**Big Data Analytics-Lab-CSE6034\_Lab-Assessment - 4:**

**Implementation of all the different types of plotting applicable in R Programming**

**Submitted By: 20MAI0001 - NIHARIKA MAITRA**

**Git Repo link :**

**Q. 1) R Programming Code for Implementation of type of Plot : Bar charts**

**Significance / Application of type of Plot : Bar Charts :-**

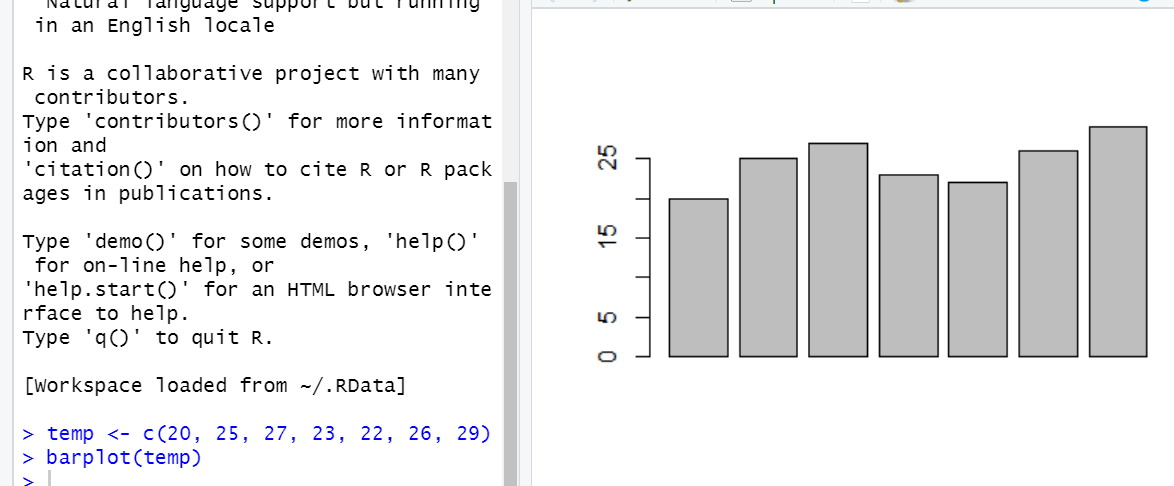
Bar Charts in R are the commonly used chart to create a graphical representation of the dataset. The Bar chart is represented as vertical or horizontal bars where the bar length or height indicates the count or frequency or any other calculated measure of the variable. As best practice a vector or a matrix can be used as input to the bar chart creation function in R for plotting bar charts. There are various labels and color assignment features are available with the bar plot function which is used to create the bar charts.

**Q. a) To create a simple Bar Chart in R**

**Solution a:-**

**>temp <- c(20, 25, 27, 23, 22, 26, 29)**

**>barplot(temp)**

****

**Q. b) Initializing some vector of numbers and creating a table () command to count them. The width of the bar can be adjusted using a parameter width () and space by space () in barplot**

**Solution b:-**

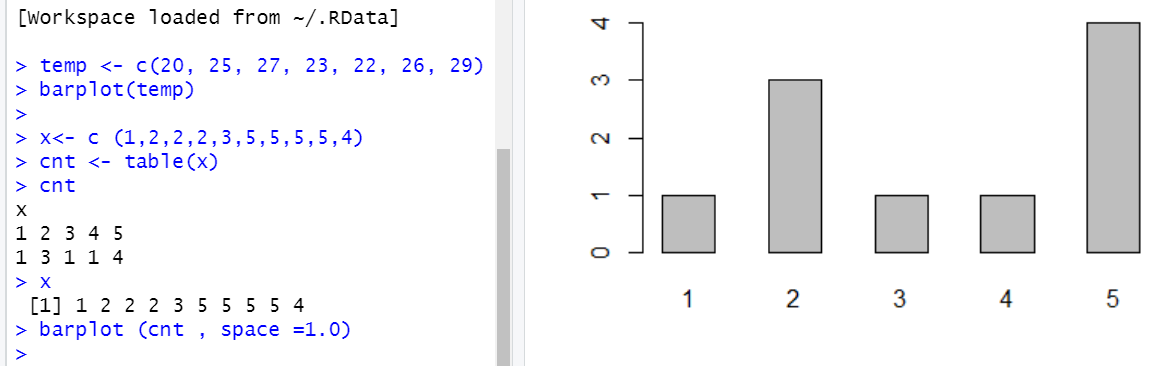
**>x<- c (1,2,2,2,3,5,5,5,5,4)**

**>cnt <- table(x)**

**>cnt**

**>x**

**>barplot (cnt , space =1.0)**

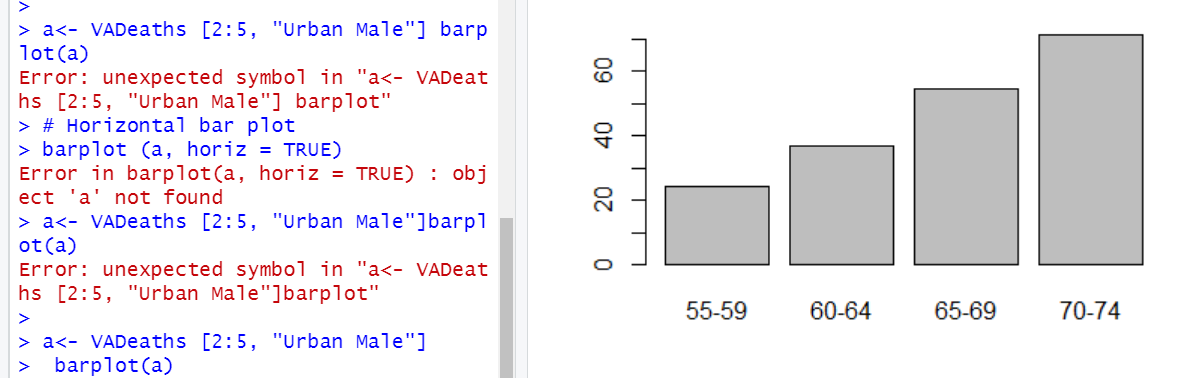
****

**Q. c) Creating a Bar chart using R built-in data set with Horizontal bar. To do so make horiz = TRUE or else vertical bars are drawn when horiz= FALSE (default option)**

