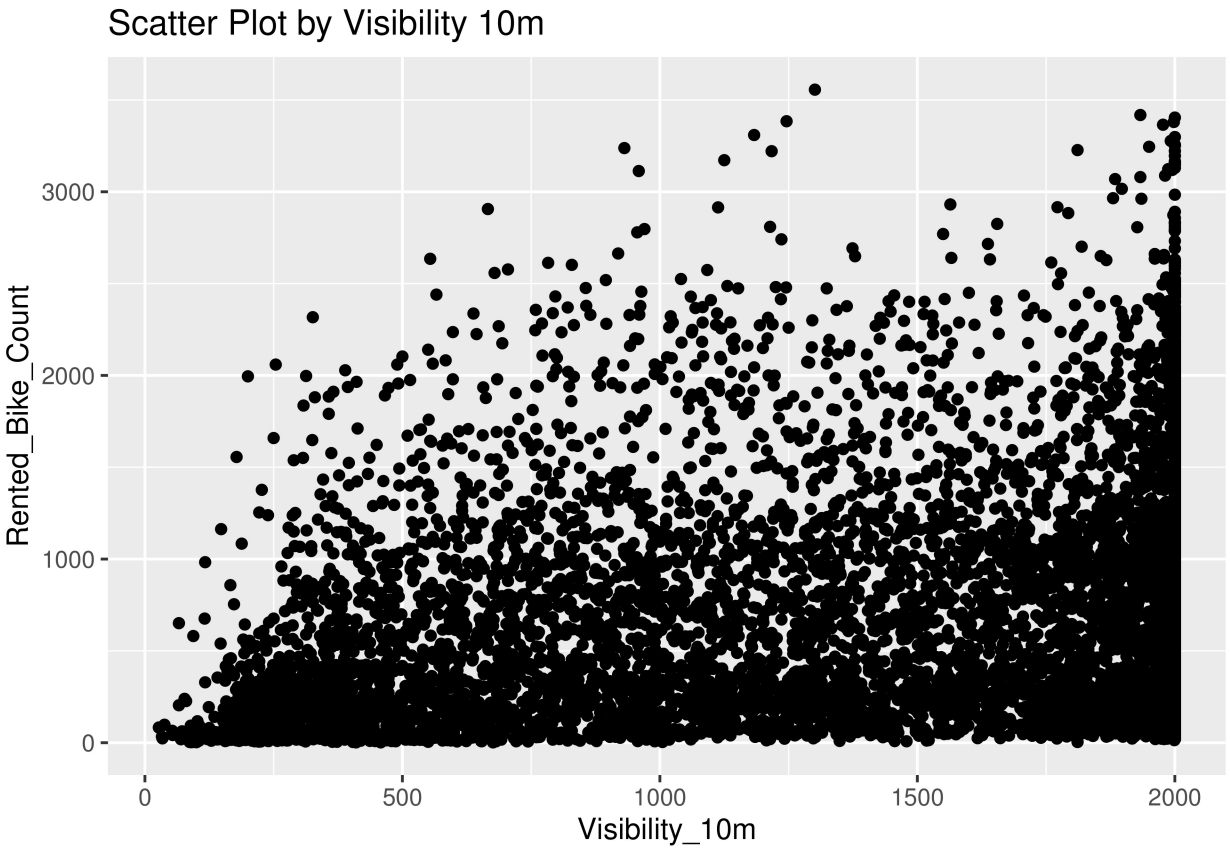
GGplot Scatter Plot by Wind Speed m/s

```
WindScatter <- ggplot(DF2, aes(x=Wind_Speed_ms, y=Rented_Bike_Count)) + geom_point()
WindScatter <- WindScatter + ggtitle("Scatter Plot by Wind Speed m/s")
WindScatter
```



