Part-B: Data Analytics

1. a) Write R program to find R-Mean, Median & Mode with the sample data.

Source code:

```
Mean:
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find Mean.
result.mean <- mean(x)
print(result.mean)
```

0utput: 8.22

Applying Trim Option:

```
# Create a vector.
x &lt;- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Find Mean.
result.mean &lt;- mean(x,trim = 0.3)
print(result.mean)
```

output:5.55

Applying NA Option:

```
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)

# Find mean.
result.mean <-  mean(x)
print(result.mean)

# Find mean dropping NA values.
result.mean <-  mean(x,na.rm = TRUE)
print(result.mean)
```

output: NA 8.22

```
# Create the vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find the median.
median.result <- median(x)
print(median.result)
```

output: 5.6

```
# Create the function.
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Create the vector with numbers.
```

```
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)

# Calculate the mode using the user function.
result <- getmode(v)
print(result)

# Create the vector with characters.
charv <- c("o","it","the","it","it")

# Calculate the mode using the user function.
result <- getmode(charv)
print(result)
```

output:2 "it"

b) Write R program to find Analysis and Covariance with the sample data and visualize the regression graphically.

Source code:

```
input <- mtcars[,c("am","mpg","hp")]
print(head(input))
# Get the dataset.
input <- mtcars
# Create the regression model.
result <- aov(mpg~hp*am,data = input)
print(summary(result))
# Create the regression model.
result <- aov(mpg~hp+am,data = input)
print(summary(result))
# Create the regression models.
result1 <- aov(mpg~hp*am,data = input)
result2 <- aov(mpg~hp+am,data = input)
# Compare the two models.
print(anova(result1,result2))
```

output:

```
result <- aov(mpg~hp*am,data = input)
> print(summary(result))
            Df Sum Sq Mean Sq F value   Pr(>F)
hp           1  678.4   678.4  77.391 1.50e-09 ***
am           1  202.2   202.2  23.072 4.75e-05 ***
hp:am        1    0.0     0.0   0.001    0.981
Residuals   28  245.4     8.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
result <- aov(mpg~hp+am,data = input)
> print(summary(result))
          Df Sum Sq Mean Sq F value   Pr(>F)
hp         1  678.4   678.4   80.15 7.63e-10 ***
am         1  202.2   202.2   23.89 3.46e-05 ***
Residuals  29  245.4     8.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
result1 <- aov(mpg~hp*am,data = input)
> result2 <- aov(mpg~hp+am,data = input)
> # Compare the two models.
> print(anova(result1,result2))
Analysis of Variance Table

Model 1: mpg ~ hp * am
Model 2: mpg ~ hp + am
  Res.Df    RSS Df  Sum of Sq     F Pr(>F)
1     28 245.43
2     29 245.44 -1 -0.0052515 6e-04 0.9806


>
```

2. Write R program to find the following Regressions with the sample data and visualize the regressions graphically.

<u>Source code:</u>

a) Linear Regression

Creating Relationship Model and Getting the Coefficients

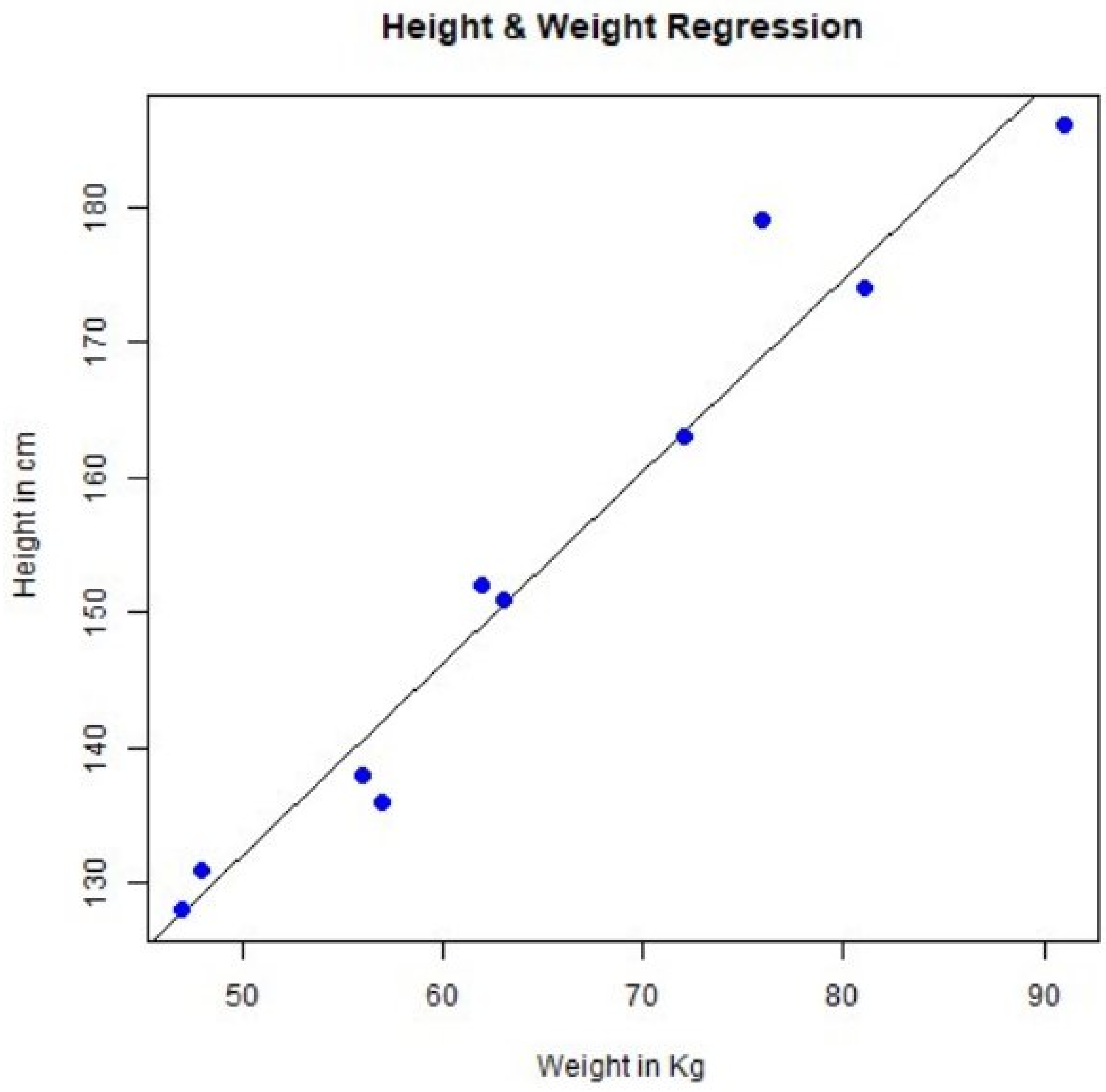
```
#Creating input vector for lm() function
x <- c(141, 134, 178, 156, 108, 116, 119, 143, 162, 130)
y <- c(62, 85, 56, 21, 47, 17, 76, 92, 62, 58)
# Applying the lm() function.
relationship_model<- lm(y~x)
#Printing the coefficient
print(relationship_model)
```