**Solution c.1:-**

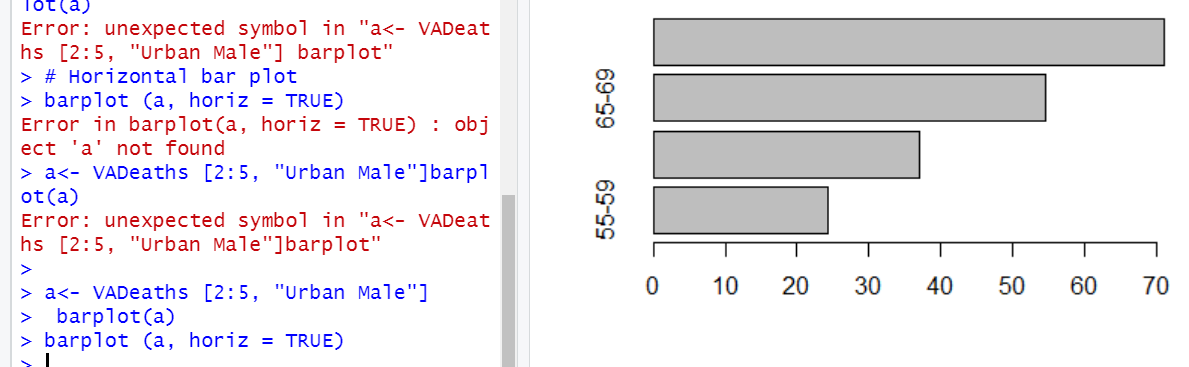
**> a<- VADeaths [2:5, "Urban Male"]**

**> barplot(a)**

****

**Solution c.2:-**

**> barplot (a, horiz = TRUE)**

****

### **Q. d) Creating a Bar Chart with Labels & Title**

**Solution d:-**

**> km <- c(11,14,14,16,17,19,17,16,17,18)**

**> table (km)**

**km**

**11 14 16 17 18 19**

**1 2 2 3 1 1**

**> km**

**[1] 11 14 14 16 17 19 17 16 17 18**

**> barplot(table(km),**

**+ main="km per distance",**

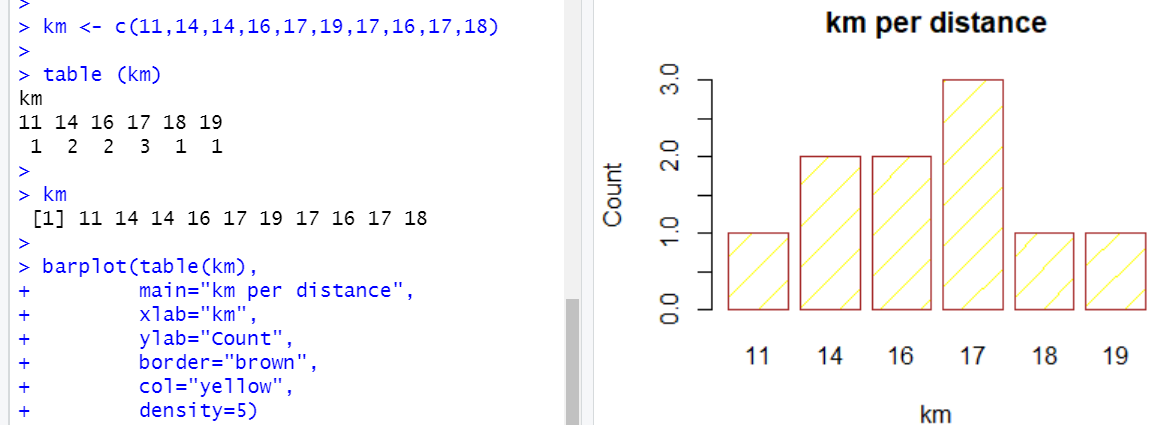
**+ xlab="km",**

**+ ylab="Count",**

**+ border="brown",**

**+ col="yellow",**

**+ density=5)**

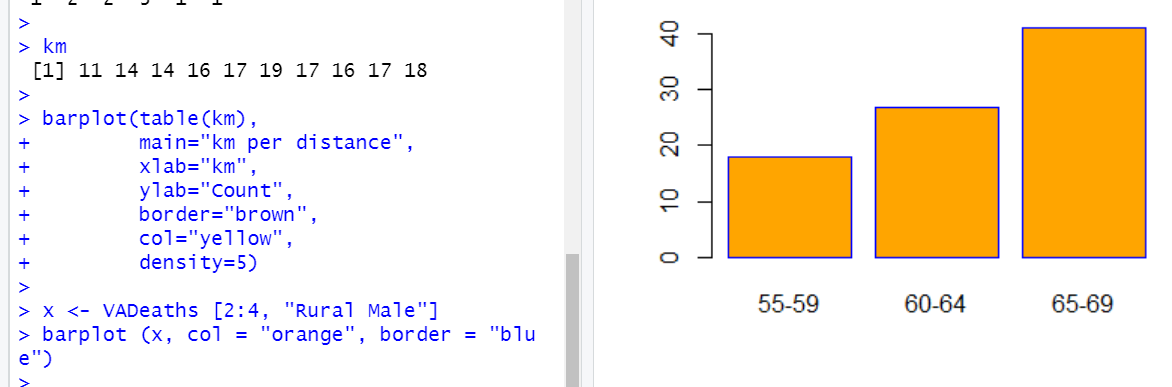
****

#### **Q. e) Assigning and changing colors**

**Solution e.1 :-**

**> x <- VADeaths [2:4, "Rural Male"]**

**> barplot (x, col = "orange", border = "blue")**

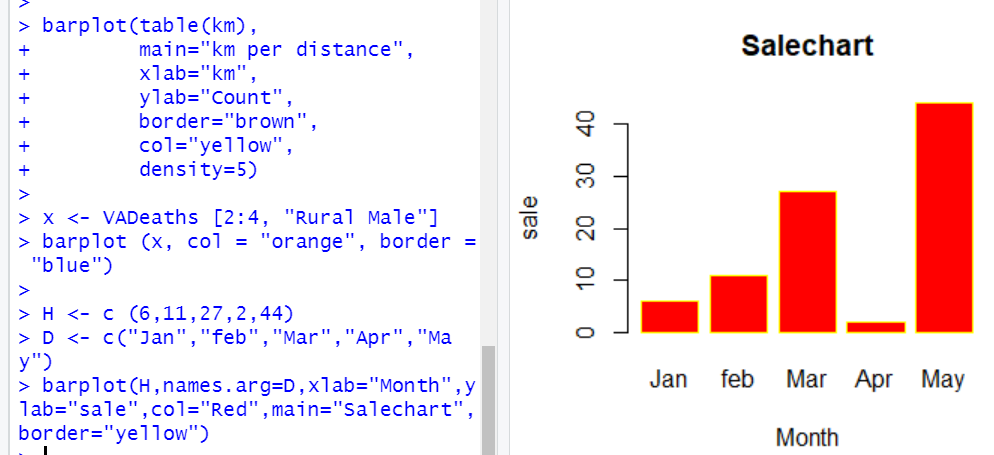
****

**Solution e.2 :-**

**> H <- c (6,11,27,2,44)**

**> D <- c("Jan","feb","Mar","Apr","May")**

**>barplot(H,names.arg=D,xlab="Month",ylab="sale",col="Red",main="Sale+chart",border="yellow")**

****

#### **Q. f) Using various Arguments**

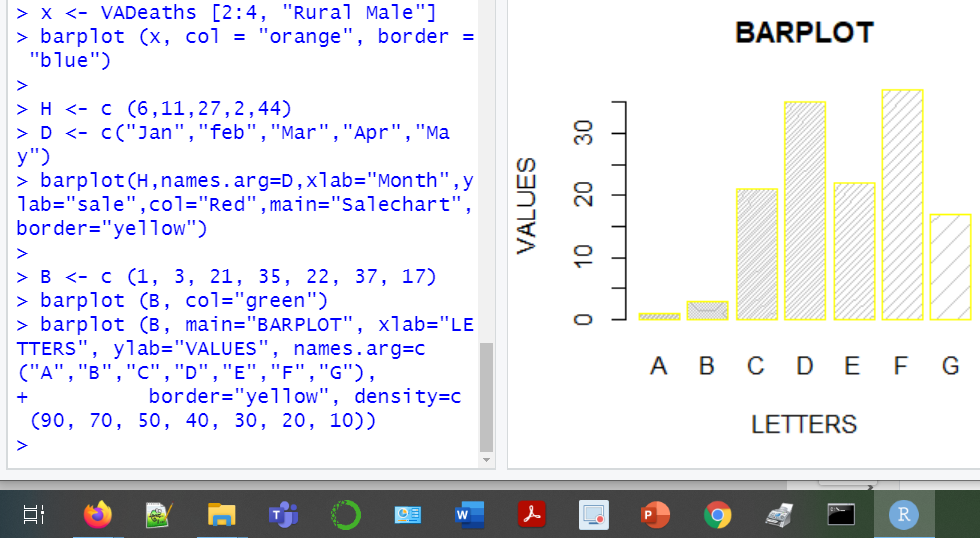
**Solution f :-**

**> B <- c (1, 3, 21, 35, 22, 37, 17)**

**> barplot (B, col="green")**

**> barplot (B, main="BARPLOT", xlab="LETTERS", ylab="VALUES", +names.arg=c("A","B","C","D","E","F","G"),**

**+ border="yellow", density=c (90, 70, 50, 40, 30, 20, 10))**

****

#### **Q. g) Using Matrix**

**Solution g :-**

**> mt <- c (3, 1, 10, 12, 14, 7, 9, 11, 18)**

**> val <- matrix (mt, nrow = 3, ncol = 3)**

**> val**

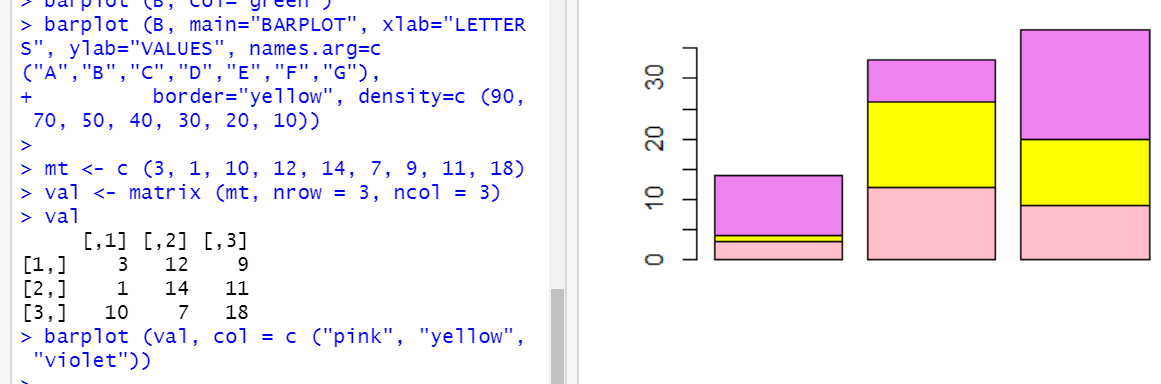
**[,1] [,2] [,3]**

**[1,] 3 12 9**

**[2,] 1 14 11**

**[3,] 10 7 18**

**> barplot (val, col = c ("pink", "yellow", "violet"))**

****

#### **Q. h) Multiple comparisons**

**Solution h:-**

**> A <- c (2,3,6,4,9)**

**> B <- c (3,5,3,4,11)**

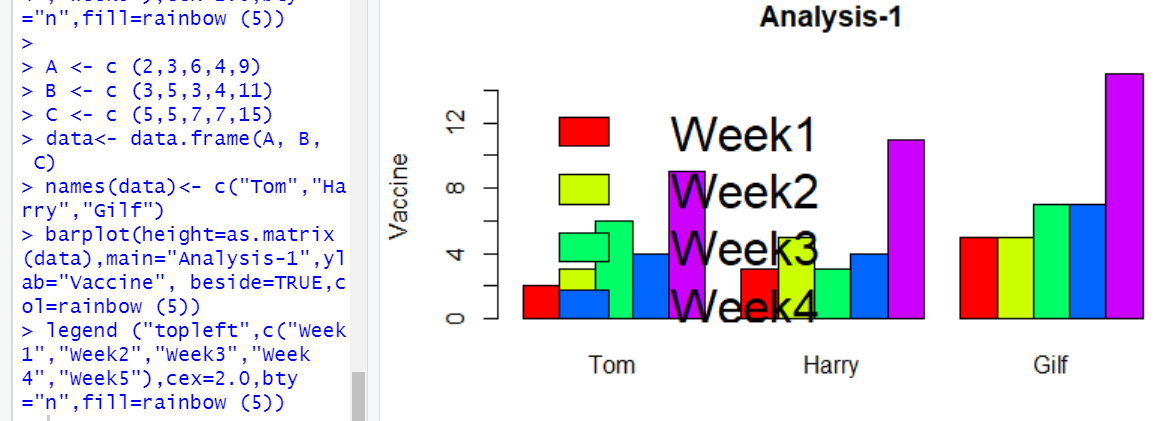
**> C <- c (5,5,7,7,15)**

**> data<- data.frame(A, B, C)**

**> names(data)<- c("Tom","Harry","Gilf")**

**> barplot(height=as.matrix(data),main="Analysis-1",ylab="Vaccine", +beside=TRUE,col=rainbow (5))**

**> legend +("topleft",c("Week1","Week2","Week3","Week4","Week5"),cex=2.0,bty="+n",fill=rainbow (5))**

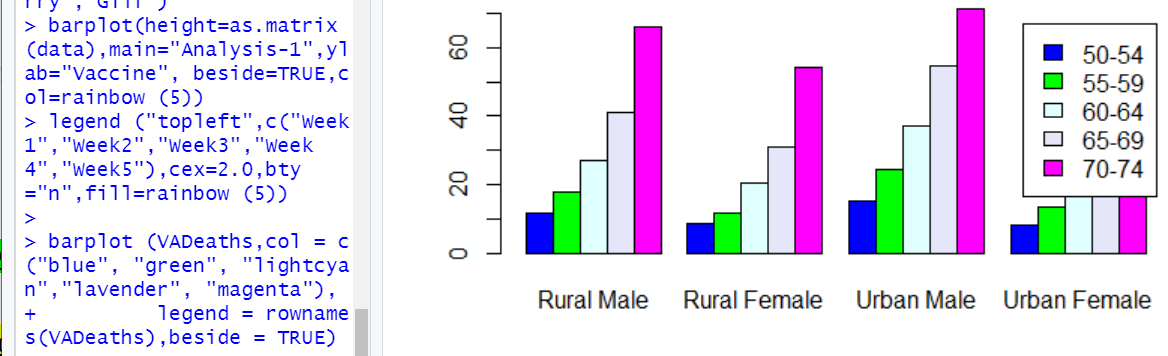
****

#### **Q. I) Grouped Bar Plots**

**Solution I.1:-**

**> barplot (VADeaths,col = c("blue", "green", "lightcyan","lavender", +"magenta"),**

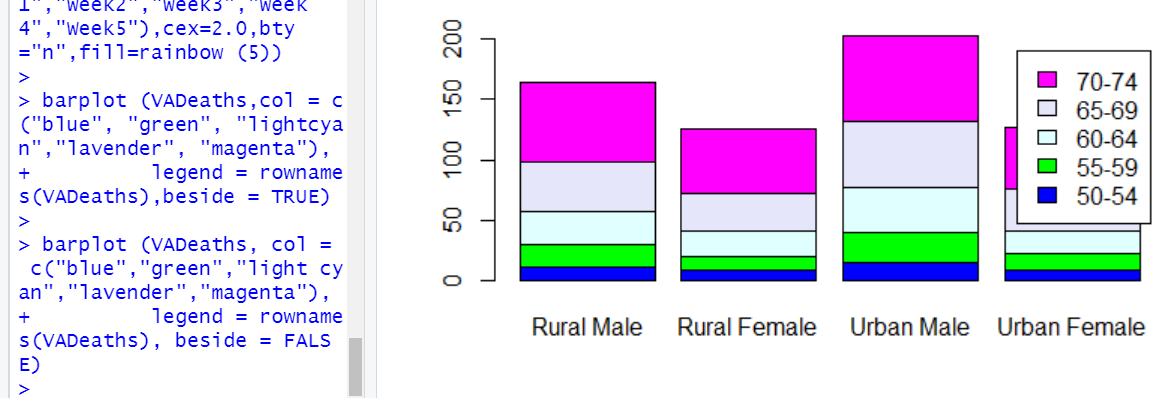
**+ legend = rownames(VADeaths),beside = TRUE)**

****

**Solution I.2:-**

**>barplot (VADeaths, col = c("blue","green","light +cyan","lavender","magenta"),**

**+legend = rownames(VADeaths), beside = FALSE)**

****

#### 

#### 

#### 

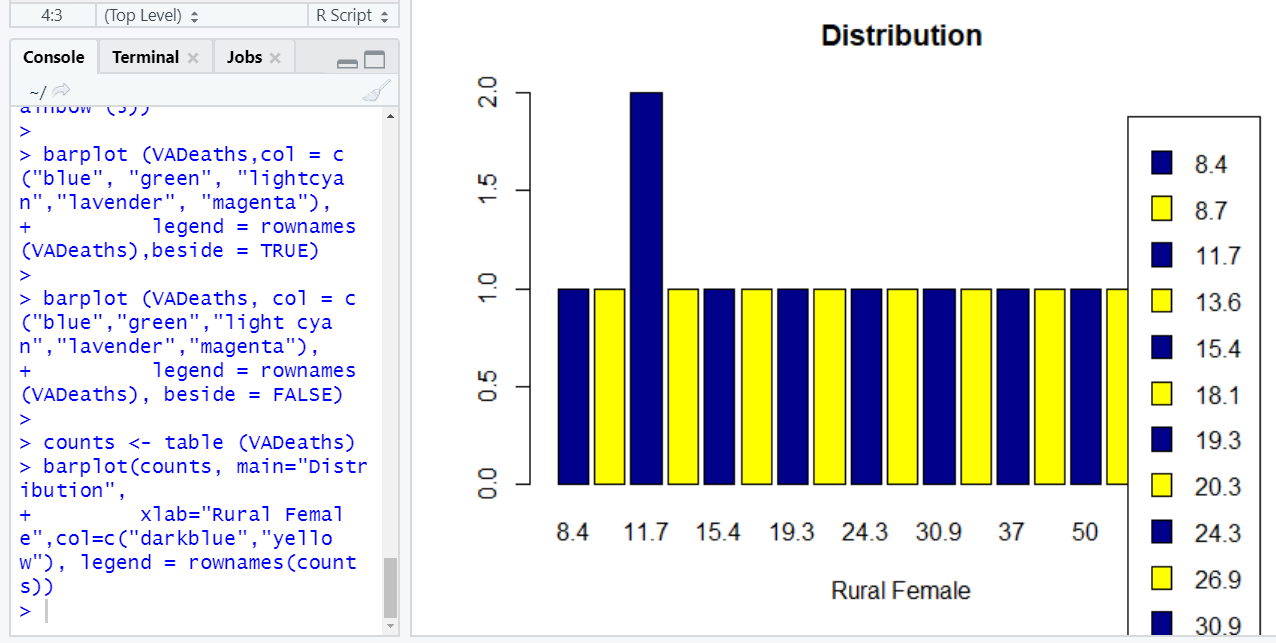
#### **Q. J) Stacked Bar Plot**

**Solution J :-**

**> counts <- table (VADeaths)**

**> barplot(counts, main="Distribution",**

**+ xlab="Rural Female",col=c("darkblue","yellow"), legend = +rownames(counts))**

****

**Q. 2) R Programming Code for Implementation of type of Plot : Pie charts**

**Significance / Application of type of Plot : Pie Charts :-**

Pie Chart in R is one of the basic chart features which are represented in the circular chart symbol. The section of the circle shows the data value proportions. The sections of the pie chart can be labeled with meaningful names. Pie charts are generally preferred for small size vector variables. Pie charts can be of two-dimensional view or three-dimensional views based upon the R packages. Pie is the function in R language which is supporting two-dimensional pie charts. Pie charts are very useful for data analysis. Pie charts in R can be assigned with a meaning title using main as a parameter in the pie function.Using the pie charts, patterns in the data can be understood easily whereas if we go through the numeric figure, often understanding takes a while. For example, if we plot the above example as a pie chart, we can understand the amount of production and proportion of production within a minute.

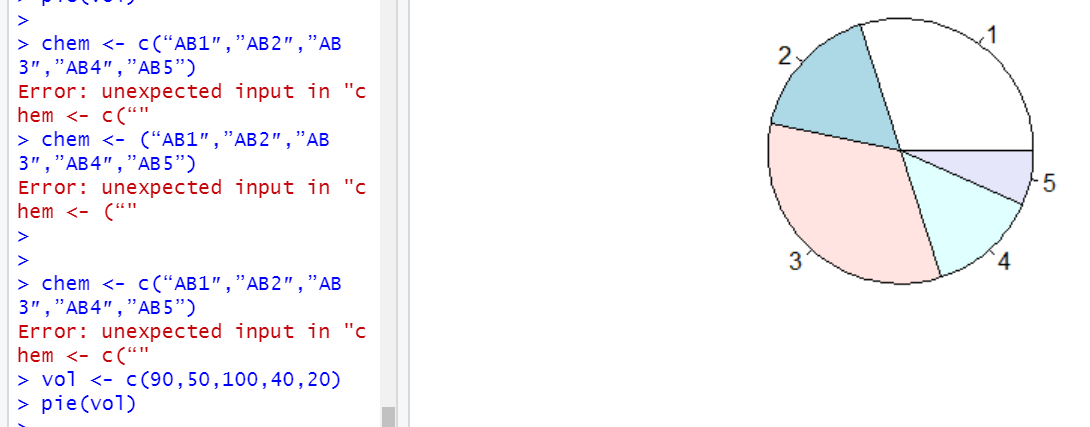
**Q. a) Plot a simple pie chart**

**Solution a.1 :-**

**>chem <- c(“AB1″,”AB2″,”AB3″,”AB4″,”AB5”)**

**>vol <- c(90,50,100,40,20)**

**>pie(vol)**

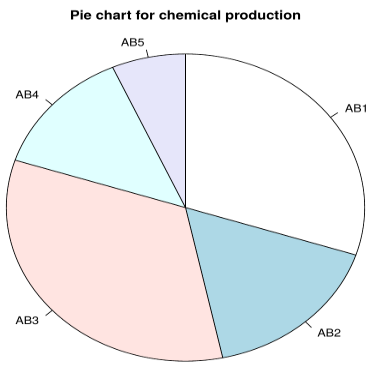
****

**Solution a.2 :-**

**>chem <- c(“AB1″,”AB2″,”AB3″,”AB4″,”AB5”)**

**>vol <- c(90,50,100,40,20)**

**>pie(x=vol, labels = chem, radius = 1,main = “Pie chart for chemical +production”, clockwise = T)**

****

### **Q. b) To change pie charts and fill color**

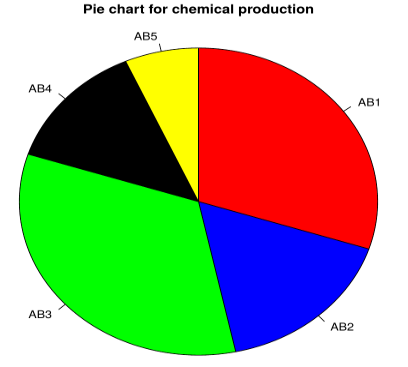
**Solution b :-**

**>pie(x=vol, labels = vol, radius = 1,main = “Pie chart for chemical +production”, clockwise = T)**

**>chem <- c(“AB1″,”AB2″,”AB3″,”AB4″,”AB5”)**

**>vol <- c(90,50,100,40,20)**

**>pie(x=vol, labels = chem, radius = 1,main = “Pie chart for chemical +production”, col=c(“red”,”blue”,”green”,”black”,”yellow”),clockwise = T)**