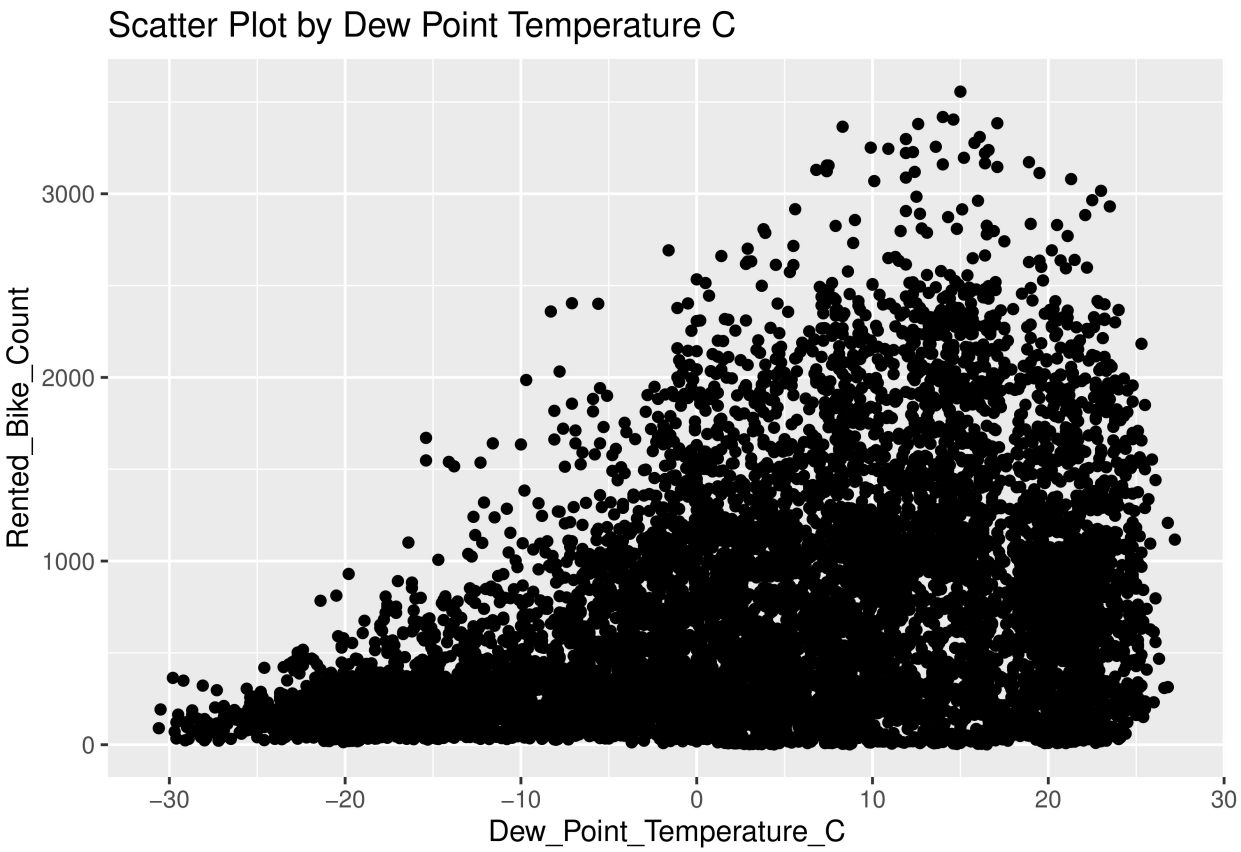
GGplot Scatter Plot by Visibility 10m

```
VisScatter <- ggplot(DF2, aes(x=Visibility_10m, y=Rented_Bike_Count)) + geom_point()
VisScatter <- VisScatter + ggtitle("Scatter Plot by Visibility 10m")
VisScatter
```