Getting Summary of Relationship Model

```
#Creating input vector for lm() function
x <- c(141, 134, 178, 156, 108, 116, 119, 143, 162, 130)
y <- c(62, 85, 56, 21, 47, 17, 76, 92, 62, 58)
 # Applying the lm() function.
relationship_model<- lm(y~x)
```

```
#Printing the coefficient
print(summary(relationship_model))
```

```
x <- c(141, 134, 178, 156, 108, 116, 119, 143, 162, 130)
y <- c(62, 85, 56, 21, 47, 17, 76, 92, 62, 58)
relationship_model<- lm(y~x)
# Giving a name to the chart file.
png(file = "linear_regression.png")
# Plotting the chart.
plot(y,x,col = "red",main = "Height and Weight Regression",abline(lm(x~y)),cex = 1.3,pch = 16,
xlab = "Weight in Kg",ylab = "Height in cm")
# Saving the file.
dev.off()
```



b) Multiple Regression

Source code:

```
#Creating input data.
input <- mtcars[,c("mpg","wt","disp","hp")]
# Creating the relationship model.
Model <- lm(mpg~wt+disp+hp, data = input)
# Showing the Model.
print(Model)
```

output:

```
data<-mtcars[,c("mpg","wt","disp","hp")]> print(head(input))                mpg    wt disp  hp
Mazda RX4         21.0 2.620  160 110
Mazda RX4 Wag     21.0 2.875  160 110
Datsun 710        22.8 2.320  108  93
Hornet 4 Drive    21.4 3.215  258 110
Hornet Sportabout 18.7 3.440  360 175
Valiant           18.1 3.460  225 105
```

```
Call:
lm(formula = mpg ~ wt + disp + hp, data = input)

Coefficients:
(Intercept)          wt         disp          hp
  37.105505    -3.800891    -0.000937    -0.031157
```

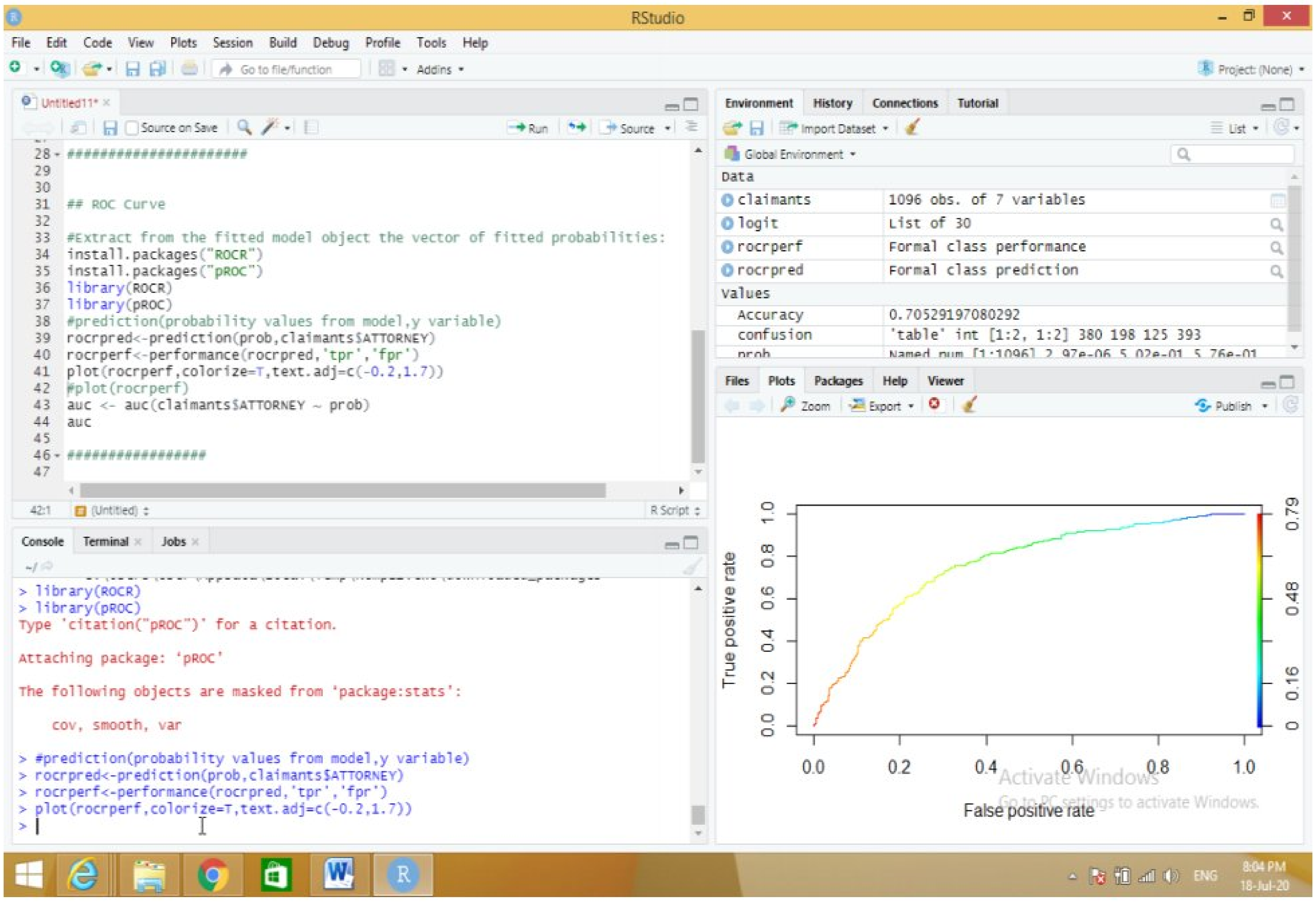
c) Logistic Regression

Source code:

```
claimants<-read.csv("C:/Users/User/Desktop/EXCELR/logistic regression/claimants.csv")
#Finding null values
sum(is.na(claimants))
#Removing null values- na.omit(dataset)
claimants <- na.omit(claimants)
# Logistic Regression
#glm(y~x,family="bin....)
logit<-glm(ATTORNEY ~ factor(CLMSEX) + factor(CLMINSUR) + factor(SEATBELT)
        + CLMAGE + LOSS,family= "binomial",data=claimants)
summary(logit)

# Confusion Matrix Table
#predict(modelobject,testdataset)
prob=predict(logit,type=c("response"),claimants)
prob

#table(dataframe1,dataframe2) ..to create 2X2 matrix
confusion<-table(prob>0.5,claimants$ATTORNEY)
confusion

# Model Accuracy
#adding diagonal elements in the confusion matrix
Accuracy<-sum(diag(confusion))/sum(confusion)
Accuracy

##############

######################


## ROC Curve

#Extract from the fitted model object the vector of fitted probabilities:
install.packages("ROCR")
install.packages("pROC")
library(ROCR)
library(pROC)
#prediction(probability values from model,y variable)
rocrpred<-prediction(prob,claimants$ATTORNEY)
rocrperf<-performance(rocrpred,'tpr','fpr')
```

```
plot(rocrperf,colorize=T,text.adj=c(-0.2,1.7))
#plot(rocrperf)
auc <- auc(claimants$ATTORNEY ~ prob)
auc

##################
```

output:



d) Poisson Regression.