****

### **Q. c) To create a 3D pie chart**

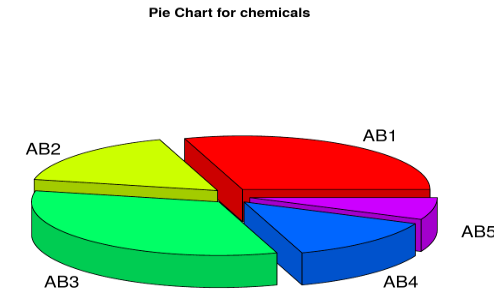
### **Solution c :-**

>chem <- c(“AB1″,”AB2″,”AB3″,”AB4″,”AB5”)

>vol <- c(90,50,100,40,20)

>library(plotrix)

>pie3D(vol,labels = chem,explode = 0.1, main = “Pie Chart for chemicals “)



**Q. 3) R Programming Code for Implementation of type of Plot : Histogram**

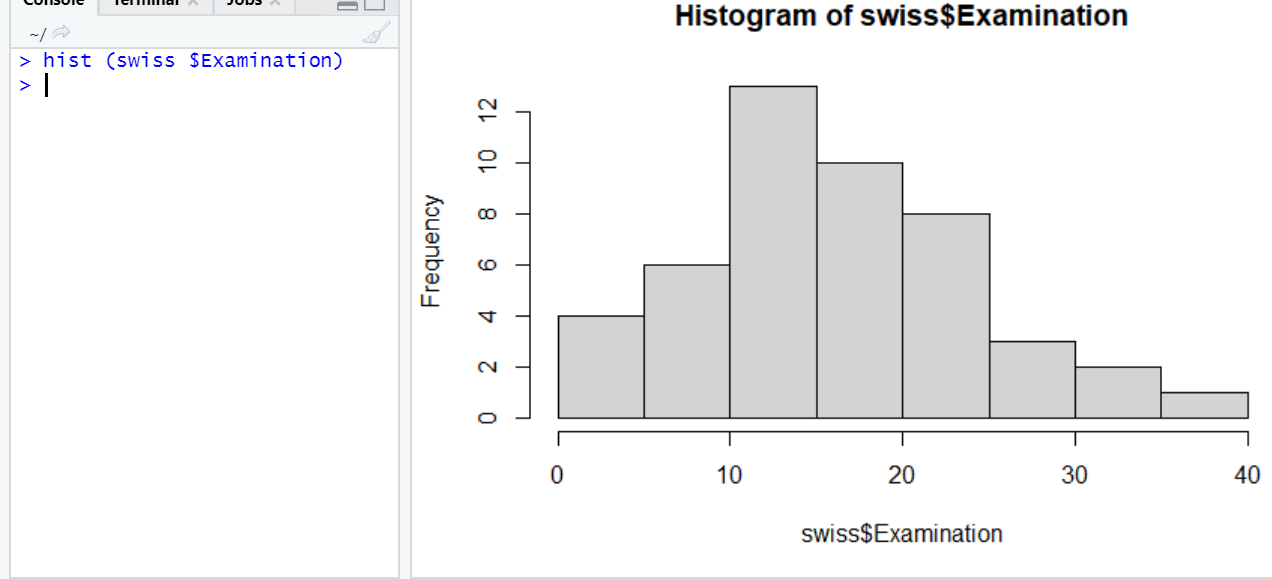
**Significance / Application of type of Plot : Histogram :-**

The histogram in R is one of the preferred plots for graphical data representation and data analysis. Histograms are generally viewed as vertical rectangles align in the two-dimensional axis which shows the data categories or groups comparison. The height of the bars or rectangular boxes shows the data counts in the y-axis and the data categories values are maintained in the x-axis. Histograms help in exploratory data analysis. The histogram in R can be created for a particular variable of the dataset which is useful for variable selection and feature engineering implementation in data science projects. R language supports out of the box packages to create histograms.The histogram is a pictorial representation of a dataset distribution with which we could easily analyze which factor has a higher amount of data and the least data. In other words, the histogram allows doing cumulative frequency plots in the x-axis and y-axis. Actually, histograms take both grouped and ungrouped data. For a grouped data histogram are constructed by considering class boundaries, whereas ungrouped data it is necessary to form the grouped frequency distribution. They help to analyze the range and location of the data effectively. Some common structure of histograms is applied like normal, skewed, cliff during data distribution.Histogram Takes continuous variable and splits into intervals it is necessary to choose the correct bin width. The major difference between the bar chart and histogram is the former uses nominal data sets to plot while histogram plots the continuous data sets. R uses hist () function to create histograms. This hist () function uses a vector of values to plot the histogram. Histogram comprises of an x-axis range of continuous values, y-axis plots frequent values of data in the x-axis with bars of variations of heights.

#### **Q. a) Create a Simple Histogram**

### **Solution a :-**

**>hist (swiss $Examination)**

****

#### **Q. b) Histogram with More Arguments**

### **Solution b :-**

**>hist (Air Passengers, xlim=c (150,600), ylim=c (0,35))**

**//In the above example x limit varies from 150 to 600 and Y – 0 to 35.**

**// Adding breaks**

**>hist (AirPassengers,**

**+main="Histogram with more Arg",**

**+xlab="Name List",**

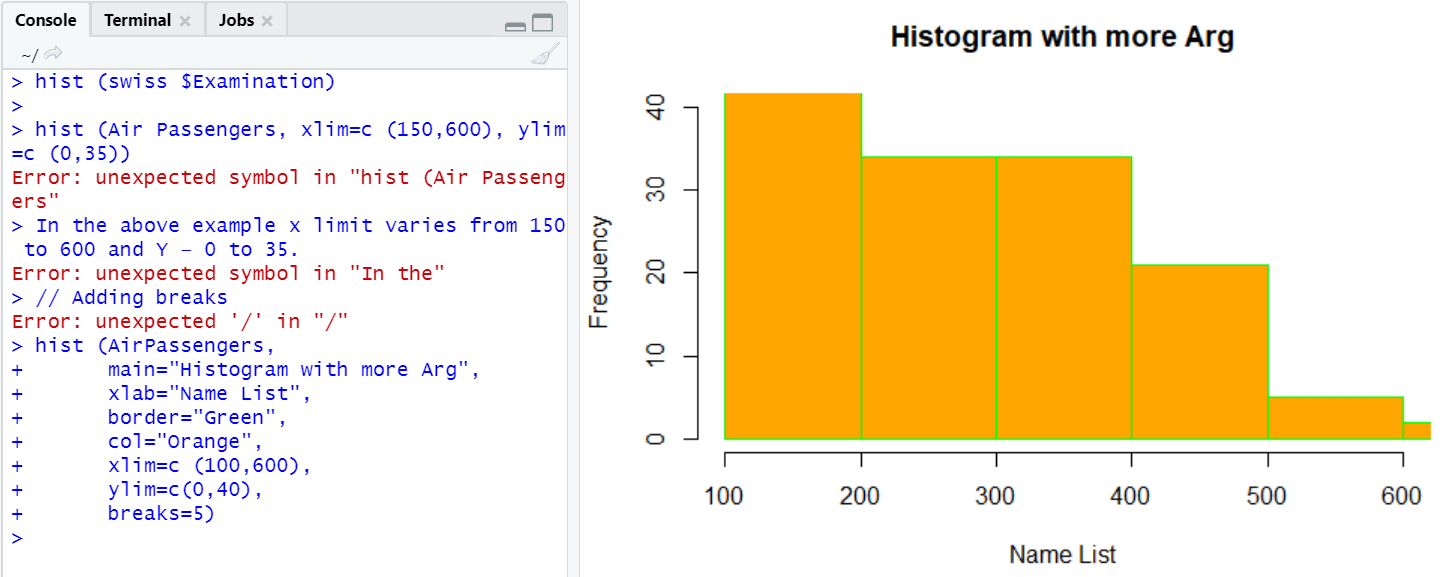
**+border="Green",**

**+col="Orange",**

**+xlim=c (100,600),**

**+ylim=c(0,40),**

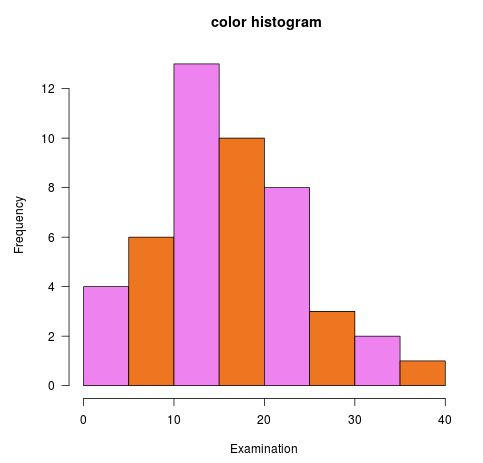
**+breaks=5)**

****

#### **Q. c.1) Adding Two Different Colors to the Bar**

### **Solution c.1 :-**

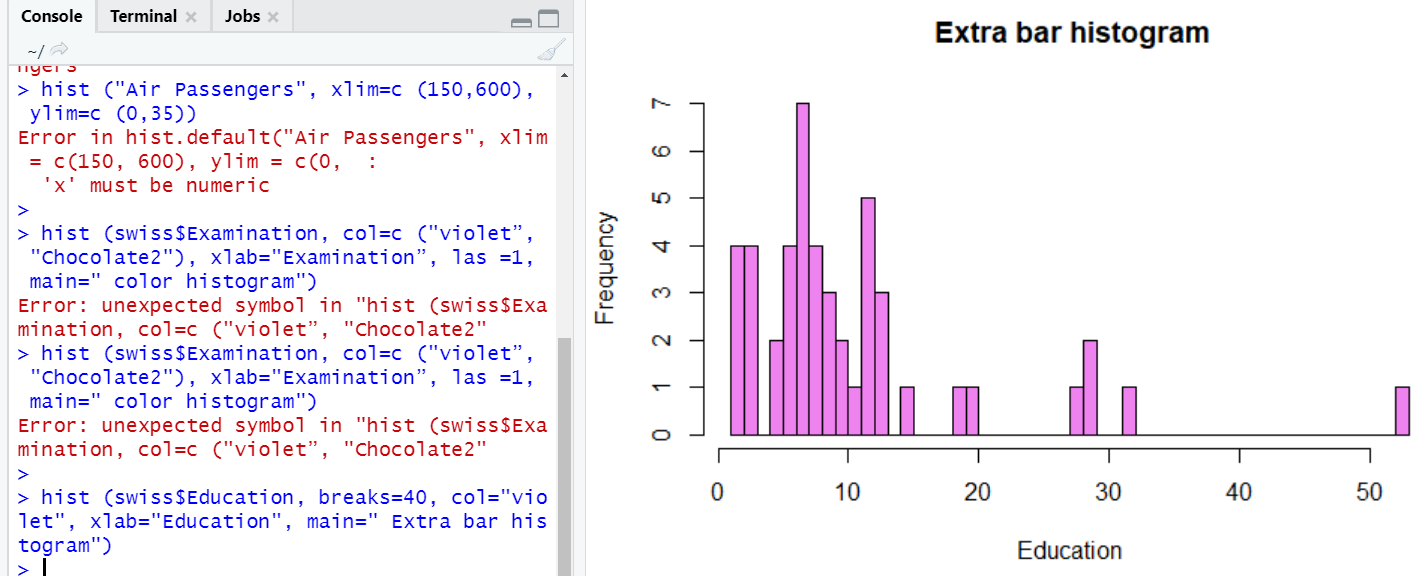
**>hist (swiss$Examination, col=c ("violet”, "Chocolate2"), +xlab="Examination”, las =1, main=" color histogram")**