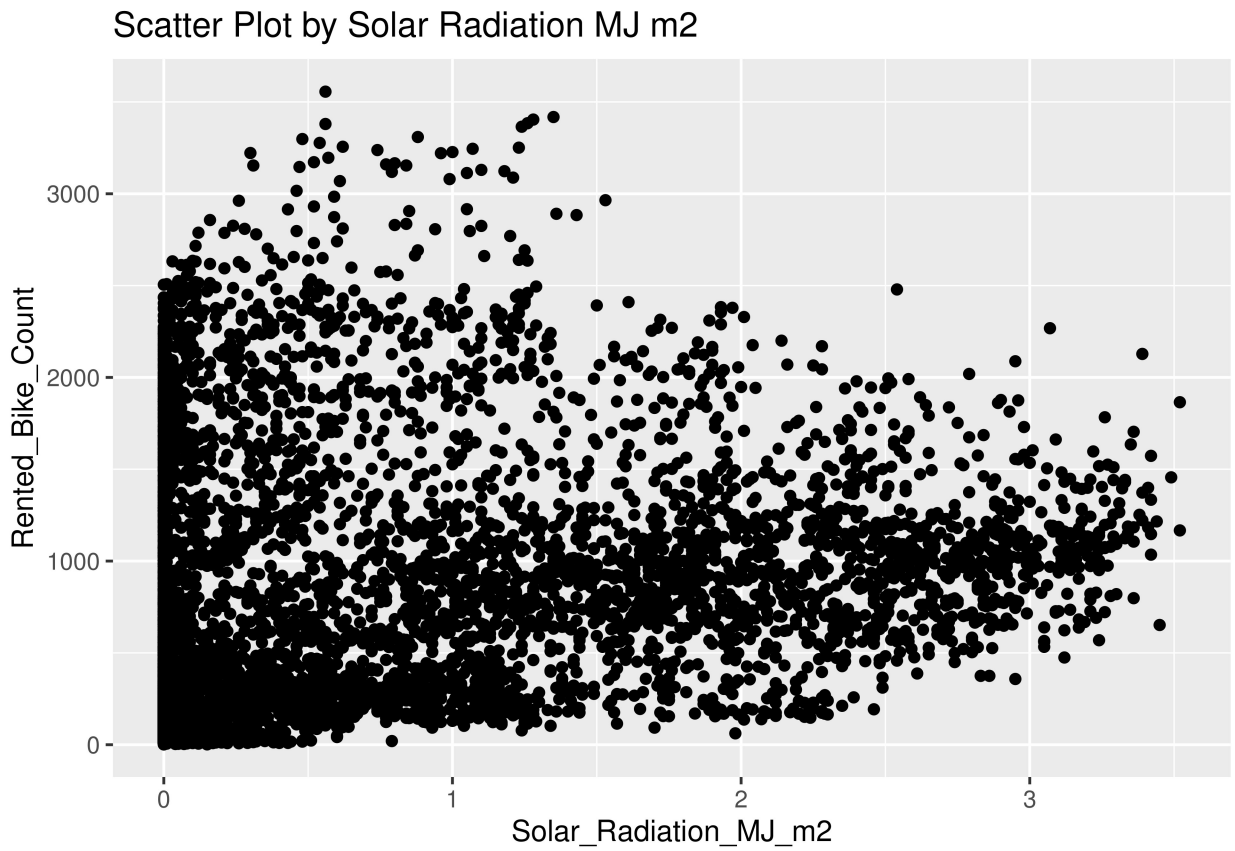
GGplot Scatter Plot by Dew Point Temperature C

```
DewScatter <- ggplot(DF2, aes(x=Dew_Point_Temperature_C, y=Rented_Bike_Count)) + geom_point()
DewScatter <- DewScatter + ggtitle("Scatter Plot by Dew Point Temperature C")
DewScatter
```