Source code:

```
input <- warpbreaks
print(head(input))

output <-glm(formula = breaks ~ wool+tension, data = warpbreaks,
  family = poisson)
print(summary(output))
```

output:

```
input <- warpbreaks
> print(head(input))
  breaks wool tension
1     26    A       L
2     30    A       L
3     54    A       L
4     25    A       L
5     70    A       L
6     52    A       L
> output <-glm(formula = breaks ~ wool+tension, data = warpbreaks,
+             family = poisson)
> print(summary(output))

Call:
glm(formula = breaks ~ wool + tension, family = poisson, data = warpbreaks)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.6871  -1.6503  -0.4269   1.1902   4.2616
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.69196    0.04541  81.302  < 2e-16 ***
woolB       -0.20599    0.05157  -3.994 6.49e-05 ***
tensionM    -0.32132    0.06027  -5.332 9.73e-08 ***
tensionH    -0.51849    0.06396  -8.107 5.21e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 297.37  on 53  degrees of freedom
Residual deviance: 210.39  on 50  degrees of freedom
AIC: 493.06

Number of Fisher Scoring iterations: 4
```

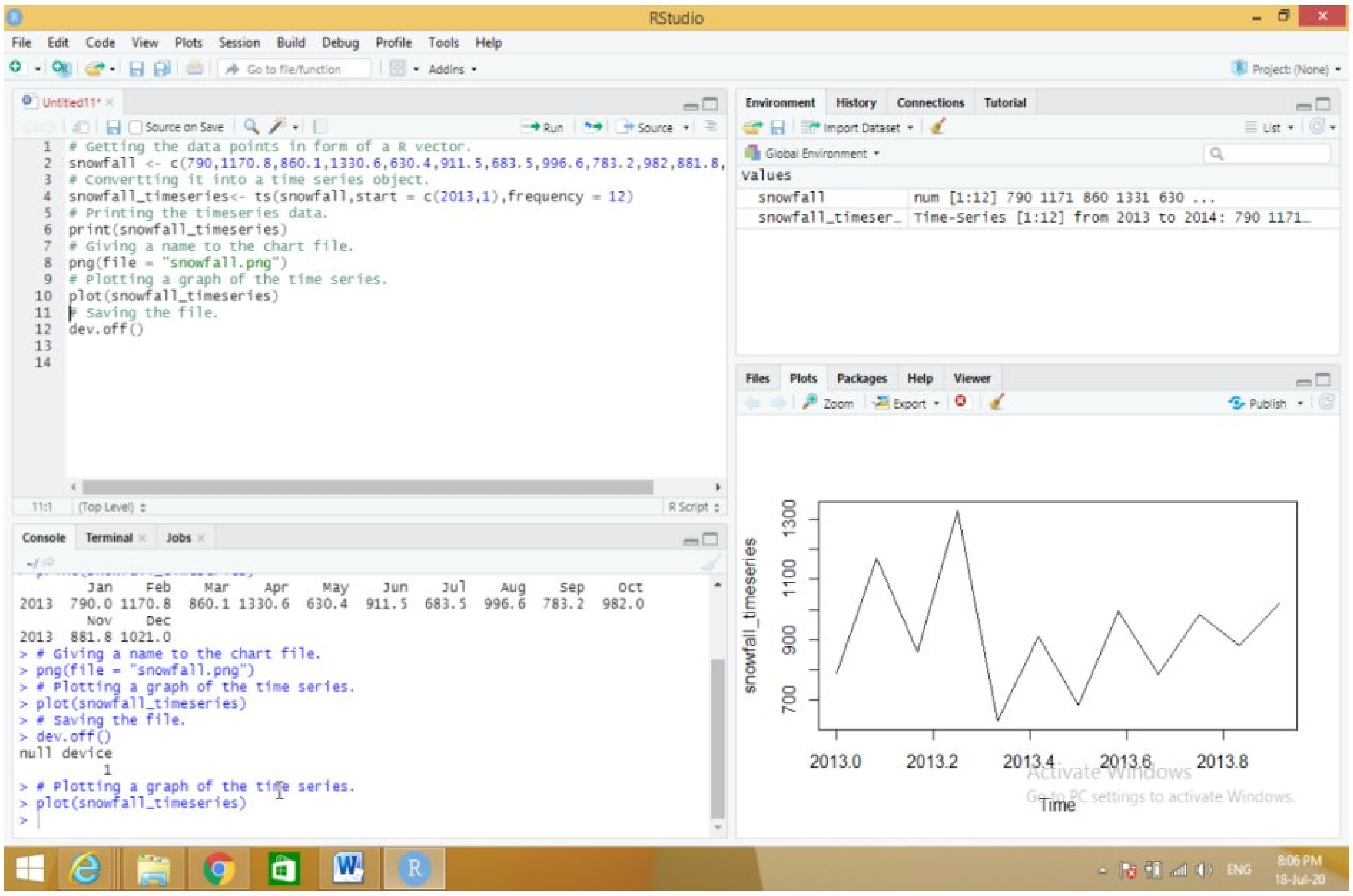
3. a) Write R program to find Time Series Analysis with the sample data and visualize the regression graphically.

Source code:

```
# Getting the data points in form of a R vector.
snowfall <- c(790,1170.8,860.1,1330.6,630.4,911.5,683.5,996.6,783.2,982,881.8,1021)
# Convertting it into a time series object.
snowfall_timeseries<- ts(snowfall,start = c(2013,1),frequency = 12)
# Printing the timeseries data.
print(snowfall_timeseries)
# Giving a name to the chart file.
png(file = "snowfall.png")
# Plotting a graph of the time series.
plot(snowfall_timeseries)
# Saving the file.
dev.off()
```

output:

```
snowfall <- c(790,1170.8,860.1,1330.6,630.4,911.5,683.5,996.6,783.2,982,881.8,1021)
> # Convertting it into a time series object.
> snowfall_timeseries<- ts(snowfall,start = c(2013,1),frequency = 12)
> # Printing the timeseries data.
> print(snowfall_timeseries)
       Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
2013  790.0 1170.8  860.1 1330.6  630.4  911.5  683.5  996.6  783.2  982.0
       Nov    Dec
2013  881.8 1021.0
> # Giving a name to the chart file.
> png(file = "snowfall.png")
> # Plotting a graph of the time series.
> plot(snowfall_timeseries)
> # Saving the file.
> dev.off()
null device
          1
> # Plotting a graph of the time series.
> plot(snowfall_timeseries)
```



b) Write R program to find Non Linear Least Square with the sample data and visualize the regression graphically.