****

#### **Q. c.2) Adding more bars to the histogram**

**Solution c.2 :-**

**>hist (swiss$Education, breaks=40, col="violet", xlab="Education", main=" +Extra bar histogram")**

****

#### **Q. d.1) Histogram in R Its Returns a Value**

#### **Solution d.1 :-**

**> Air <- AirPassengers**

**> hist (Air)**

**> h <- hist (Air)**

**> h**

**$breaks**

**[1] 100 150 200 250 300 350 400 450 500**

**[10] 550 600 650**

**$counts**

**[1] 24 24 21 13 21 13 13 8 4 1 2**

**$density**

**[1] 0.0033333333 0.0033333333**

**[3] 0.0029166667 0.0018055556**

**[5] 0.0029166667 0.0018055556**

**[7] 0.0018055556 0.0011111111**

**[9] 0.0005555556 0.0001388889**

**[11] 0.0002777778**

**$mids**

**[1] 125 175 225 275 325 375 425 475 525**

**[10] 575 625**

**$xname**

**[1] "Air"**

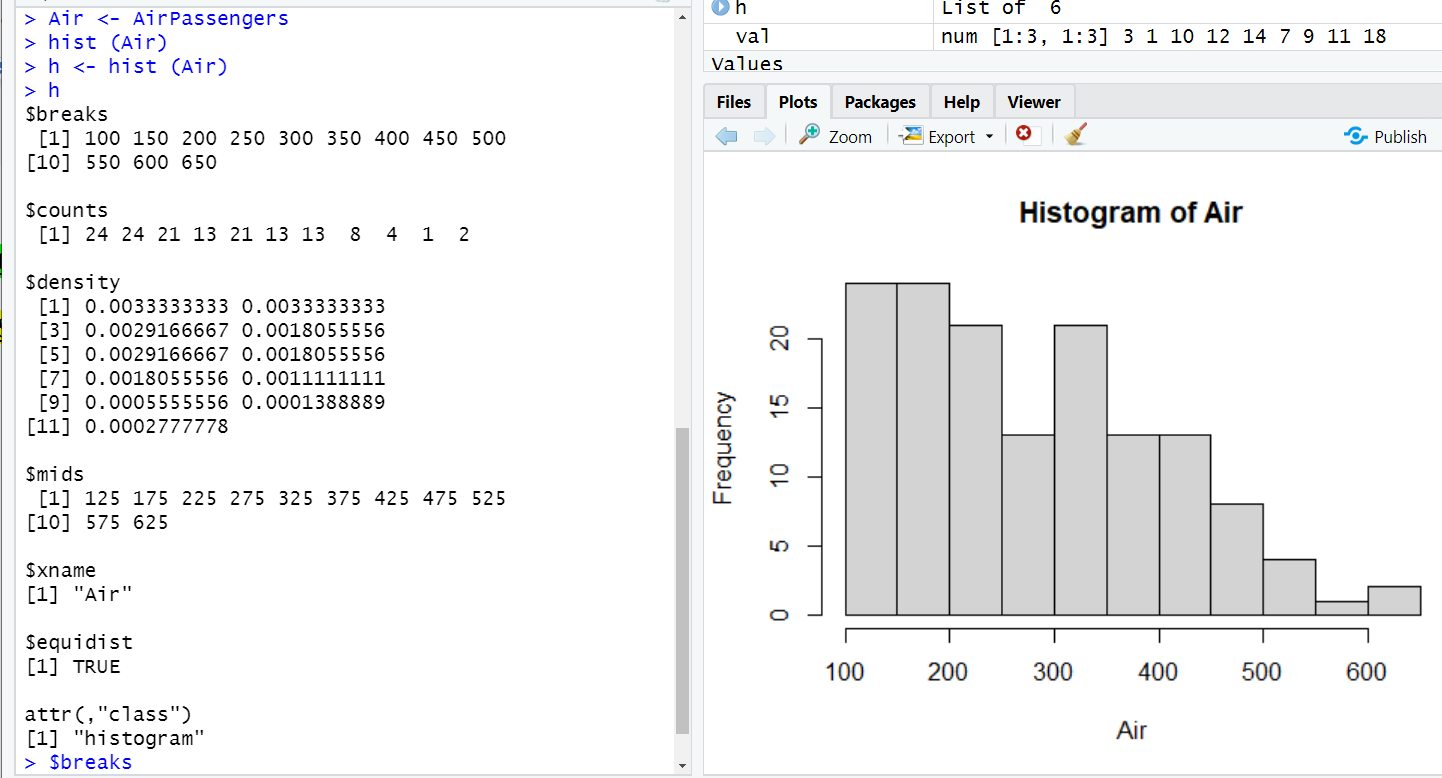
**$equidist**

**[1] TRUE**

**attr(,"class")**

**[1] "histogram"**

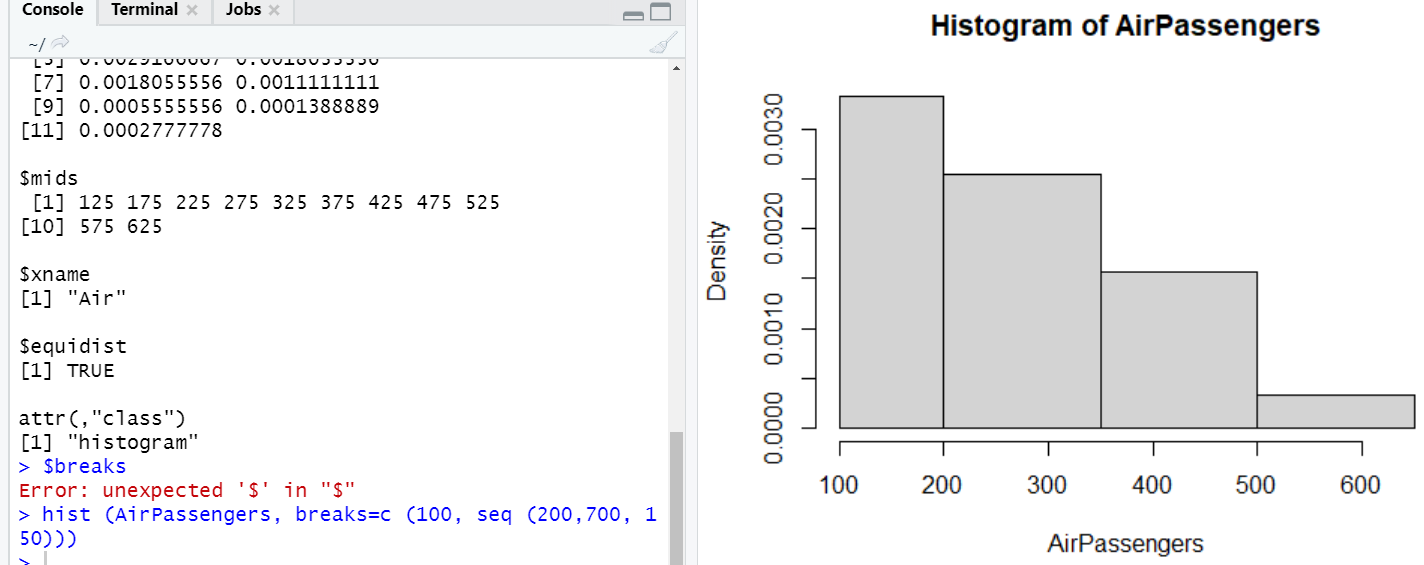
**> $breaks**

****

#### **Q. d.2) Using Break Argument to Change the Bin Width**

#### **Solution d.2 :-**

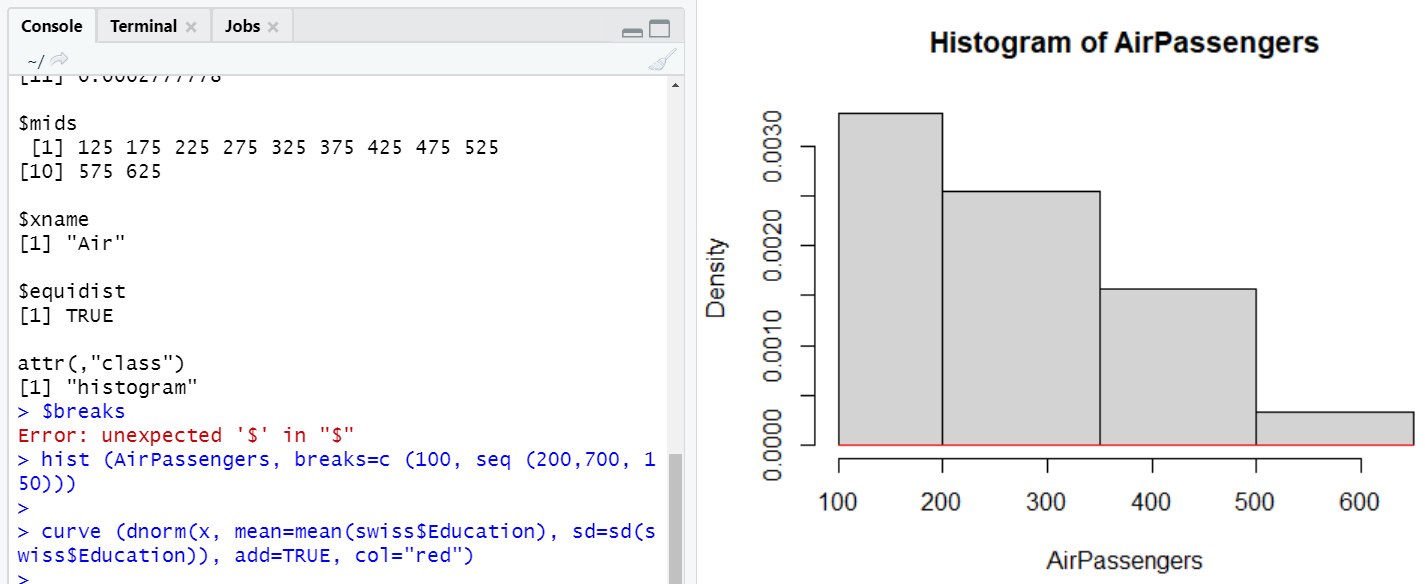
**>hist (AirPassengers, breaks=c (100, seq (200,700, 150)))**

****

#### **Q. e) Implementing the Normal Distribution Curve in Histogram**

#### **Solution e :-**

**>curve (dnorm(x, mean=mean(swiss$Education), sd=sd(swiss$Education)), +add=TRUE, col="red")**

****

#### 

#### 

#### 

#### 

#### **Q. f.1) Plotting Probability Distribution**

#### **Solution f.1 :-**

**>hist (AirPassengers,**

**+main="Histogram ",**

**+xlab="Passengers",**

**+border="Yellow",**

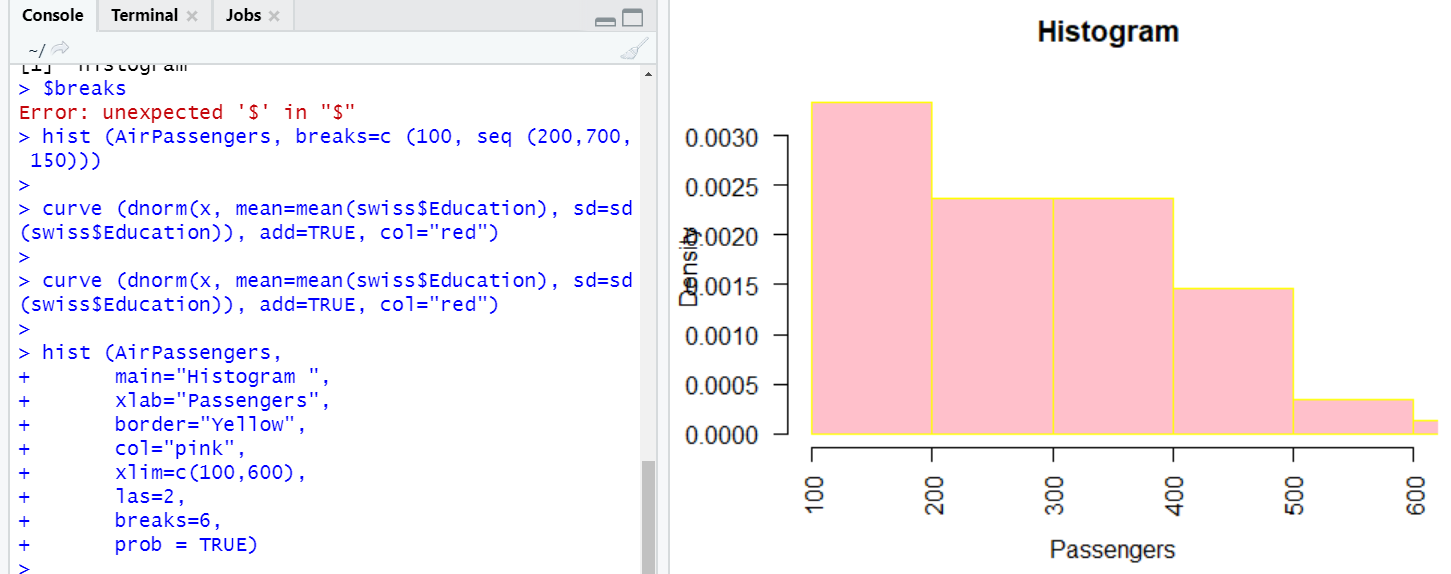
**+col="pink",**

**+xlim=c(100,600),**

**+las=2,**

**+breaks=6,**

**+prob = TRUE)**

****

#### **Q. f.2) Creating Density Plots in Histogram in R**

**Solution f.2 :-**

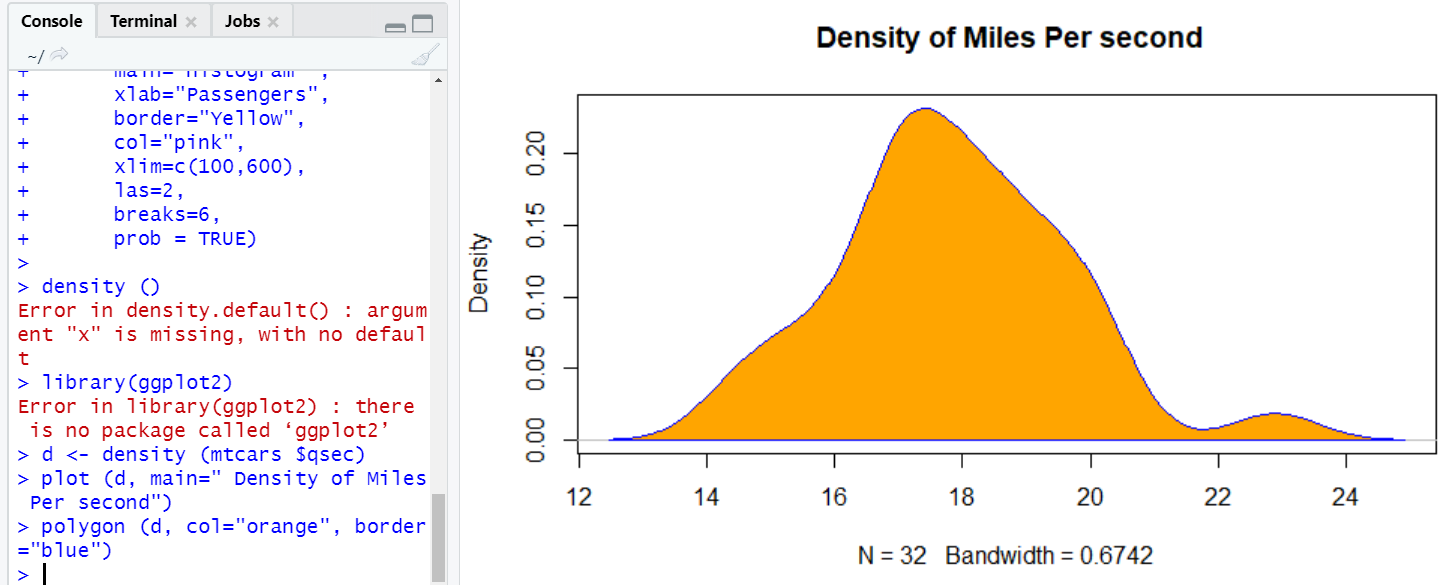
**>density () // this function returns the density of the data**