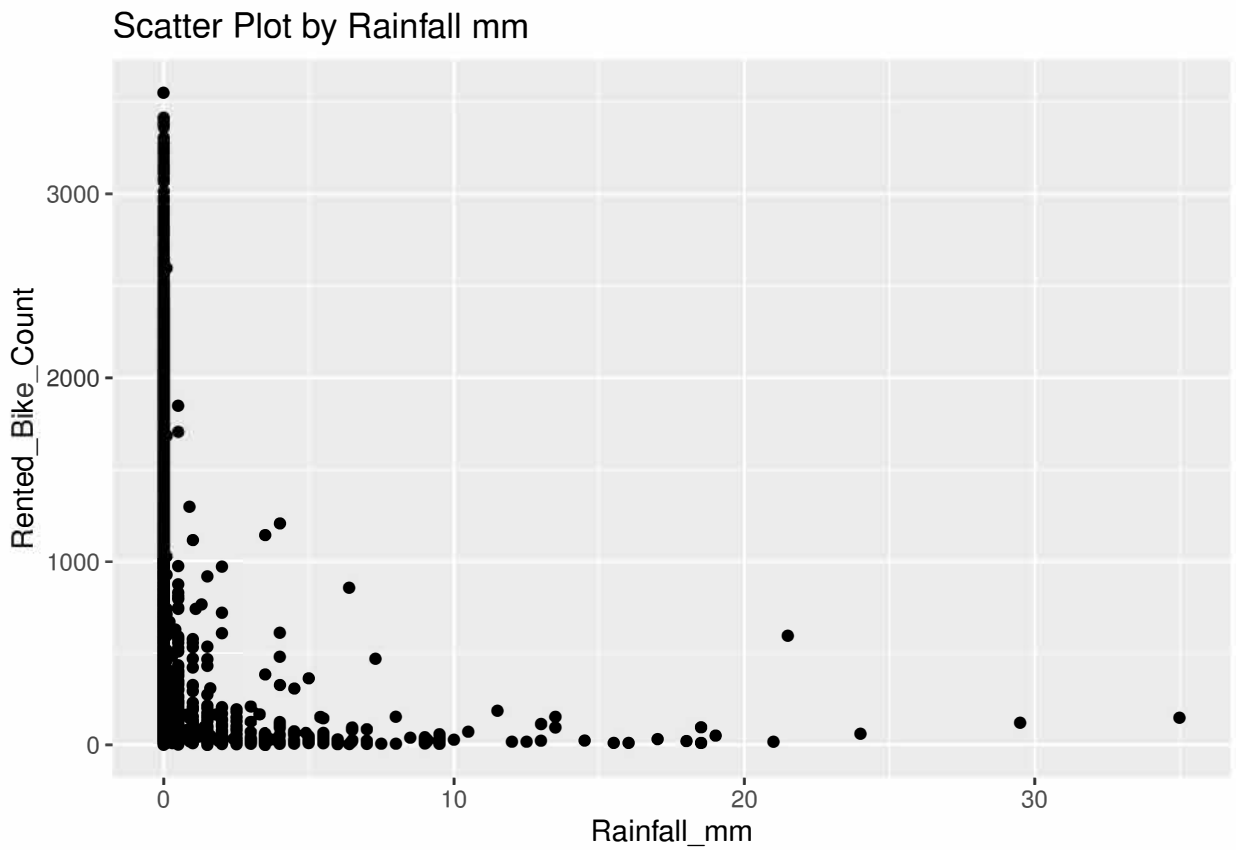
GGplot Scatter Plot by Solar Radiation MJ m2

```
SolarScatter <- ggplot(DF2, aes(x=Solar_Radiation_MJ_m2, y=Rented_Bike_Count)) + geom_point()
SolarScatter <- SolarScatter + ggtitle("Scatter Plot by Solar Radiation MJ m2")
SolarScatter
```