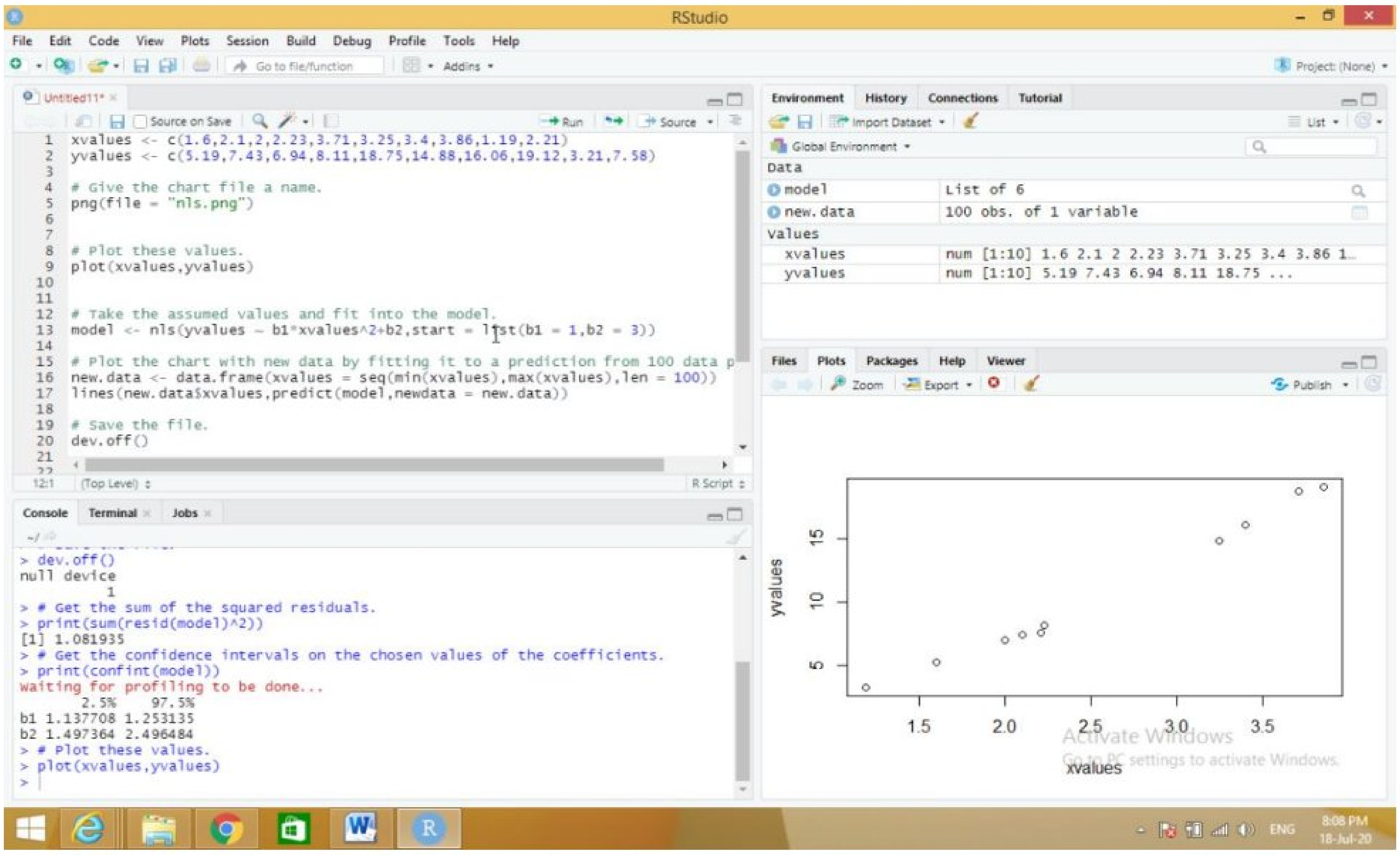
<u>Source code:</u>

```
xvalues <- c(1.6,2.1,2,2.23,3.71,3.25,3.4,3.86,1.19,2.21)
yvalues <- c(5.19,7.43,6.94,8.11,18.75,14.88,16.06,19.12,3.21,7.58)

# Give the chart file a name.
png(file = "nls.png")

# Plot these values.
plot(xvalues,yvalues)

# Take the assumed values and fit into the model.
model <- nls(yvalues ~ b1*xvalues^2+b2,start = list(b1 = 1,b2 = 3))

# Plot the chart with new data by fitting it to a prediction from 100 data points.
new.data <- data.frame(xvalues = seq(min(xvalues),max(xvalues),len = 100))
lines(new.data$xvalues,predict(model,newdata = new.data))

# Save the file.
dev.off()
```

# Get the sum of the squared residuals.
print(sum(resid(model)^2))

# Get the confidence intervals on the chosen values of the coefficients.
print(confint(model))

output:



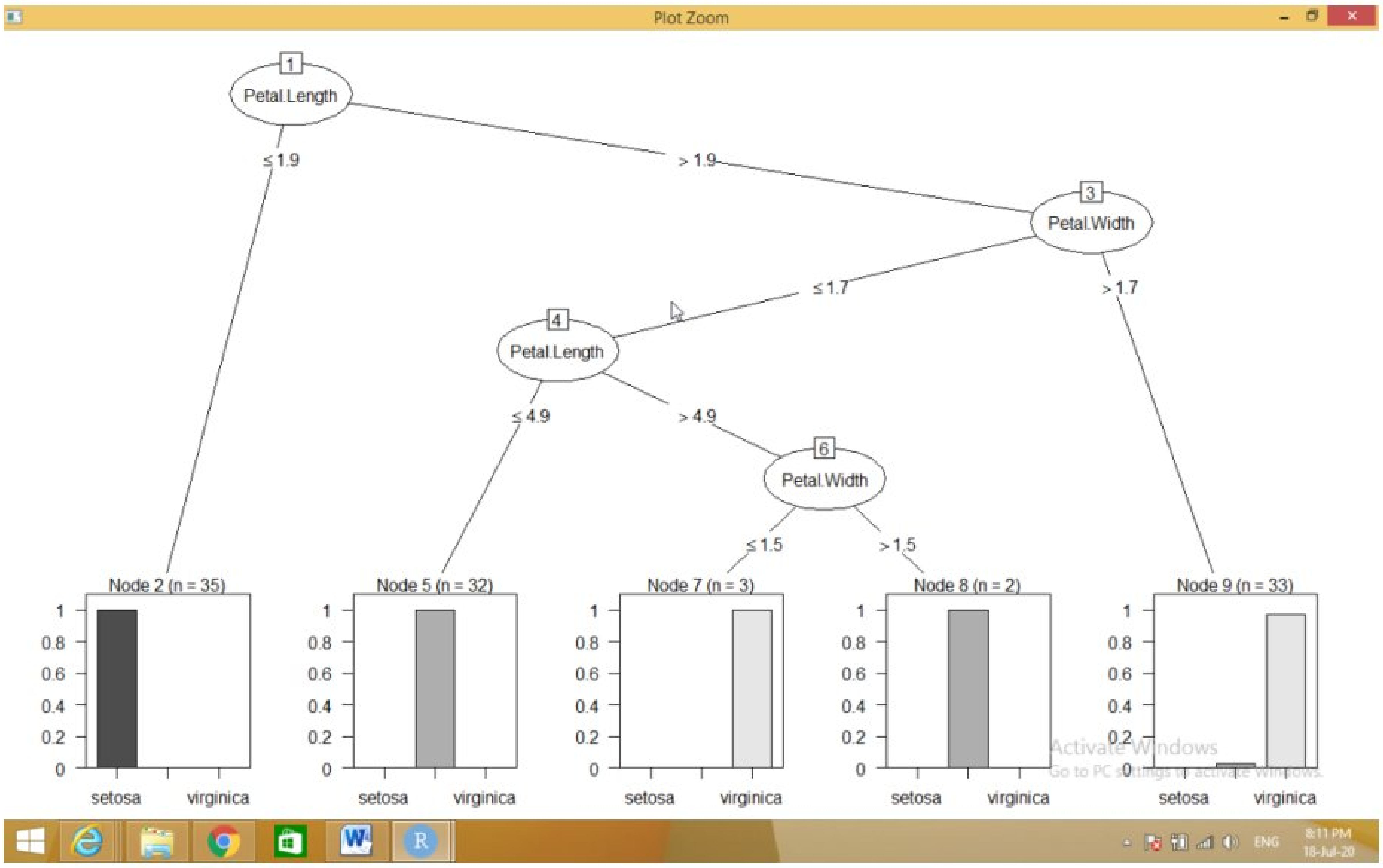
c) Write R program to find Decision Tree with the sample data and visualize the regression graphically.