**>library(ggplot2)**

**>d <- density (mtcars $qsec)**

**>plot (d, main=" Density of Miles Per second")**

**>polygon (d, col="orange", border="blue")**

****

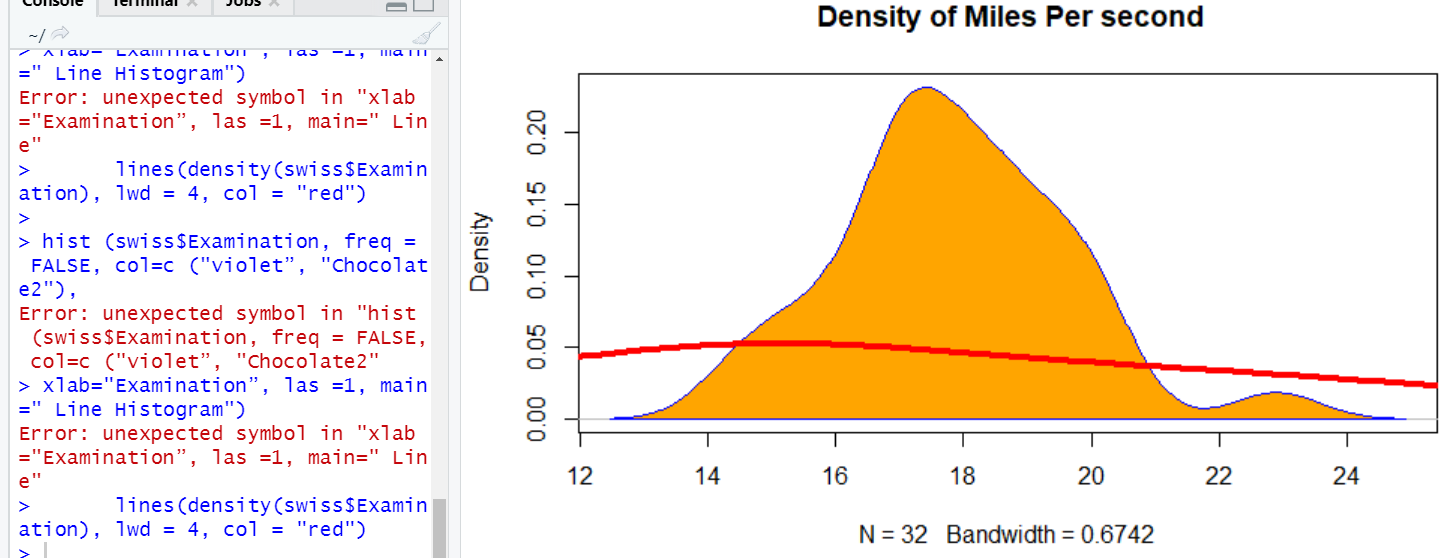
#### **Q. f.3) Using Line () function**

**Solution f.3 :-**

**>hist (swiss$Examination, freq = FALSE, col=c ("violet”, "Chocolate2"),**

**+xlab="Examination”, las =1, main=" Line Histogram")**

**+lines(density(swiss$Examination), lwd = 4, col = "red")**

****

**Q. 4) R Programming Code for Implementation of type of Plot : Line Graph**

**Significance / Application of type of Plot : Line Graph :-**

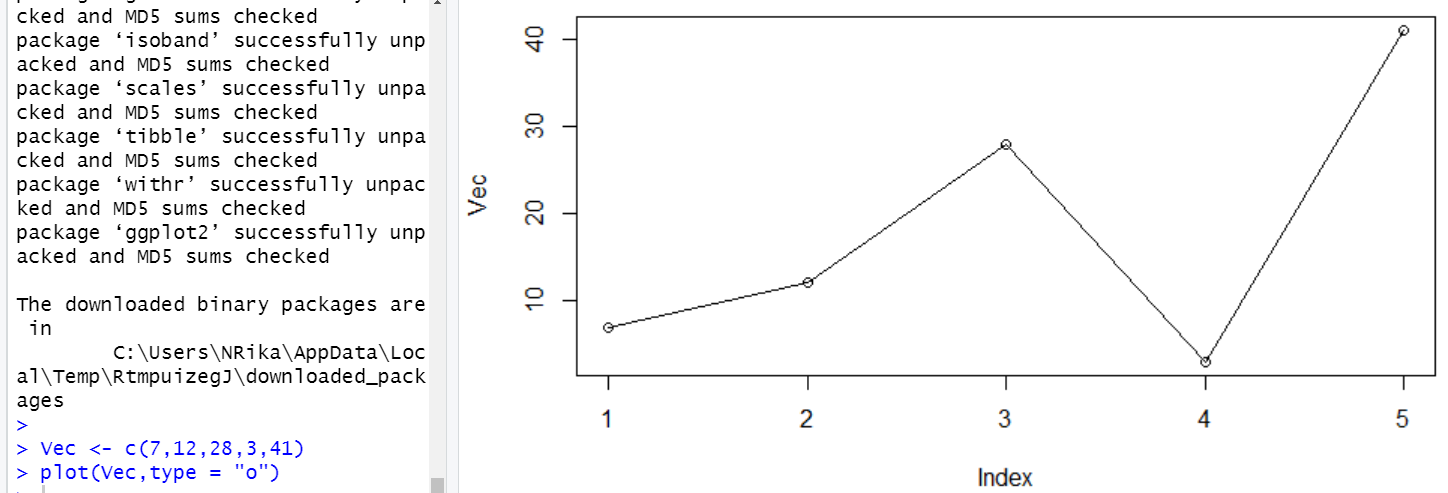
Line Graph in R is a basic chart in R language which forms lines by connecting the data points of the data set. Line charts can be used for exploratory data analysis to check the data trends by observing the line pattern of the line graph. Line Graph is plotted using plot function in the R language. The line graph can be associated with meaningful labels and titles using the function parameters. The line graphs can be colored using the color parameter to signify the multi-line graphs [for better graph representation](https://www.educba.com/graph-representation/). The line graphs in R are useful for time-series data analysis.A line graph is a basic yet very powerful chart to describe events over a certain time. R being a popular statistical tool, one must know how to plot line chart and how to customize its parameters to get the view as per one’s requirement. Once one gets comfortable with line graphs, other graphs should also be explored, to get a good grip [over data visualization](https://www.educba.com/what-is-data-visualization/).

#### **Q. a) Simple Line Graph in R code (with Plot function):**

**Solution a.1 :-**

**>Vec <- c(7,12,28,3,41) #Create the data for the chart**

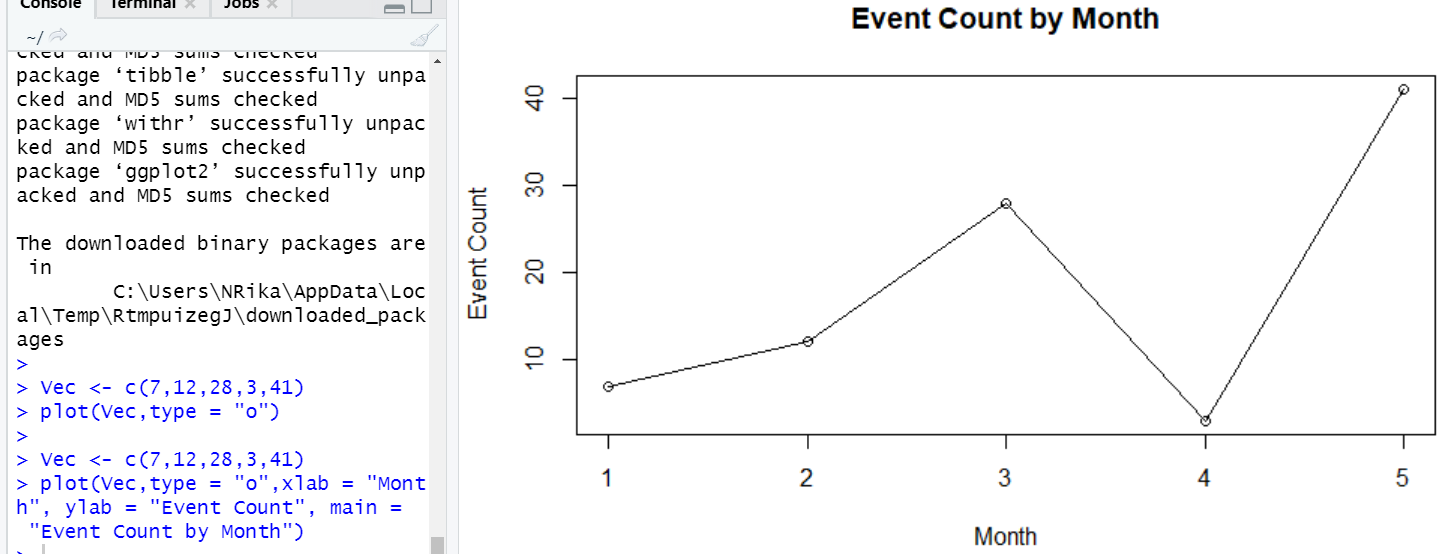
**>plot(Vec,type = "o") # Plot the bar chart**

****

**Solution a.2 :-**

**>Vec <- c(7,12,28,3,41) #Create the data for the chart.**

**>plot(Vec,type = "o",xlab = "Month", ylab = "Event Count", main = "Event +Count by Month")**

****

#### **Q. b) Multiple Lines in Line Chart**

#### **Solution b:-**

**> events1 <- c(7,12,28,3,41)**

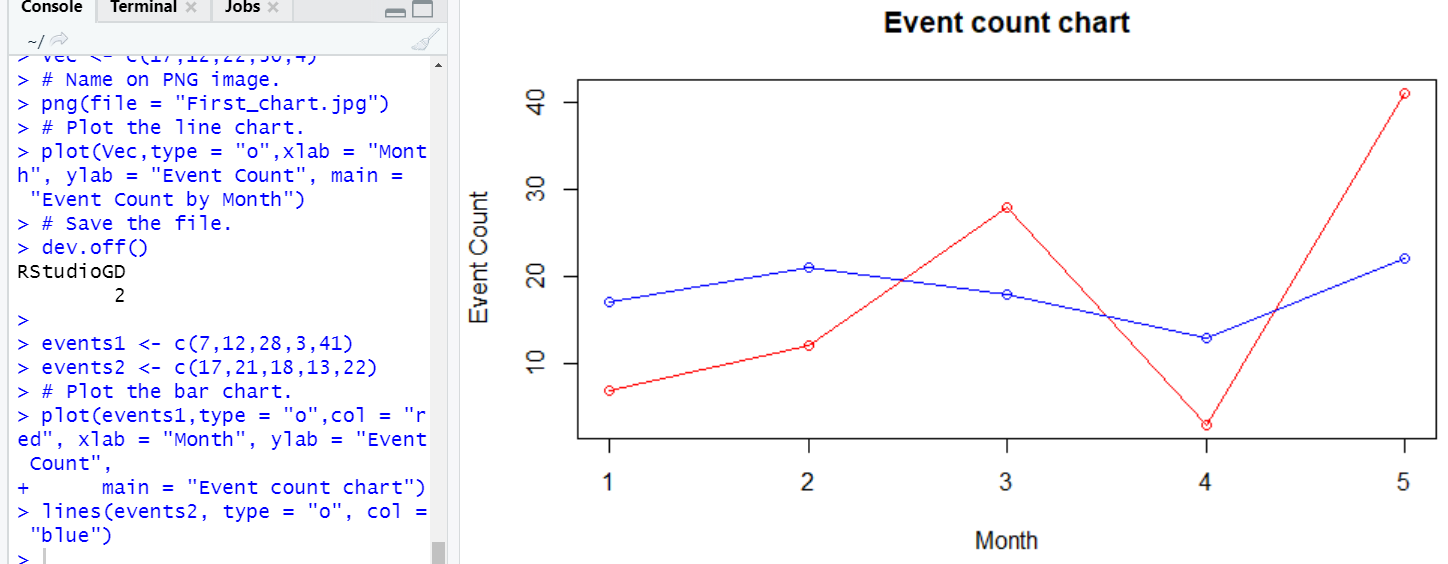
**> events2 <- c(17,21,18,13,22)**

**> # Plot the bar chart.**

**> plot(events1,type = "o",col = "red", xlab = "Month", ylab = "Event Count",**

**+ main = "Event count chart")**

**> lines(events2, type = "o", col = "blue")**

****

**Q. c) Add a legend to Line Graph**

#### **Solution c.1 :-**

**> events1 <- c(7,12,28,3,41)**

**> events2 <- c(17,21,18,13,22)**

**> # Plot the bar chart.**

**> plot(events1,type = "o",col = "red", xlab = "Month", ylab = "Event Count",**

**+ main = "Event count chart")**

**> lines(events2, type = "o", col = "blue")**

**>**

**> events1 <- c(7,12,28,3,41)**

**> events2 <- c(17,21,18,13,22)**

**> # Plot the bar chart.**

**> plot(events1,type = "o",col = "red", xlab = "Month", ylab = "Event Count",**

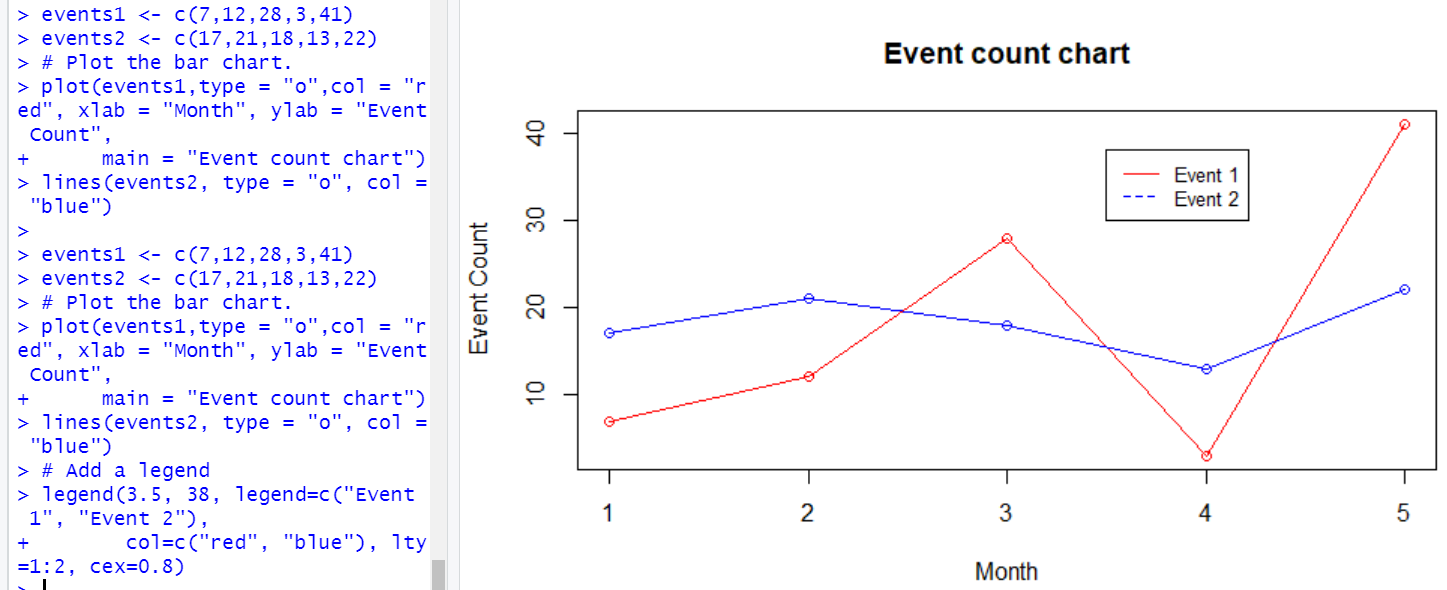
**+ main = "Event count chart")**

**> lines(events2, type = "o", col = "blue")**

**> # Add a legend**

**> legend(3.5, 38, legend=c("Event 1", "Event 2"),**

**+ col=c("red", "blue"), lty=1:2, cex=0.8)**

****

#### **Solution c.2 :-**

**> events1 <- c(7,12,28,3,41)**

**> events2 <- c(17,21,18,13,22)**

**> # Plot the bar chart.**

**> plot(events1,type = "o",col = "red", xlab = "Month", ylab = "Event Count",**

**+ main = "Event count chart")**

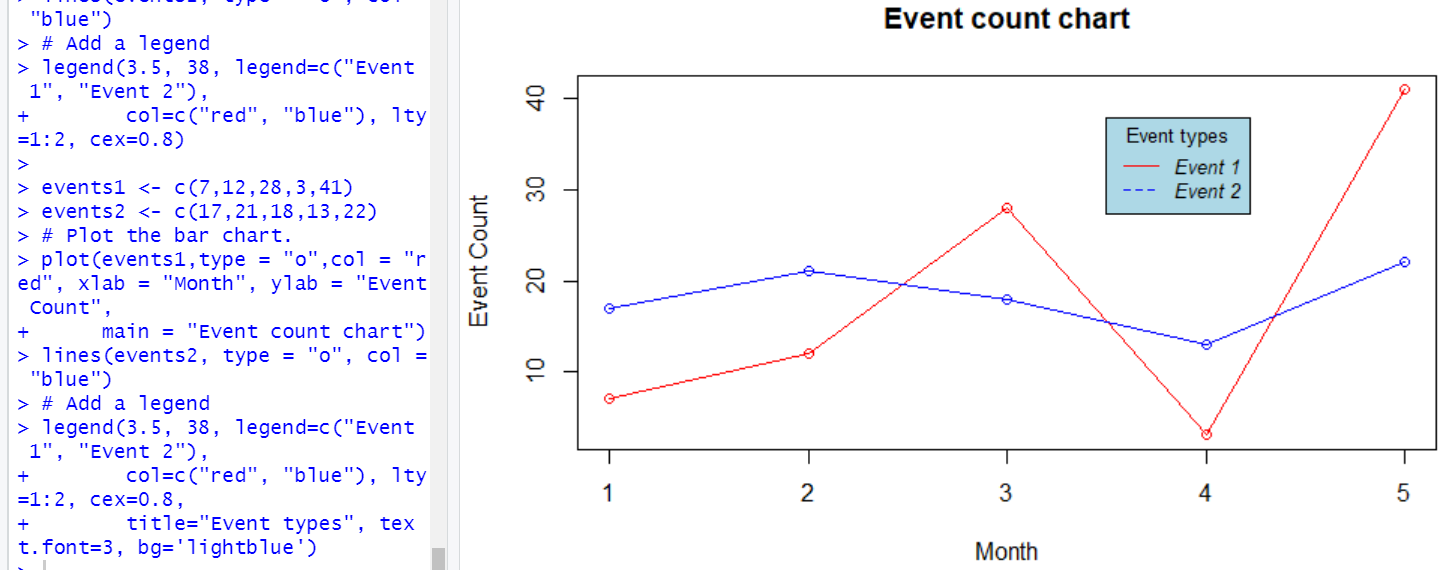
**> lines(events2, type = "o", col = "blue")**