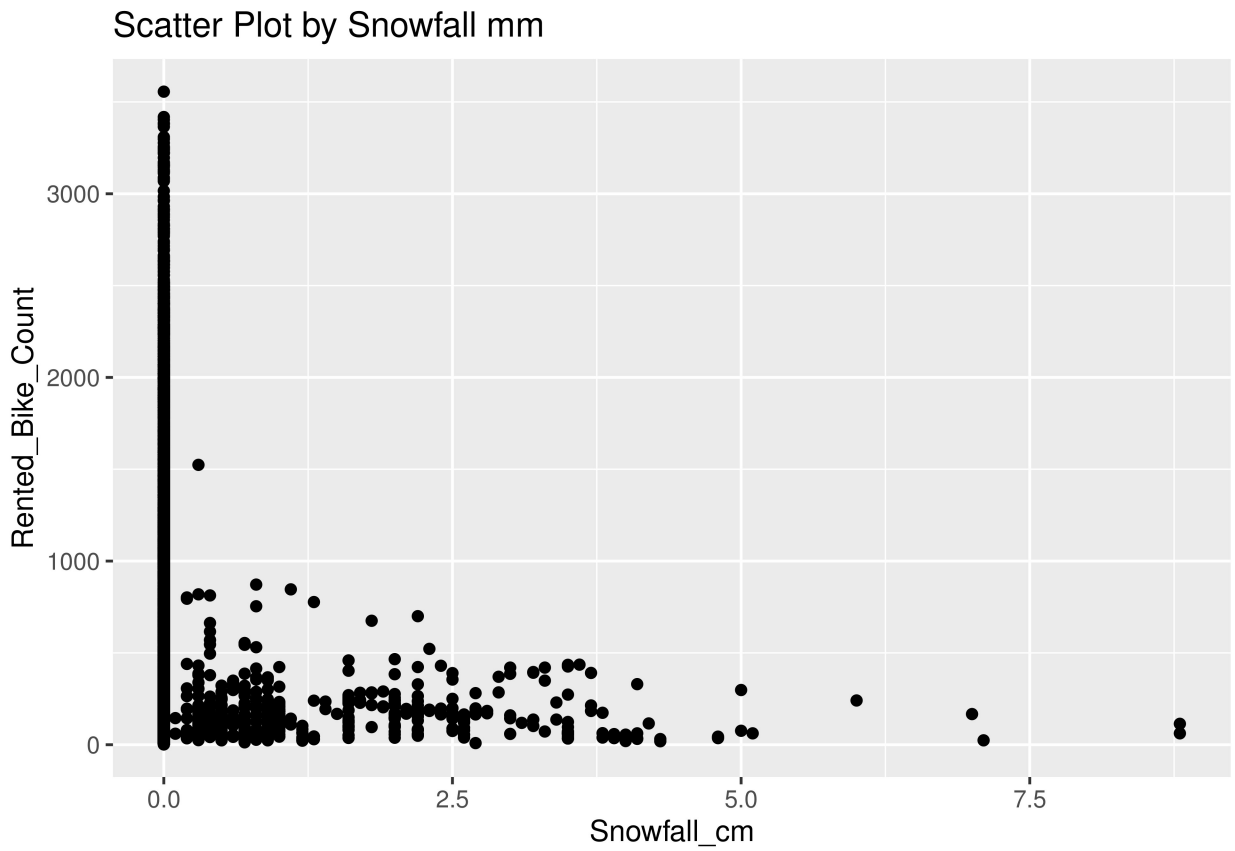
GGplot Scatter Plot by Rainfall mm

```
RainScatter <- ggplot(DF2, aes(x=Rainfall_mm, y=Rented_Bike_Count)) + geom_point()
RainScatter <- RainScatter + ggtitle("Scatter Plot by Rainfall mm")
RainScatter
```



GGplot Scatter Plot by Snowfall mm

```
SnowScatter <- ggplot(DF2, aes(x=Snowfall_cm, y=Rented_Bike_Count)) + geom_point()
SnowScatter <- SnowScatter + ggtitle("Scatter Plot by Snowfall mm")
SnowScatter
```



Regression For All

```
AllVarlm <- lm(Rented_Bike_Count~Rainfall_mm + Hour + Temperature._C + Humidity_Percent + Wind_Speed_ms
summary(AllVarlm)
```

```
##
## Call:
## lm(formula = Rented_Bike_Count ~ Rainfall_mm + Hour + Temperature._C +
##     Humidity_Percent + Wind_Speed_ms + Visibility_10m + Dew_Point_Temperature_C +
##     Solar_Radiation_MJ_m2 + Snowfall_cm, data = DF2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1288.90  -283.73   -44.48   211.22  2249.50
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              585.03562   96.25210   6.078 1.27e-09 ***
## Rainfall_mm              -62.74302    4.51256 -13.904  < 2e-16 ***
## Hour                      28.51675    0.76564  37.246  < 2e-16 ***
## Temperature._C            25.51688    3.77231   6.764 1.43e-11 ***
## Humidity_Percent          -9.29241    1.06914  -8.692  < 2e-16 ***
## Wind_Speed_ms              5.44051    5.32335   1.022  0.30681
## Visibility_10m             0.03111    0.01005   3.095  0.00198 **
## Dew_Point_Temperature_C    7.37912    3.97480   1.856  0.06342 .
## Solar_Radiation_MJ_m2    -78.67244    7.96583  -9.876  < 2e-16 ***
## Snowfall_cm               17.48928   11.50052   1.521  0.12836
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 448.7 on 8455 degrees of freedom
## Multiple R-squared:  0.5125, Adjusted R-squared:  0.512
## F-statistic: 987.7 on 9 and 8455 DF,  p-value: < 2.2e-16
```