Source code:

```
#Data Load
data("iris")
#Install the required packages
install.packages("caret")
install.packages("C50")
#Library invoke
library(caret)
library(C50)
#To make the results consistent across the runs

set.seed(7)
#Data Partition
inTraininglocal<-createDataPartition(iris$Species,p=.70,list = F)
```

```
training<-iris[inTraininglocal,]
testing<-iris[-inTraininglocal,]

#Model Building
model<-C5.0(Species~.,data = training)
#Generate the model summary
summary(model)
#Predict for test data set
pred<-predict.C5.0(model,testing[,-5]) #type ="prob"
#Accuracy of the algorithm
a<-table(testing$Species,pred)
sum(diag(a))/sum(a)
#Visualize the decision tree
plot(model)
```

output:



4. Write R program to find the following Distribution with the sample data and visualize the linear regression graphically.

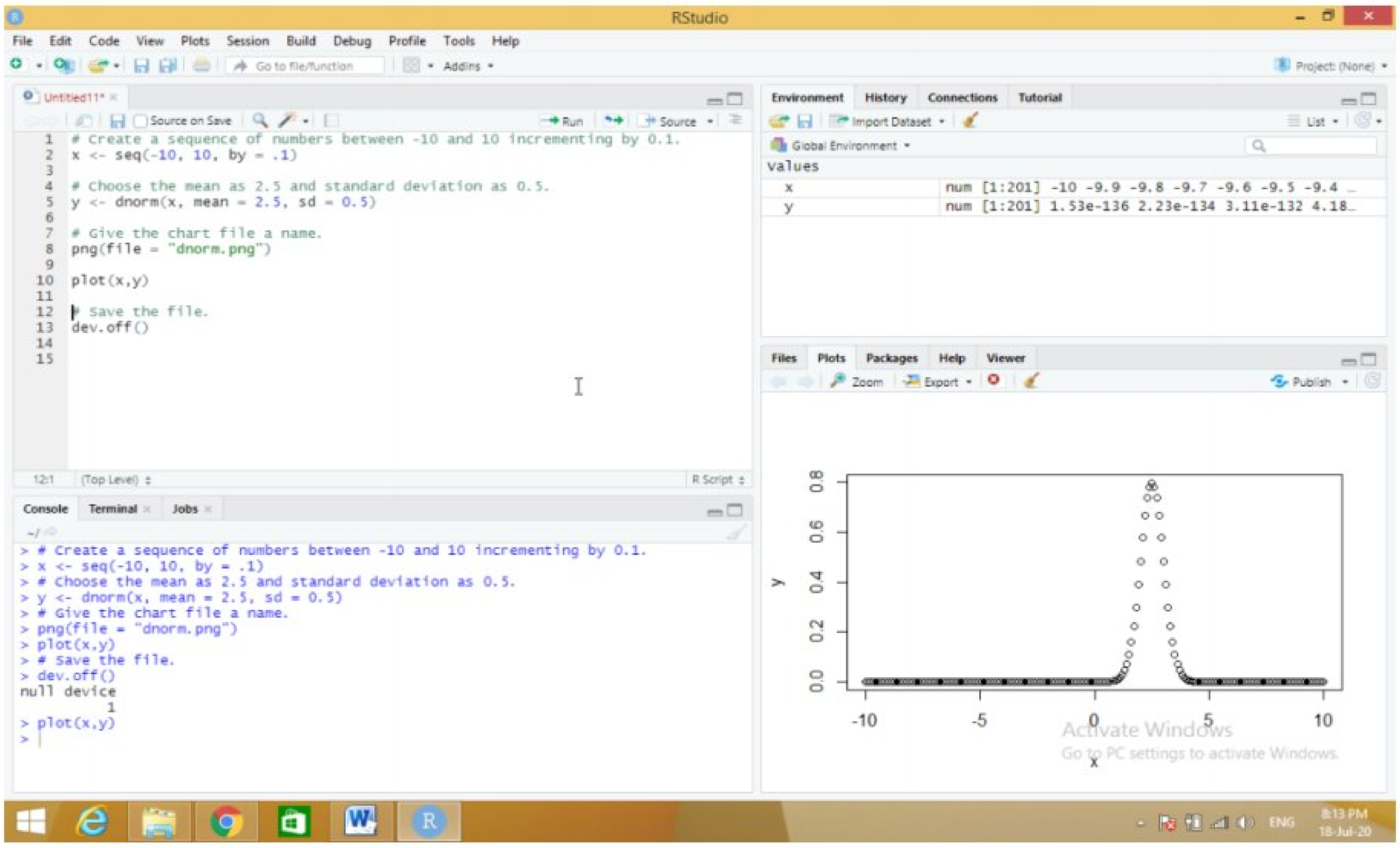
<u>Source code:</u>

a) Normal Distribution

dnorm

```
# Create a sequence of numbers between -10 and 10 incrementing by 0.1.
x <- seq(-10, 10, by = .1)

# Choose the mean as 2.5 and standard deviation as 0.5.
y <- dnorm(x, mean = 2.5, sd = 0.5)

# Give the chart file a name.
png(file = "dnorm.png")

plot(x,y)

# Save the file.
dev.off()
```