**> # Add a legend**

**> legend(3.5, 38, legend=c("Event 1", "Event 2"),**

**+ col=c("red", "blue"), lty=1:2, cex=0.8,**

**+ title="Event types", text.font=3, bg='lightblue')**

****

#### **Q. d.1) Line Graph with ggplot2 library**

##### **Solution d.1 :-**

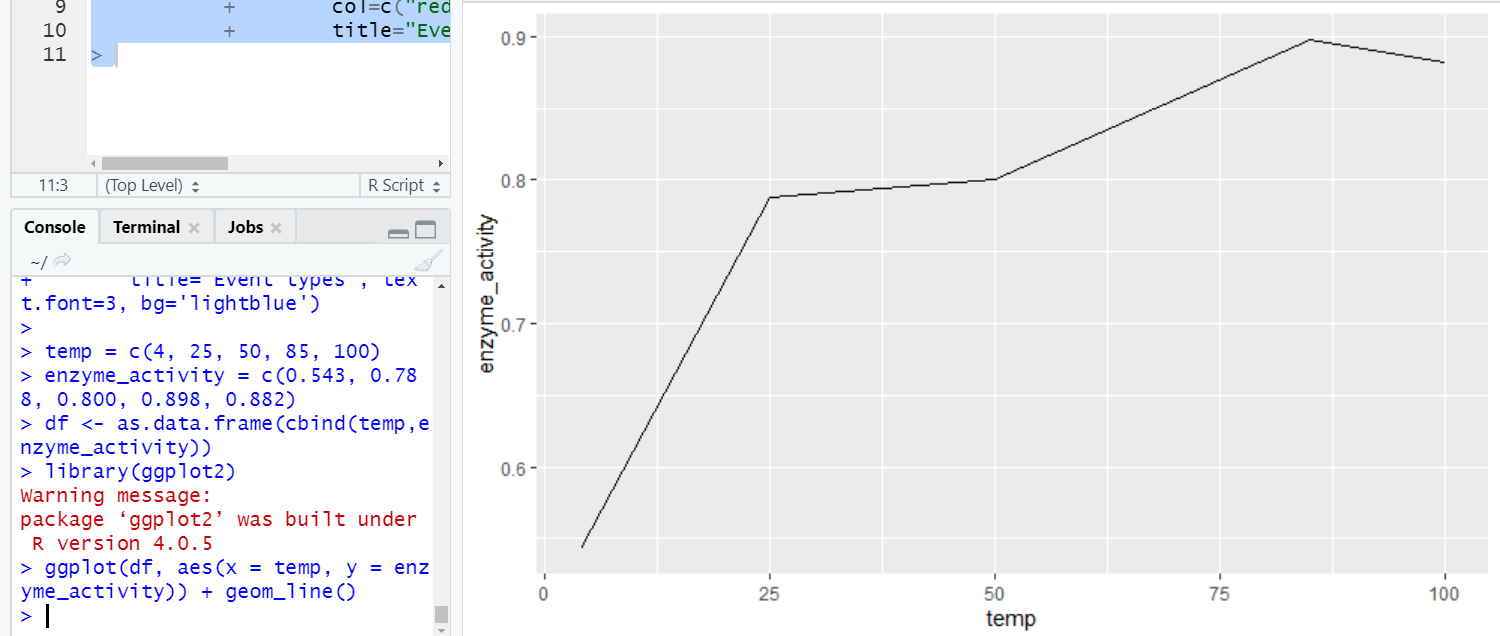
**>temp = c(4, 25, 50, 85, 100)**

**>enzyme\_activity = c(0.543, 0.788, 0.800, 0.898, 0.882)**

**>df <- as.data.frame(cbind(temp,enzyme\_activity))**

**>library(ggplot2)**

**>ggplot(df, aes(x = temp, y = enzyme\_activity)) + geom\_line()**



#### **Q. d.2) Multiple Lines in Line Graph with ggplot2 library**

##### **Solution d.2 :-**

**> library(ggplot2)**

**> temp = c(4, 25, 50, 85, 100)**

**> enzyme\_one\_activity = c(0.543, 0.788, 0.800, 0.898, 0.882)**

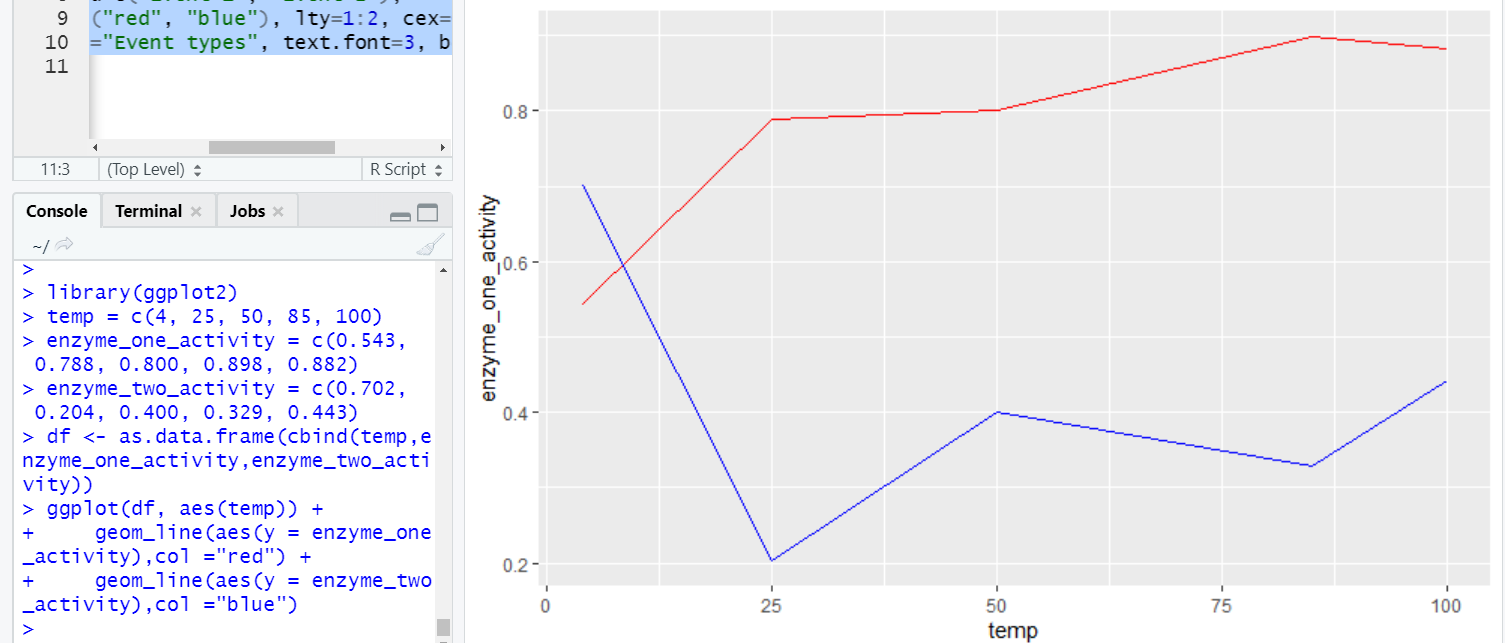
**> enzyme\_two\_activity = c(0.702, 0.204, 0.400, 0.329, 0.443)**

**> df <- as.data.frame(cbind(temp,enzyme\_one\_activity,enzyme\_two\_activity))**

**> ggplot(df, aes(temp)) +**

**+ geom\_line(aes(y = enzyme\_one\_activity),col ="red") +**

**+ geom\_line(aes(y = enzyme\_two\_activity),col ="blue")**

****

#### **Q. d.3) More Details to Graph in Line Graph with ggplot2 library**

##### **Solution d.3 :-**

**> library(ggplot2)**

**> temp = c(4, 25, 50, 85, 100)**

**> enzyme\_one\_activity = c(0.543, 0.788, 0.800, 0.898, 0.882)**

**> enzyme\_two\_activity = c(0.702, 0.204, 0.400, 0.329, 0.443)**

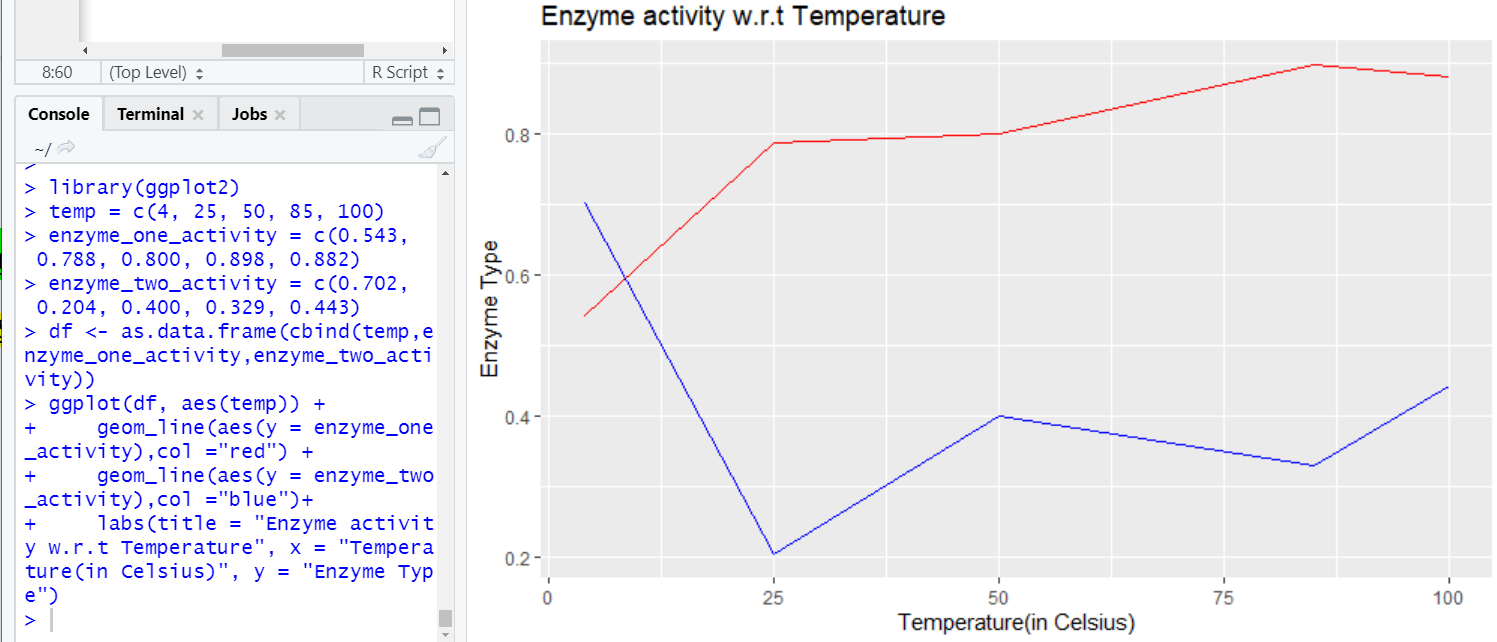
**> df <- as.data.frame(cbind(temp,enzyme\_one\_activity,enzyme\_two\_activity))**

**> ggplot(df, aes(temp)) +**

**+ geom\_line(aes(y = enzyme\_one\_activity),col ="red") +**

**+ geom\_line(aes(y = enzyme\_two\_activity),col ="blue")+**

**+ labs(title = "Enzyme activity w.r.t Temperature", x = "Temperature(in Celsius)", y = "Enzyme Type")**

****

**Q. 5) R Programming Code for Implementation of type of Plot : Scatter Plot**

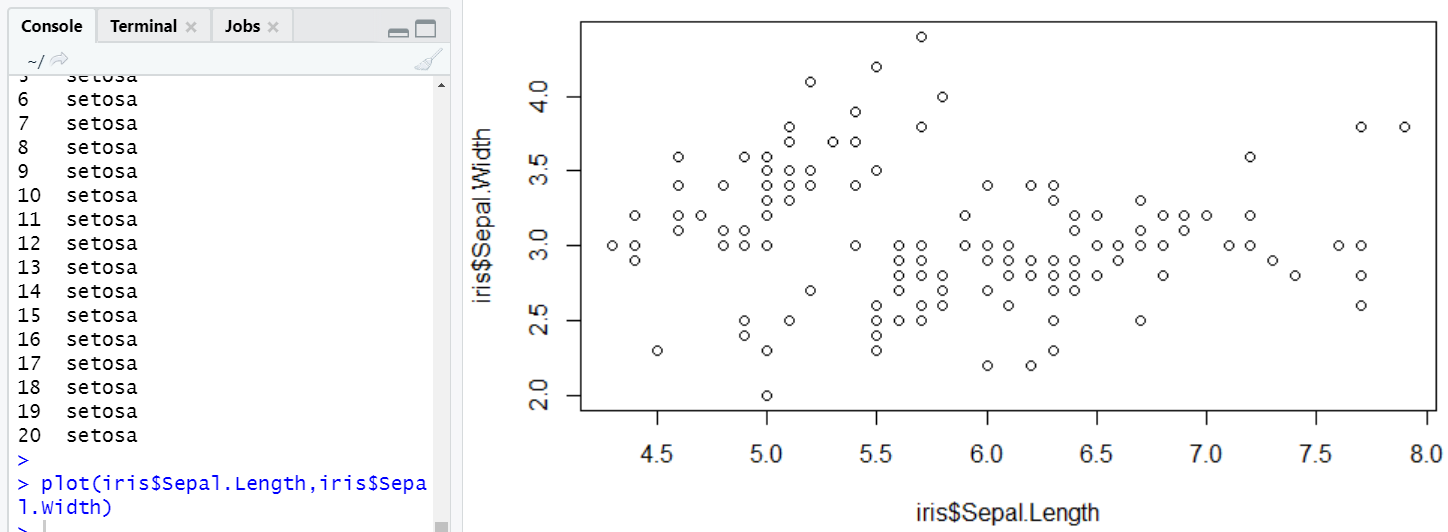
**Significance / Application of type of Plot : Scatter Plot :-**

A very important tool in an exploratory analysis, which is used to represent and analyze the relation between two variables in a dataset as a visual representation, in the form of an X-Y chart, with one variable acting as X-coordinate and another variable acting as Y-coordinate is termed as a scatterplot in R. R programming provides a very effective and robust mechanism being facilitated but not limited to functions such as plot(), with various functionalities in R providing options to improve visualization aesthetics.The scatter plot using plot() function provides basic features of representation, however, implementation of the ggplot2 package provides additional representation features like advance color grouping and various symbols type to the scatter plot. The scatter plot in R can be added with more meaningful levels and colors for better presentation.

#### **Q. a.1) To create a scatterplot with sepal. Length and sepal.Width variables using plot() function in R programming**

### **Solution a.1 :-**

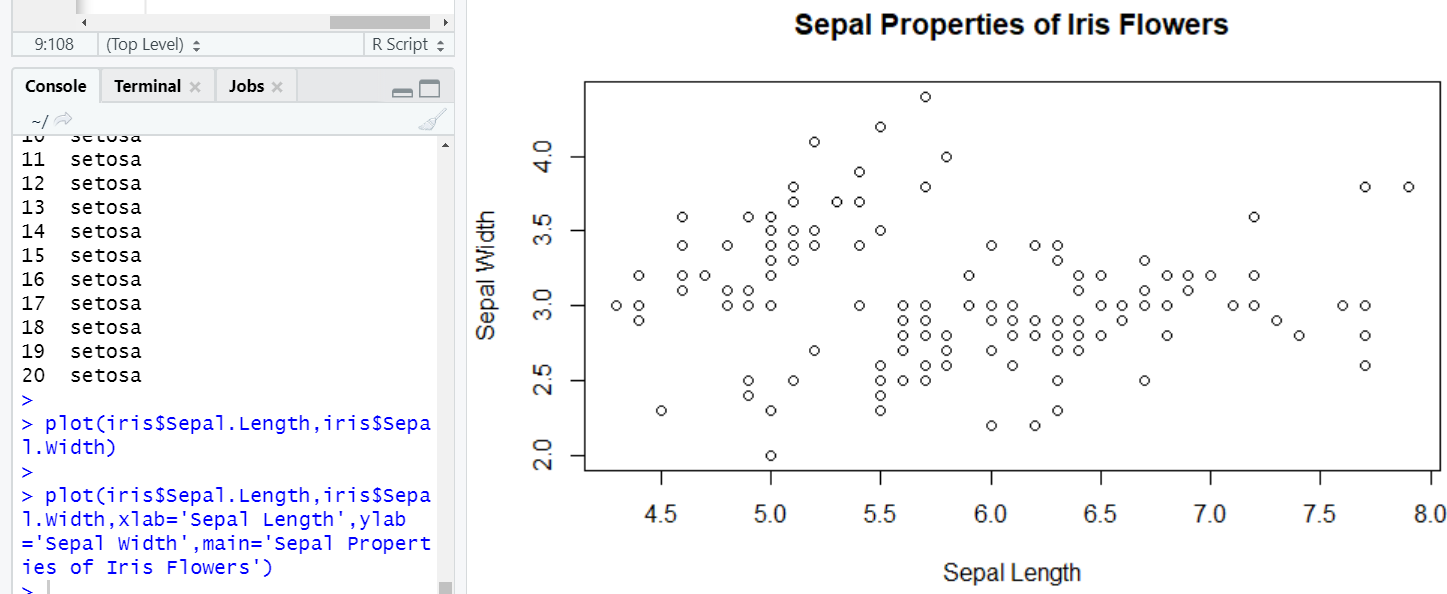
**>plot(iris$Sepal.Length,iris$Sepal.Width)**

****

**Q. a.2) To apply more parameters to the plot function to improve the scatter plot representation**

### **Solution a.2 :-**

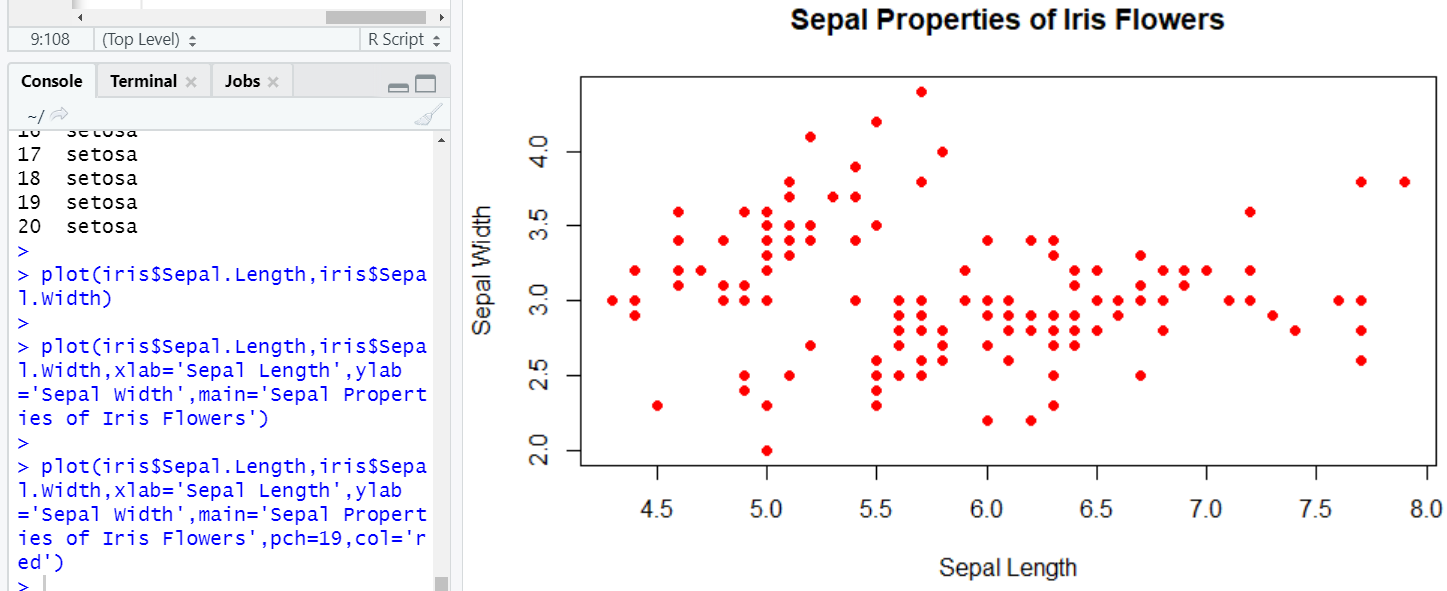
**>plot(iris$Sepal.Length,iris$Sepal.Width,xlab='Sepal Length',ylab='Sepal +Width',main='Sepal Properties of Iris Flowers')**

****

**Q. a.3) To apply further enhancements to the scatter plot by adding color and shapes to the scatter points**

### **Solution a.3 :-**

**>plot(iris$Sepal.Length,iris$Sepal.Width,xlab='Sepal Length',ylab='Sepal +Width',main='Sepal Properties of Iris Flowers',pch=19,col='red')**

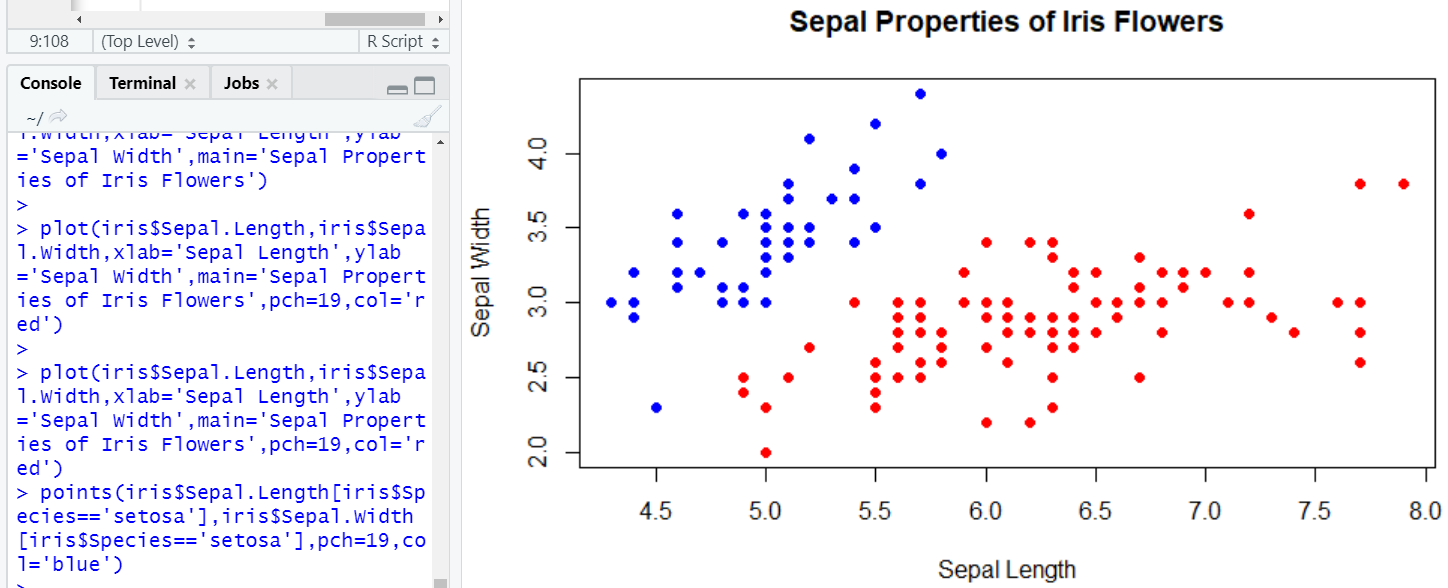
****

**Q. a.4) To apply points() function to segregate the color for setosa category of iris species and changing the color to blue**

### **Solution a.4 :-**

**>plot(iris$Sepal.Length,iris$Sepal.Width,xlab='Sepal Length',ylab='Sepal +Width',main='Sepal Properties of Iris Flowers',pch=19,col='red')**

**+points(iris$Sepal.Length[iris$Species=='setosa'],iris$Sepal.Width[iris$Species=='setosa'],pch=19,col='blue')**

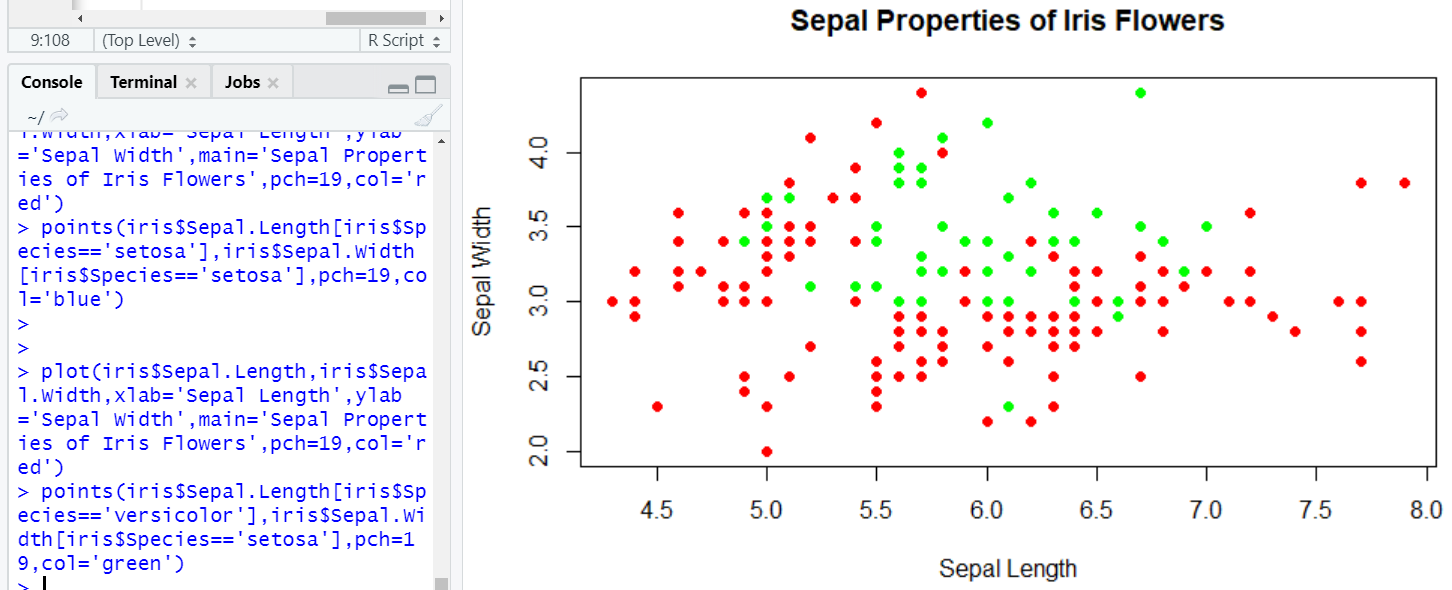
****

**Q. a.5) To apply green color to the Versicolor species category using another point () function**

### **Solution a.5 :-**

**>plot(iris$Sepal.Length,iris$Sepal.Width,xlab='Sepal Length',ylab='Sepal +Width',main='Sepal Properties of Iris Flowers',pch=19,col='red')**

**+points(iris$Sepal.Length[iris$Species=='versicolor'],iris$Sepal.Width[iris$Species=='setosa'],pch=19,col='green')**

****

**Q. 6) R Programming Code for Implementation of type of Plot : Box Plot**

**Significance / Application of type of Plot : Box Plot :-**

Labels are used in box plots which help to represent the data distribution based upon the mean, median and variance of the data set. R boxplot labels are generally assigned to the x-axis and y-axis of the boxplot diagram to add more meaning to the boxplot. The boxplot displays the minimum and the maximum value at the start and end of the boxplot. The mean label is represented in the center of the boxplot and it also shows the first and third quartile labels associating with the mean position.

Plotting the boxplot graph

* We need five valued inputs like mean, variance, median, first and third quartile.
* Identifying if there are any outliers in the data.
* Design the model to plot the data

The data grouping is made easy with the help of boxplots. Box plot supports multiple variables as well as various optimizations. We can also vary the scales according to data.

Boxplots can be used to compare various data variables or sets.The usability of the boxplot is easy and convenient. We need consistent data and proper labels. Boxplots are often used in data science and even by sales teams to group and compare data. Boxplot gives insights on the potential of the data and optimizations that can be done to increase sales.

#### **Q. a.1) To create a Simple Box plot in R programming**

### **Solution a.1 :-**

**> data<-data.frame(Stat1=rnorm(10,mean=3,sd=2))**

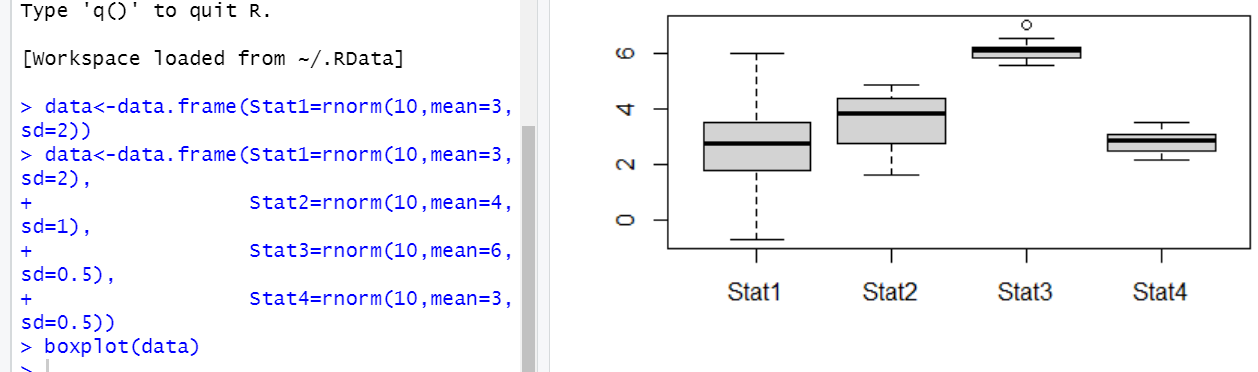
**> data<-data.frame(Stat1=rnorm(10,mean=3,sd=2),**

**+ Stat2=rnorm(10,mean=4,sd=1),**

**+ Stat3=rnorm(10,mean=6,sd=0.5),**

**+ Stat4=rnorm(10,mean=3,sd=0.5))**

**> boxplot(data)**

****

#### **Q. a.2) To to change the colour in the plot and add the parameter col = color in the boxplot() function**

### **Solution a.2 :-**

### **> data<-data.frame(Stat1=rnorm(10,mean=3,sd=2),**

**+ Stat2=rnorm(10,mean=4,sd=1),**

**+ Stat3=rnorm(10,mean=6,sd=0.5),**

**+ Stat4=rnorm(10,mean=3,sd=0.5))**

**> boxplot(data,las=2,col="red")**

**> data**

**Stat1 Stat2 Stat3**

**1 4.2184697 3.788716 5.915464**

**2 1.2683999 2.497891 5.165880**

**3 4.5133350 3.414957 5.418080**

**4 3.0520668 4.117741 6.048414**

**5 3.5504336 2.821499 6.205525**

**6 4.4076432 3.519117 5.456556**

**7 0.5500942 3.568465 6.534382**

**8 0.9173630 5.045526 6.223382**

**9 7.0853534 5.414540 5.514115**

**10 -0.4571851 3.489960 6.289704**

**Stat4**

**1 2.968653**

**2 2.643432**

**3 3.167392**

**4 2.162570**

**5 2.491187**

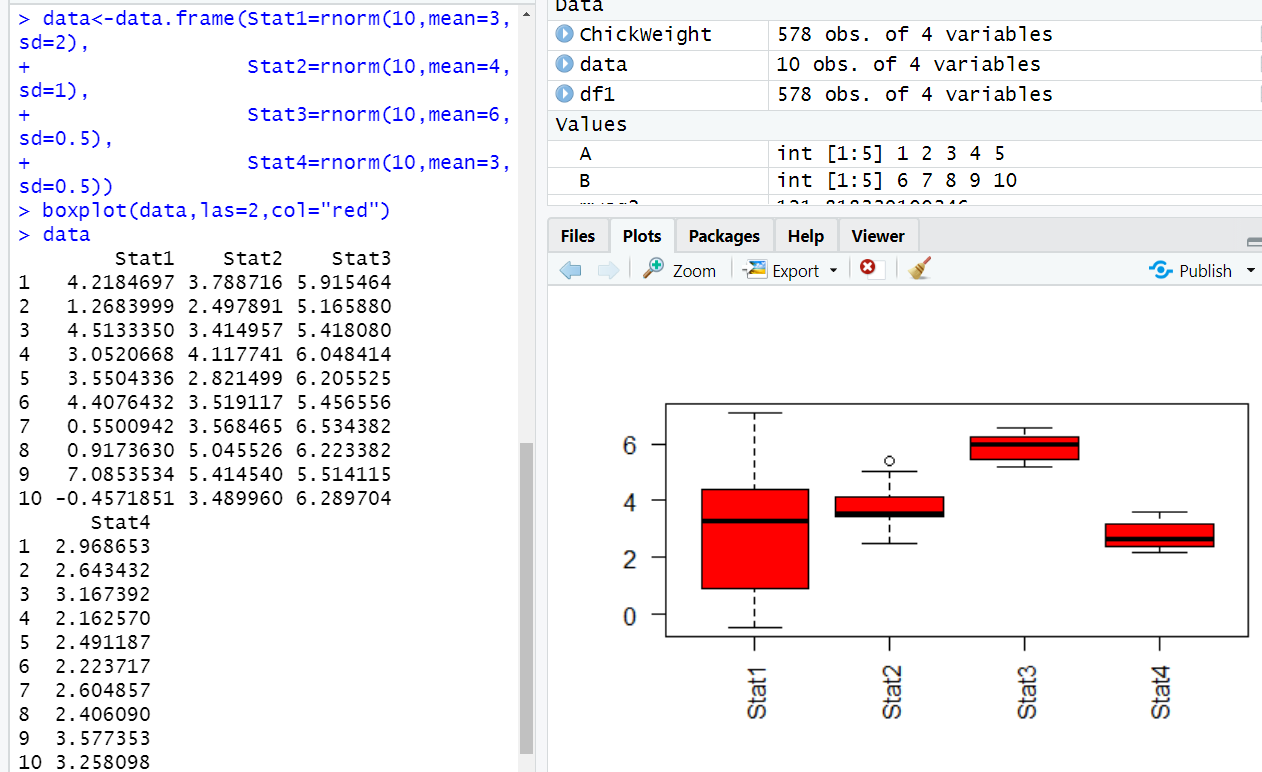
**6 2.223717**

**7 2.604857**

**8 2.406090**

**9 3.577353**

**10 3.258098**

****

#### **Q. a.3) To add multiple colours to the plot in the box plot**

### **Solution a.3 :-**

**>data<-data.frame(Stat1=rnorm(10,mean=3,sd=2),**

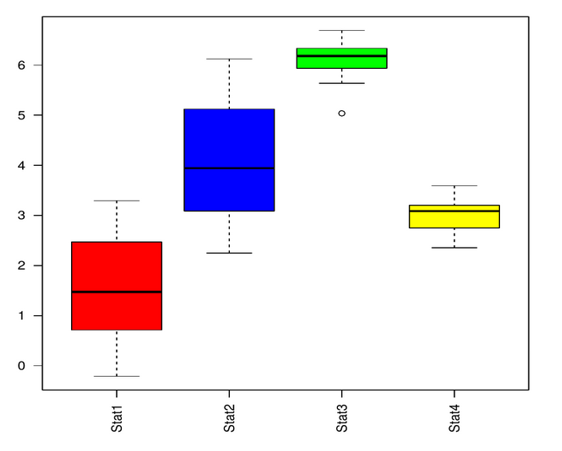
**+Stat2=rnorm(10,mean=4,sd=1),**

**+Stat3=rnorm(10,mean=6,sd=0.5),**

**+Stat4=rnorm(10,mean=3,sd=0.5))**

**>boxplot(data,las=2,col=c("red","blue","green","yellow")**

**>data**

****

#### **Q. a.4) To add Labels to the plot in the box plot**

#### **Solution a.4 :-**

**> data<-data.frame(Stat1=rnorm(10,mean=3,sd=2),**

**+ Stat2=rnorm(10,mean=4,sd=1),**

**+ Stat3=rnorm(10,mean=6,sd=0.5),**

**+ Stat4=rnorm(10,mean=3,sd=0.5))**

**> boxplot(data,las=2,xlab="statistics",ylab="random numbers",col=c("red","blue","green","yellow"))**

**> data**

**Stat1 Stat2 Stat3 Stat4**

**1 4.4304872 2.905229 6.094029 2.490670**

**2 2.3223449 3.108559 5.564624 2.341896**

**3 0.1369901 4.915104 6.277488 3.518430**

**4 2.1908523 4.223079 6.607680 2.805845**

**5 5.5997115 2.768256 5.303364 2.131868**

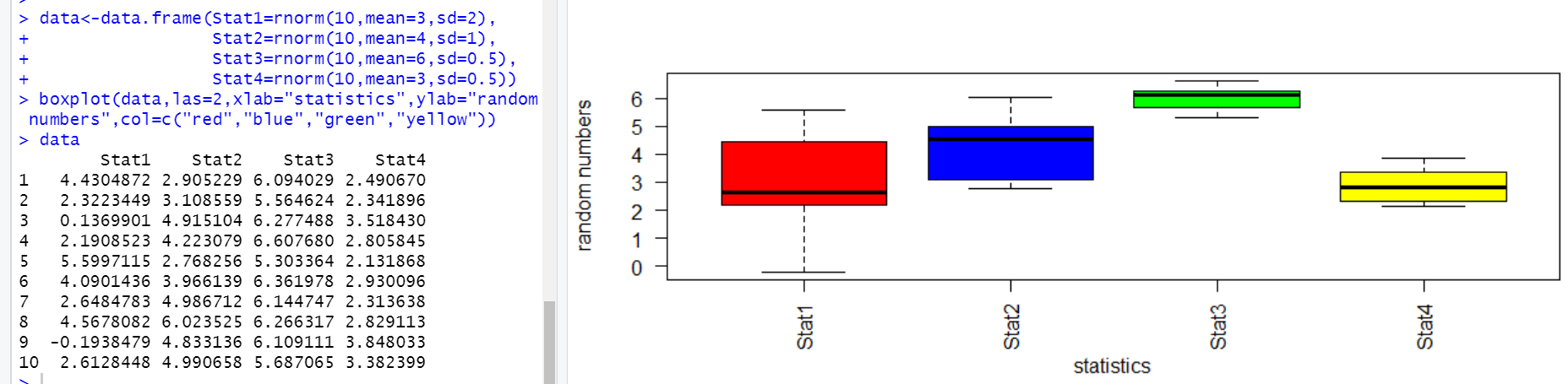
**6 4.0901436 3.966139 6.361978 2.930096**

**7 2.6484783 4.986712 6.144747 2.313638**

**8 4.5678082 6.023525 6.266317 2.829113**

**9 -0.1938479 4.833136 6.109111 3.848033**

**10 2.6128448 4.990658 5.687065 3.382399**

****

#### **Q. a.5) To add heading to the plot in the box plot by using the main parameter**

#### **Solution a.5 :-**

**> data<-data.frame(Stat1=rnorm(10,mean=3,sd=2),**

**+ Stat2=rnorm(10,mean=4,sd=1),**

**+ Stat3=rnorm(10,mean=6,sd=0.5),**

**+ Stat4=rnorm(10,mean=3,sd=0.5))**

**> boxplot(data,las=2,xlab="statistics",ylab="random numbers",main="Random relation",notch=TRUE,col=c("red","blue","green","yellow"))**

**Warning message:**

**In bxp(list(stats = c(-2.20401913547929, -0.802495957300304, 1.28982015618695, :**

**some notches went outside hinges ('box'): maybe set notch=FALSE**

**> data**

**Stat1 Stat2 Stat3 Stat4**

**1 1.8329650 5.381604 5.367963 3.191935**

**2 1.8272577 5.160662 5.251078 3.670963**

**3 6.6682260 5.806534 5.918439 2.577374**

**4 4.6641372 3.071919 6.677961 2.395785**

**5 0.3513341 3.846384 6.059116 3.859840**

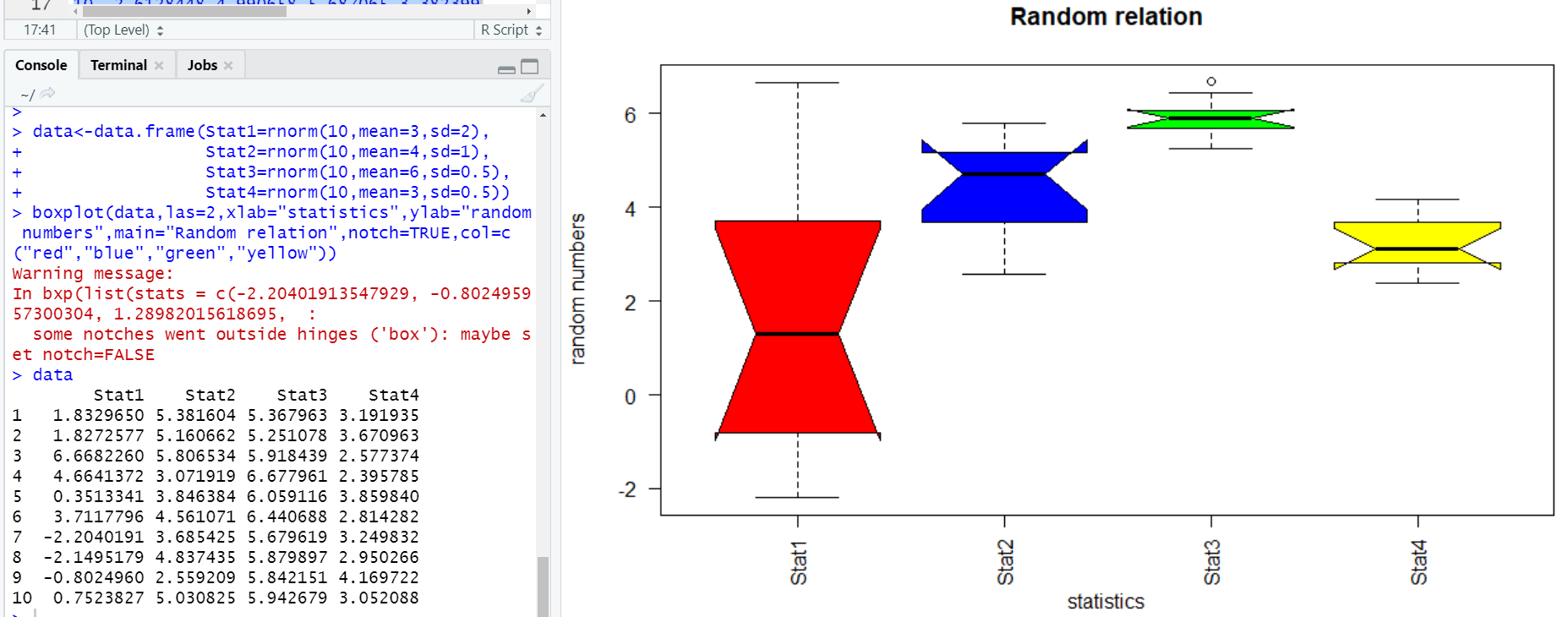
**6 3.7117796 4.561071 6.440688 2.814282**

**7 -2.2040191 3.685425 5.679619 3.249832**

**8 -2.1495179 4.837435 5.879897 2.950266**

**9 -0.8024960 2.559209 5.842151 4.169722**

**10 0.7523827 5.030825 5.942679 3.052088**

****