Random sampling, Create training and test set

```
samplesize = 0.60 * nrow(DF3)
set.seed(80)
index = sample( seq_len ( nrow ( DF3 ) ), size = samplesize )

datatrain = DF3[ index, ]
datatest = DF3[ -index, ]
```

Scale data for neural network

```
max = apply(DF3 , 2 , max)
min = apply(DF3, 2 , min)
scaled = as.data.frame(scale(DF3, center = min, scale = max - min))
```

creating training and test set, fitting and testing NN

```
trainNN = scaled[index , ]
testNN = scaled[-index , ]

set.seed(2)
NN = neuralnet(Rented_Bike_Count ~ Hour + Temperature._C + Humidity_Percent + Wind_Speed_ms + Visibilit

plot(NN)
```

Prediction Table Using Neural Network

```
temp_test <- subset(testNN, select = c("Hour","Temperature._C", "Humidity_Percent", "Wind_Speed_ms", "V
head(temp_test)
```

```
##         Hour Temperature._C Humidity_Percent Wind_Speed_ms Visibility_10m
## 1  0.3478261      0.1783217        0.3775510    0.14864865      1.0000000
## 3  0.8260870      0.3111888        0.7857143    0.22972973      1.0000000
## 5  0.3913043      0.1975524        0.2755102    0.06756757      0.9635073
## 6  0.6956522      0.3321678        0.5510204    0.56756757      0.3882413
## 9  0.5652174      0.3531469        0.2551020    0.21621622      1.0000000
## 10 0.5217391      0.3409091        0.2346939    0.18918919      1.0000000
##    Dew_Point_Temperature_C Solar_Radiation_MJ_m2 Rainfall_mm Snowfall_cm
## 1                0.1868512           0.002840909           0           0
## 3                0.4688581           0.000000000           0           0
## 5                0.1418685           0.065340909           0           0
## 6                0.4083045           0.068181818           0           0
```

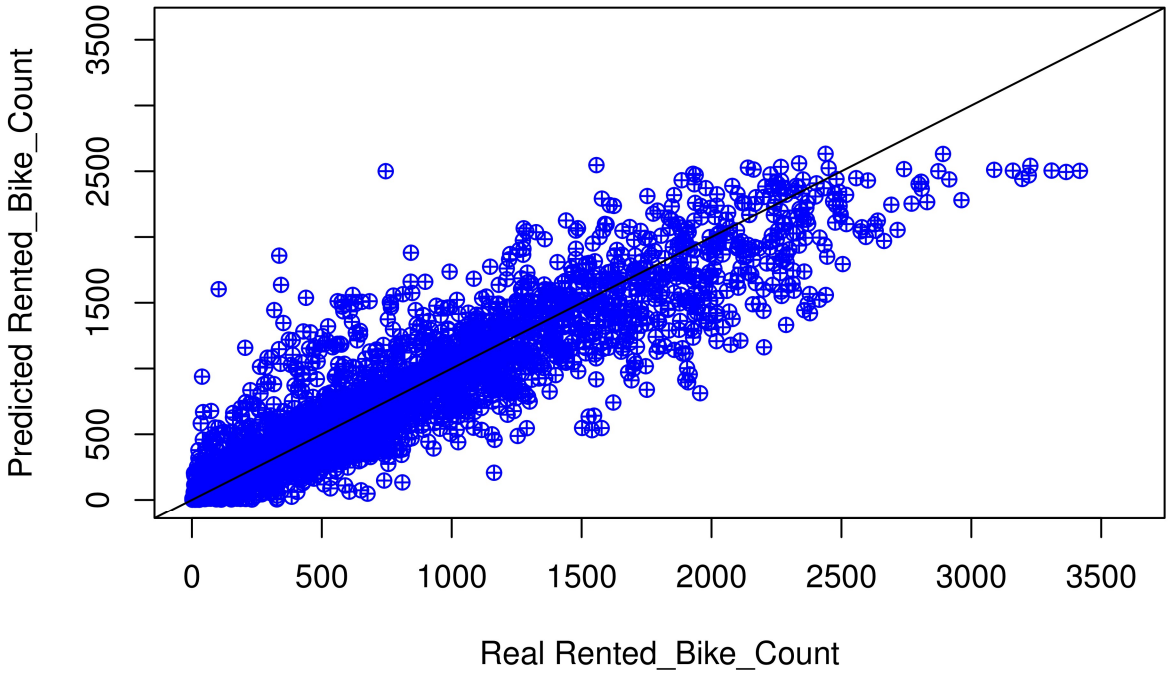
```
## 9                0.2595156           0.329545455           0           0
## 10               0.2318339           0.315340909           0           0
```

```
nn.results <- compute(NN, temp_test)
results <- data.frame(actual = testNN$Rented_Bike_Count, prediction = nn.results$net.result)
head(results)
```

```
##       actual prediction
## 1  0.2611142 0.10972785
## 3  0.1682611 0.09281737
## 5  0.1373101 0.07931785
## 6  0.1356218 0.05321749
## 9  0.1263365 0.09340427
## 10 0.1257738 0.09206768
```

Prediction using neural network via Graph

```
predict_testNN = compute(NN, testNN[,c(4:12)])
predict_testNN = (predict_testNN$net.result * (max(DF3$Rented_Bike_Count) - min(DF3$Rented_Bike_Count))

plot(datatest$Rented_Bike_Count, predict_testNN, col='blue', pch=10, ylim = range(0:3600), xlim = range


abline(0,1)
```



Calculating Root Mean Square Error (RMSE)

```
RMSE.NN = (sum((datatest$Rented_Bike_Count - predict_testNN)^2) / nrow(datatest)) ^ 0.5
RMSE.NN
```

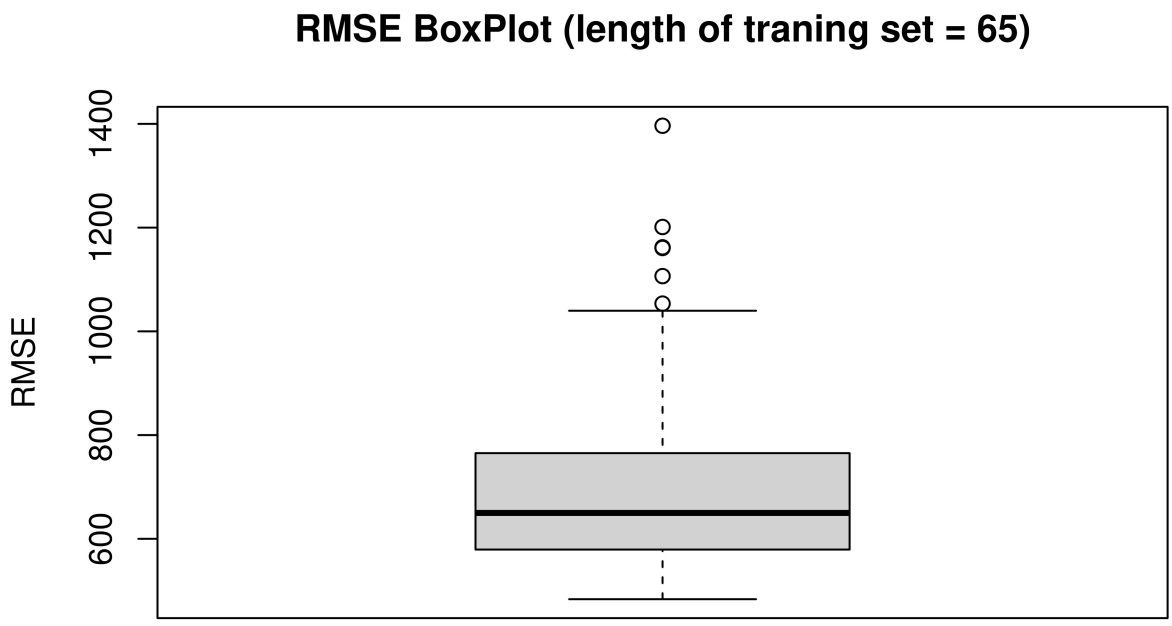
```
## [1] 251.451
```

Running our validationn of our NN within nested loop

```
set.seed(50)
k = 100
RMSE.NN = NULL

List = list( )

for(j in 10:65){
    for (i in 1:k) {
        index = sample(1:nrow(DF3),j )

        trainNN = scaled[index,]
        testNN = scaled[-index,]
        datatest = DF3[-index,]

        NN = neuralnet(Rented_Bike_Count ~ Hour + Temperature._C + Humidity_Percent + Wind_Speed_ms + V
        predict_testNN = compute(NN,testNN[,c(3:12)])
        predict_testNN = (predict_testNN$net.result*(max(DF3$Rented_Bike_Count)-min(DF3$Rented_Bike_Cou

        RMSE.NN [i]<- (sum((datatest$Rented_Bike_Count - predict_testNN)^2)/nrow(datatest))^0.5
    }
    List[[j]] = RMSE.NN
}

Matrix.RMSE = do.call(cbind, List)
```

Creating RMSE boxplot

```
boxplot(Matrix.RMSE[,3], ylab = "RMSE", main = "RMSE BoxPlot (length of traning set = 65)")
```



Measuring the variation in the RMSE over training sets

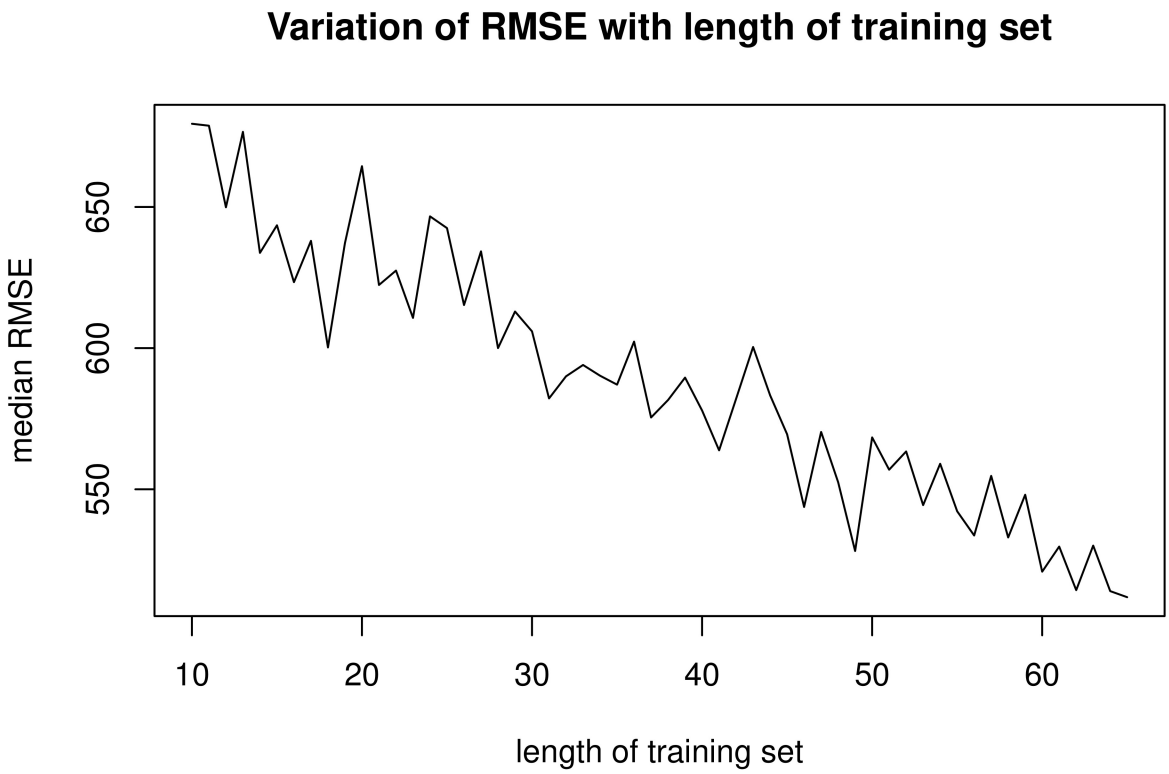
```
library(matrixStats)
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
## The following object is masked from 'package:plyr':
##
##     count
```

```
med = colMedians(Matrix.RMSE)

X = seq(10,65)

plot (med~X, type = "l", xlab = "length of training set", ylab = "median RMSE", main = "Variation of RM
```


Variation of RMSE with length of training set
median RMSE
650
600
550
10
20
30
40
50
60
length of training set