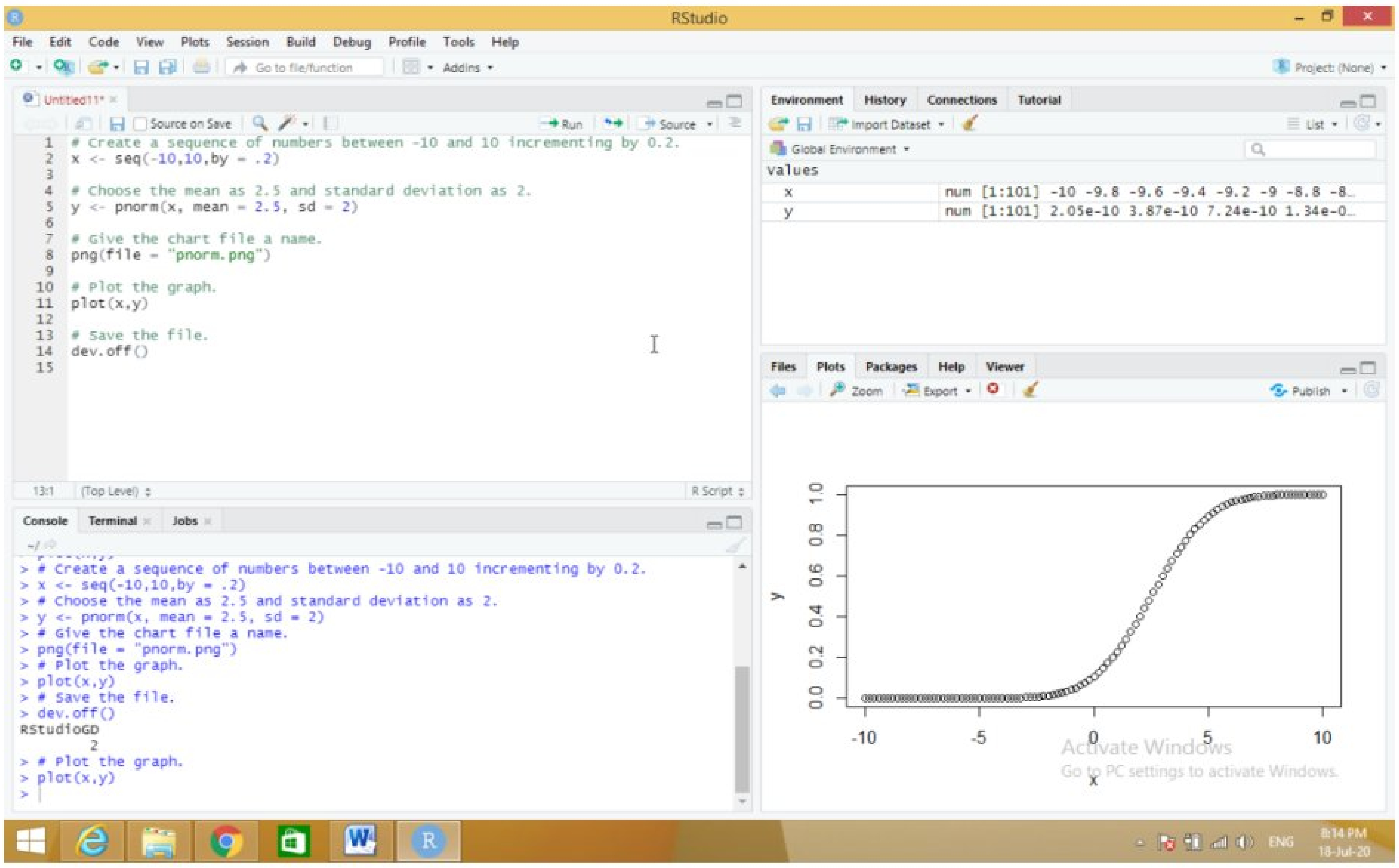
<u>output:</u>



Pnorm

```
# Create a sequence of numbers between -10 and 10 incrementing by 0.2.
x <- seq(-10,10,by = .2)

# Choose the mean as 2.5 and standard deviation as 2.
y <- pnorm(x, mean = 2.5, sd = 2)

# Give the chart file a name.
png(file = "pnorm.png")

# Plot the graph.
plot(x,y)

# Save the file.
dev.off()
```

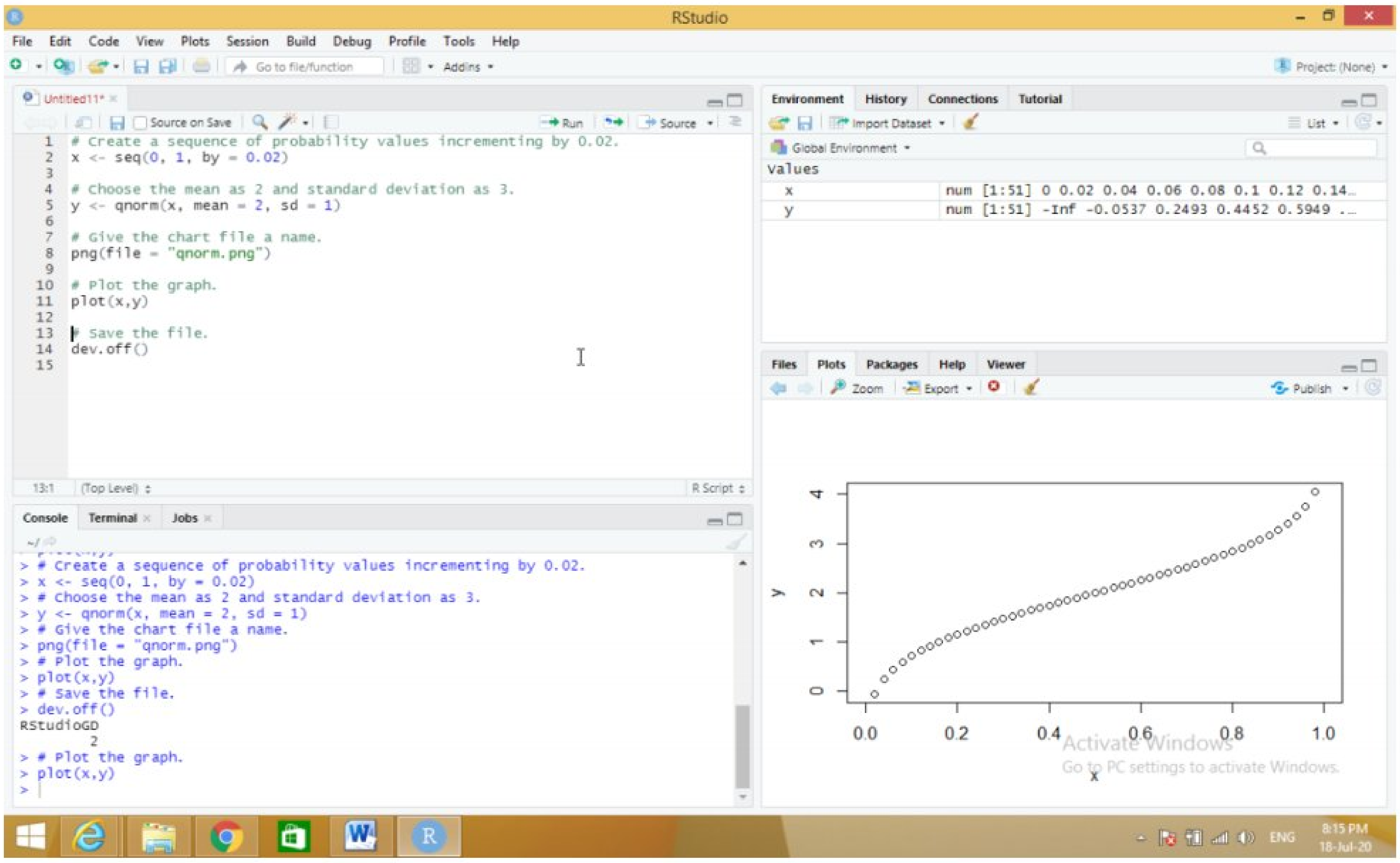
output:



qnorm

```
# Create a sequence of probability values incrementing by 0.02.
x <- seq(0, 1, by = 0.02)

# Choose the mean as 2 and standard deviation as 3.
y <- qnorm(x, mean = 2, sd = 1)

# Give the chart file a name.
png(file = "qnorm.png")

# Plot the graph.
plot(x,y)

# Save the file.
dev.off()
```

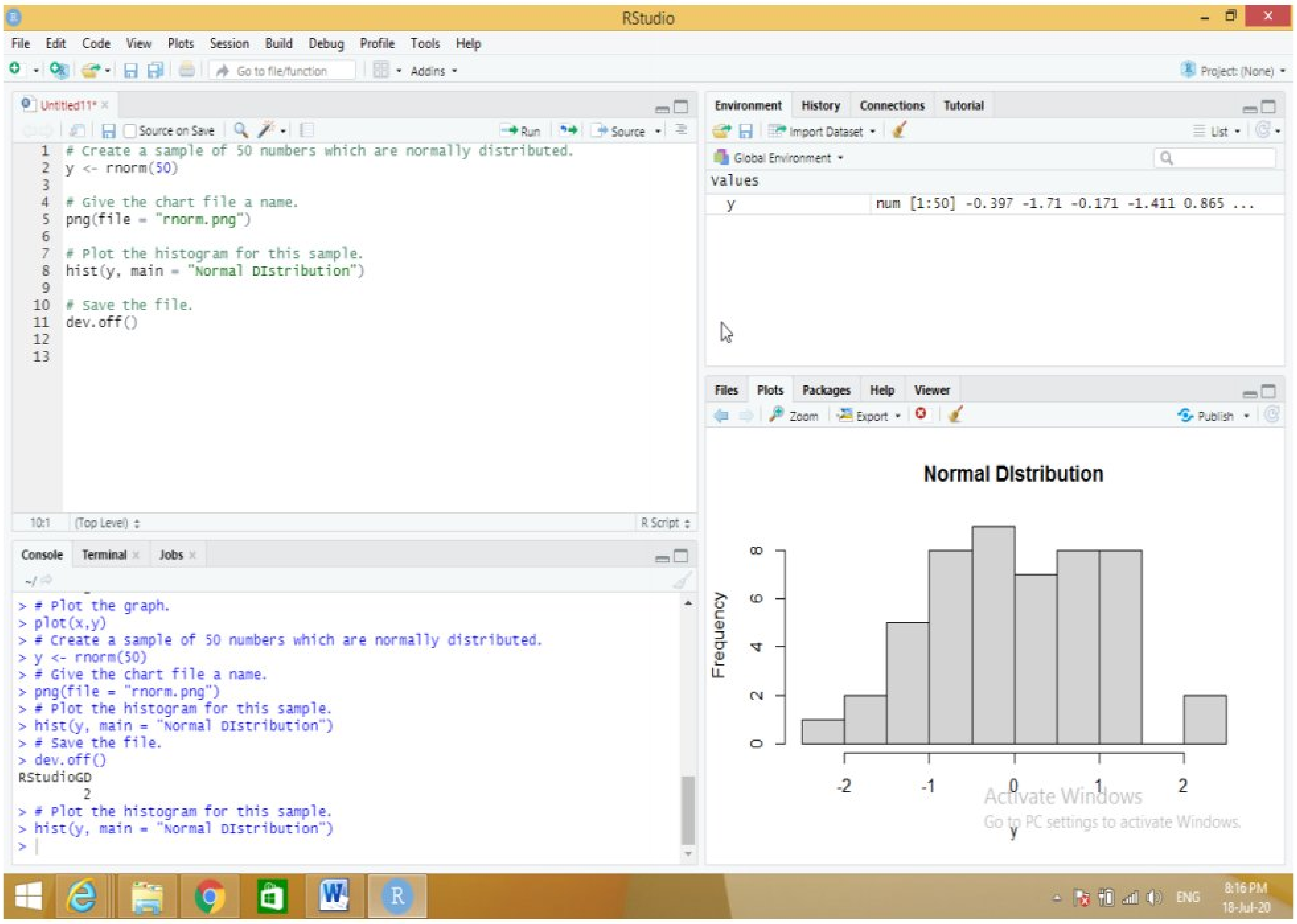
output:



rnorm

```
# Create a sample of 50 numbers which are normally distributed.
y <- rnorm(50)

# Give the chart file a name.
png(file = "rnorm.png")

# Plot the histogram for this sample.
hist(y, main = "Normal DIstribution")

# Save the file.
dev.off()
```

output:



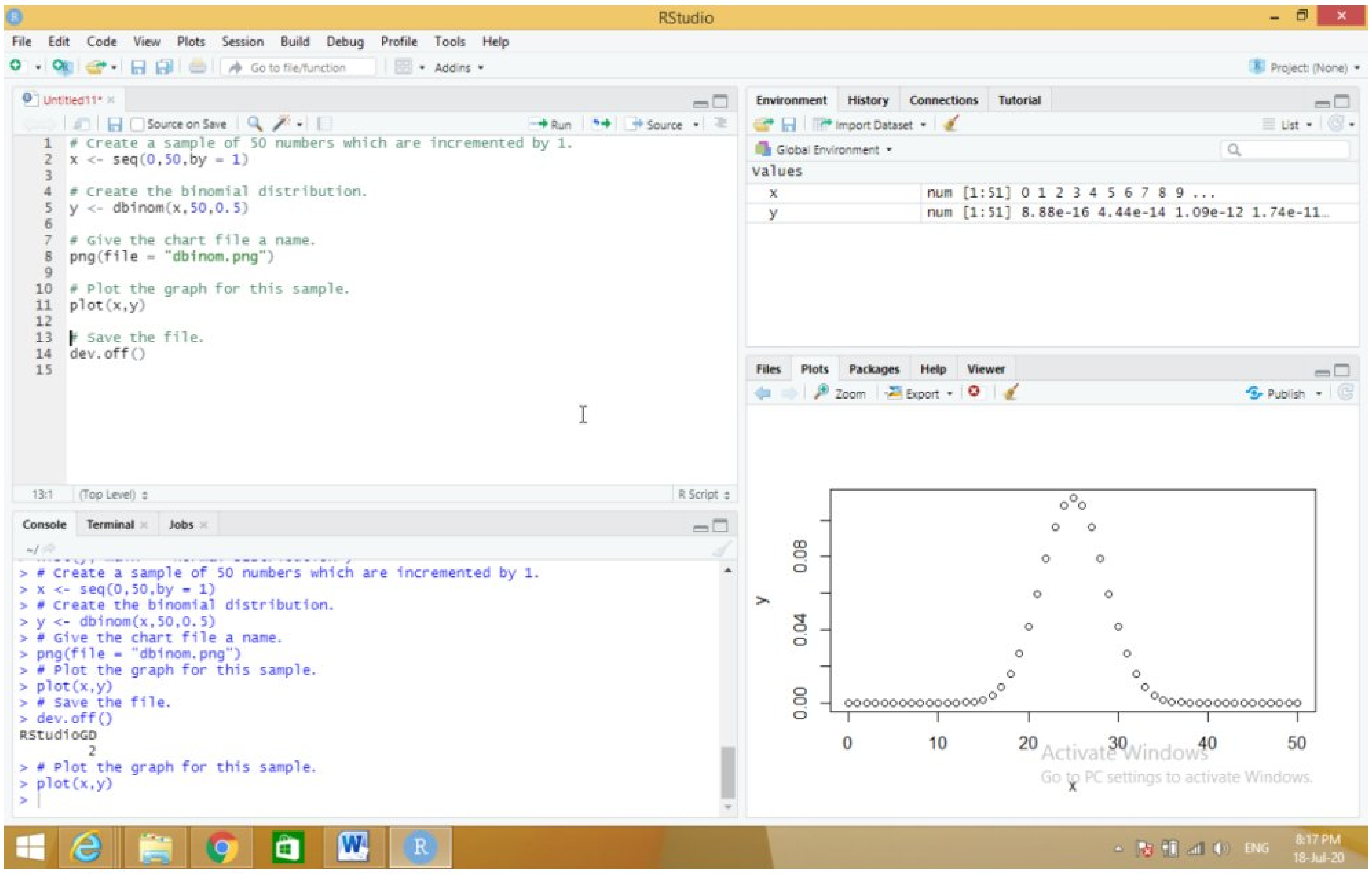
b) Binomial Distribution

dbinom

<u>Source code:</u>

```
# Create a sample of 50 numbers which are incremented by 1.
x <- seq(0,50,by = 1)
# Create the binomial distribution.
y <- dbinom(x,50,0.5)
# Give the chart file a name.
png(file = "dbinom.png")
# Plot the graph for this sample.
plot(x,y)
# Save the file.
dev.off()
```

output:



Pbinom

```
# Probability of getting 26 or less heads from a 51 tosses of a coin.
x <- pbinom(26,51,0.5)

print(x)
```

output:

```
print(x)
[1] 0.610116
```

qbinom

```
# How many heads will have a probability of 0.25 will come out when a coin
# is tossed 51 times.
x <- qbinom(0.25,51,1/2)

print(x)
```

output:

```
print(x)
[1] 23
```

rbinom

```
# Find 8 random values from a sample of 150 with probability of 0.4.
x <- rbinom(8,150,.4)

print(x)
```

output:

```
print(x)
[1] 68 59 55 49 51 59 53 55
```

5. Write R program to do the following tests with the sample data and visualize the results graphically.

Source code:

a) χ2-test

```
library("MASS")
print(str(Cars93))
# Loading the Mass library.

# Creating a data frame from the main data set.
car_data<- data.frame(Cars93$AirBags, Cars93$Type)
# Creating a table with the needed variables.
car_data = table(Cars93$AirBags, Cars93$Type)
print(car_data)
# Performing the Chi-Square test.
print(chisq.test(car_data))
```

output:

```
print(str(Cars93))
'data.frame':    93 obs. of  27 variables:
 $ Manufacturer      : Factor w/ 32 levels "Acura","Audi",..: 1 1 2 2 3 4 4 4 4 5 ...
 $ Model             : Factor w/ 93 levels "100","190E","240",..: 49 56 9 1 6 24 54 74 73 35 ...
 $ Type              : Factor w/ 6 levels "Compact","Large",..: 4 3 1 3 3 3 2 2 3 2 ...
 $ Min.Price         : num  12.9 29.2 25.9 30.8 23.7 14.2 19.9 22.6 26.3 33 ...
 $ Price             : num  15.9 33.9 29.1 37.7 30 15.7 20.8 23.7 26.3 34.7 ...
 $ Max.Price         : num  18.8 38.7 32.3 44.6 36.2 17.3 21.7 24.9 26.3 36.3 ...
 $ MPG.city          : int  25 18 20 19 22 22 19 16 19 16 ...
 $ MPG.highway       : int  31 25 26 26 30 31 28 25 27 25 ...
 $ AirBags           : Factor w/ 3 levels "Driver & Passenger",..: 3 1 2 1 2 2 2 2 2 2 ...
 $ DriveTrain        : Factor w/ 3 levels "4WD","Front",..: 2 2 2 2 3 2 2 3 2 2 ...
 $ Cylinders         : Factor w/ 6 levels "3","4","5","6",..: 2 4 4 4 2 2 4 4 4 5 ...
 $ EngineSize        : num  1.8 3.2 2.8 2.8 3.5 2.2 3.8 5.7 3.8 4.9 ...
 $ Horsepower        : int  140 200 172 172 208 110 170 180 170 200 ...
 $ RPM               : int  6300 5500 5500 5500 5700 5200 4800 4000 4800 4100 ...
 $ Rev.per.mile      : int  2890 2335 2280 2535 2545 2565 1570 1320 1690 1510 ...
 $ Man.trans.avail   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 1 1 1 ...
 $ Fuel.tank.capacity: num  13.2 18 16.9 21.1 21.1 16.4 18 23 18.8 18 ...
 $ Passengers        : int  5 5 5 6 4 6 6 6 5 6 ...
 $ Length            : int  177 195 180 193 186 189 200 216 198 206 ...
 $ Wheelbase         : int  102 115 102 106 109 105 111 116 108 114 ...
 $ Width             : int  68 71 67 70 69 69 74 78 73 73 ...
 $ Turn.circle       : int  37 38 37 37 39 41 42 45 41 43 ...
 $ Rear.seat.room    : num  26.5 30 28 31 27 28 30.5 30.5 26.5 35 ...
 $ Luggage.room      : int  11 15 14 17 13 16 17 21 14 18 ...
 $ Weight            : int  2705 3560 3375 3405 3640 2880 3470 4105 3495 3620 ...
 $ Origin            : Factor w/ 2 levels "USA","non-USA": 2 2 2 2 2 1 1 1 1 1 ...
 $ Make              : Factor w/ 93 levels "Acura Integra",..: 1 2 4 3 5 6 7 9 8 10 ...
```

```
NULL
> # Creating a data frame from the main data set.
> car_data<- data.frame(Cars93$AirBags, Cars93$Type)
> # Creating a table with the needed variables.
> car_data = table(Cars93$AirBags, Cars93$Type)
> print(car_data)

                    Compact Large Midsize Small Sporty Van
  Driver & Passenger      2     4       7     0      3   0
  Driver only             9     7      11     5      8   3
  None                    5     0       4    16      3   6
> # Performing the Chi-Square test.
> print(chisq.test(car_data))

        Pearson's Chi-squared test

data:  car_data
X-squared = 33.001, df = 10, p-value = 0.0002723
```

b) t-test

```
x  <- c(0.593, 0.142, 0.329, 0.691, 0.231, 0.793, 0.519, 0.392, 0.418)
t.test(x, alternative="greater", mu=0.3)
```

output:

```
t.test(x, alternative="greater", mu=0.3)

        One Sample t-test

data:  x
t = 2.2051, df = 8, p-value = 0.02927
alternative hypothesis: true mean is greater than 0.3
95 percent confidence interval:
 0.3245133       Inf
sample estimates:
mean of x
0.4564444
```

c) F-test

```
install.packages("randomForest")
# Load the party package. It will automatically load other
# required packages.
library(party)

# Print some records from data set readingSkills.
print(head(readingSkills))
# Load the party package. It will automatically load other
# required packages.
library(party)
library(randomForest)
# Create the forest.
output.forest <- randomForest(nativeSpeaker ~ age + shoeSize + score,
                    data = readingSkills)

# View the forest results.
print(output.forest)
```

output:

```
print(output.forest)

Call:
 randomForest(formula = nativeSpeaker ~ age + shoeSize + score,      data = readingSkills)
           Type of random forest: classification
                 Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 1.5%
Confusion matrix:
    no yes class.error
no  99   1        0.01
yes  2  98        0.02
```

Viva Questions: