| Batch: A1 | Roll No.: 16010123012 |
|---|---|
| Experiment No. 1 | |

**Title: Exploring R for Data Science**

**Course Outcome:**

CO1, CO3

**Books/ Journals/ Websites referred:**

1. The Comprehensive R Archive Network
2. Posit

**Resources used:**
**https://www.rdocumentation.org/**
**https://www.w3schools.com/r/**
**https://www.geeksforgeeks.org/r-programming-language-introduction/**

---

**In this lab, we will introduce some simple R commands. R is a programming language for statistical computing and data visualization. It has been adopted in the fields of data mining, bioinformatics and data analysis**

**The best way to learn a new language is to try out the commands. R can be downloaded from http://cran.r-project.org/ . We recommend that you run R within an integrated development environment (IDE) such as RStudio, which can be freely downloaded from http://rstudio.com**

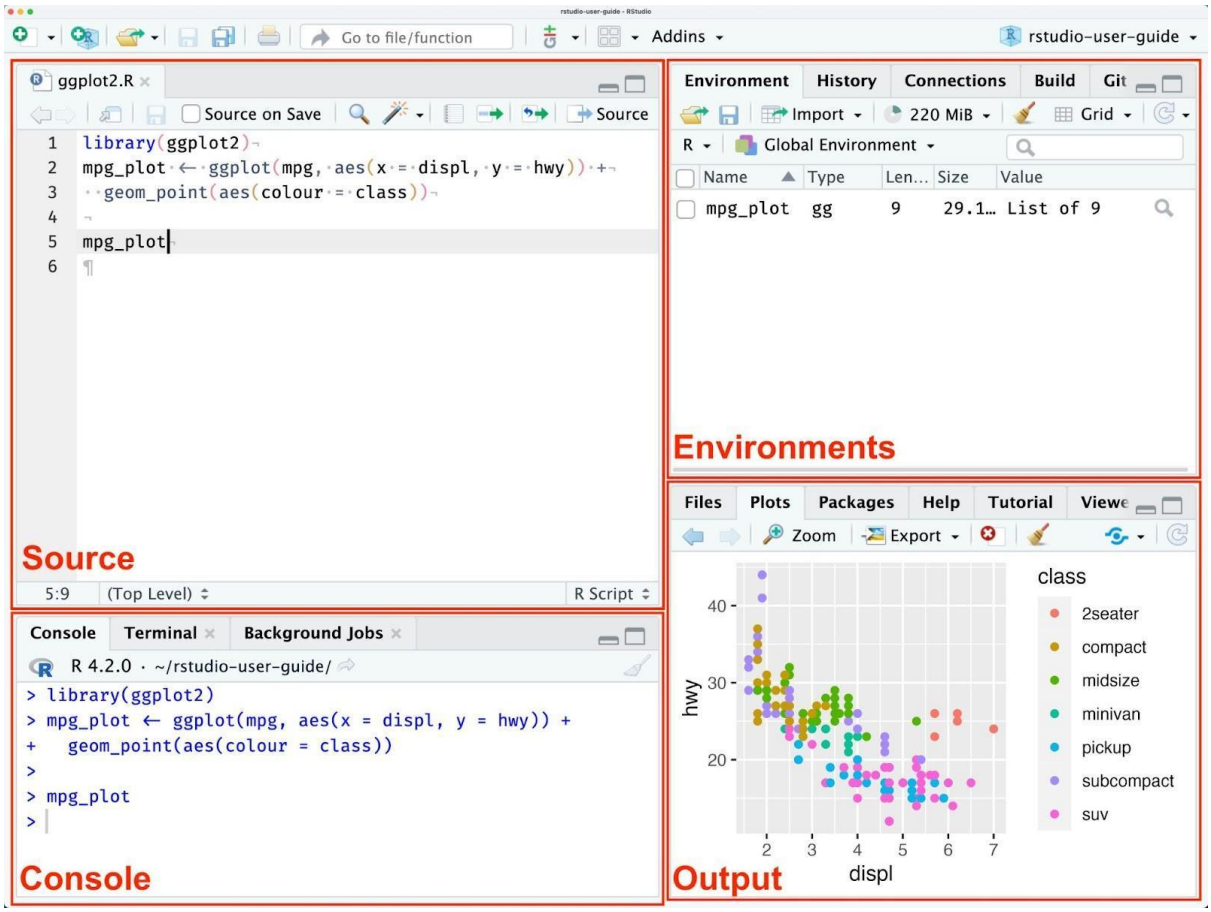
Pane Layout

**The RStudio user interface has 4 primary panes:**

- **Source pane**
- **Console pane**
- **Environment pane, containing the Environment, History, Connections, Build, VCS , and Tutorial tabs**

- **Output pane, containing the Files, Plots, Packages, Help, Viewer, and Presentation tabs**

**Each pane can be minimized or maximized within the column by clicking the minimize/maximize buttons.**




**> print("Hello, World!")**

**[1] "Hello, World!"**

**Basic Arithmetic Operators**

- **Arithmetic operations: +, -, *, /, ^, %%, %/%**
- **Assigning values: <- and =**

```
> x <- 10
> y <- 5
> quotient <- x %/% y
> remainder <- x %% y
> quotient
[1] 2
> remainder
[1] 0
```

## Basic Data Types

```
> # numeric
> x <- 10.5
> class(x)
[1] "numeric"
>
> # integer
> x <- 1000L
> class(x)
[1] "integer"
>
> # complex
> x <- 9i + 3
> class(x)
[1] "complex"
>
> # character/string
> x <- "R is exciting"
> class(x)
[1] "character"
>
> # logical/boolean
> x <- TRUE
> class(x)
[1] "logical"
```

```
> num <- as.numeric("123")
> char <- as.character(123)
> num
[1] 123
> typeof(num)
[1] "double"
> char
[1] "123"
> typeof(char)
[1] "character"
```

## Vectors and Basic Vector Operations

**R uses functions to perform operations.**
**To run a function called *funcname*, we type *funcname(input1, input2)*, where the inputs (or arguments) *input1* and *input2* tell R how to run the function.**

**A function can have any number of inputs. For example, to create a vector of numbers, we use the function c() (for concatenate).**

**Any numbers inside the parentheses are joined together.**

**The following command instructs R to join together the numbers 1, 3, 2, and 5, and to save them as a vector named x.**

```
> x <- c(1, 3, 2, 5)
> x
[1] 1 3 2 5
```

**When we type x, it gives us back the vector.**

**Other ways to create vectors are using seq() and rep()**

```
> seq_vec <- seq(1, 10, by = 2)
> seq_vec
[1] 1 3 5 7 9
> rep_vec <- rep(5, times = 3)
> rep_vec
[1] 5 5 5
```

**We can tell R to add two sets of numbers together. It will then add the first number from x to the first number from y, and so on. However, x and y should be the same length. We can check their length using the length() function.**

```
> y = c(1, 4, 3)
```

```
> length(x)
[1] 3
> length(y)
[1] 3
> x + y
[1]  2 10  5
```

**The ls() function allows us to look at a list of all of the objects, such as data and functions, that we have saved so far. The rm() function can be used to delete any that we don't want.**

```
> ls()
[1] "x" "y"
> rm(x, y)
```

```
> ls()
character(0)
```

**It's also possible to remove all objects at once:**

```
> rm(list = ls())
```

**The matrix() function can be used to create a matrix of numbers. Before we use the matrix() function, we can learn more about it:**

```
> ?matrix
```

**The help file reveals that the matrix() function takes a number of inputs, but for now we focus on the first three: the data (the entries in the matrix), the number of rows, and the number of columns. First, we create a simple matrix.**

```
> x <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

**Note that we could just as well omit typing data=, nrow=, and ncol= in the matrix() command above: that is, we could just type**

```
> x <- matrix(c(1, 2, 3, 4), 2, 2)
```

**and this would have the same effect. However, it can sometimes be useful to specify the names of the arguments passed in, since otherwise R will assume that the function arguments are passed into the function in the same order that is given in the function's help file. As this example illustrates, by default R creates matrices by successively filling in columns. Alternatively, the byrow = TRUE option can be used to populate the matrix in order of the rows.**

```
> matrix(c(1, 2, 3, 4), 2, 2, byrow = TRUE)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

**Notice that in the above command we did not assign the matrix to a value such as x. In this case the matrix is printed to the screen but is not saved for future calculations.**

```
> mat <- matrix(1:9, nrow = 3, byrow = TRUE)
>
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> mat[1,2]
[1] 2
```

**The sqrt() function returns the square root of each element of a vector or matrix. The command x^2 raises each element of x to the power 2; any powers are possible, including fractional or negative powers.**

```
> sqrt(x)
     [,1] [,2]
[1,] 1.00 1.73
[2,] 1.41 2.00
> x^2
     [,1] [,2]
[1,]    1    9
[2,]    4   16
```

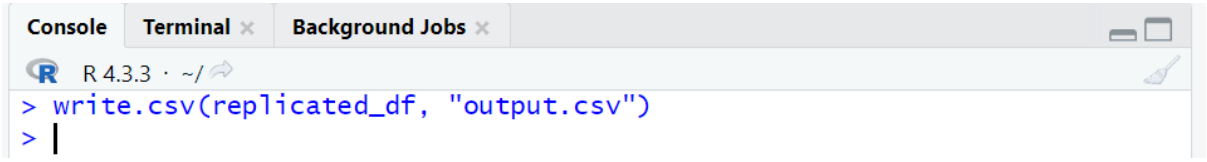
### Creating a list

```
> my_list <- list(name = "John", age = 25)
> my_list
$name
[1] "John"

$age
[1] 25

> my_list$name
[1] "John"
```

## Creating dataframes

```
> # Create a sample data frame
> df <- data.frame(
+     Name = c("Alice", "Bob", "Charlie", "David"),
+     Age = c(25, 30, 35, 40),
+     City = c("New York", "London", "Paris", "Tokyo")
+ )
> df
     Name Age     City
1   Alice  25 New York
2     Bob  30   London
3 Charlie  35    Paris
4   David  40    Tokyo
> # Replicate each row twice
> replicated_df <- cbind(df, rep(row.names(df), each = 2))
> replicated_df
     Name Age     City rep(row.names(df), each = 2)
1   Alice  25 New York                            1
2     Bob  30   London                            1
3 Charlie  35    Paris                            2
4   David  40    Tokyo                            2
5   Alice  25 New York                            3
6     Bob  30   London                            3
7 Charlie  35    Paris                            4
8   David  40    Tokyo                            4
```

## Writing dataframe to csv file



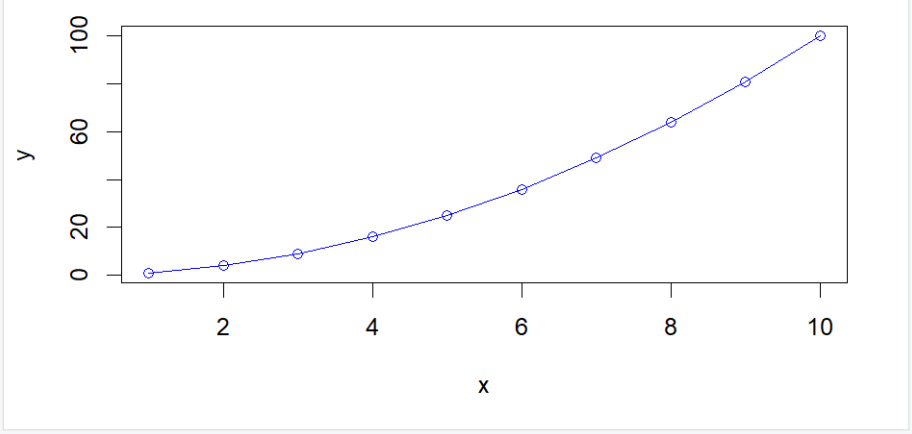
## Reading dataframe from a csv file

```
> write.csv(replicated_df, "output.csv")
> new_df <- read.csv("output.csv")
> new_df
  X    Name Age     City rep.row.names.df...each...2.
1 1   Alice  25 New York                             1
2 2     Bob  30   London                             1
3 3 Charlie  35    Paris                             2
4 4   David  40    Tokyo                             2
5 5   Alice  25 New York                             3
6 6     Bob  30   London                             3
7 7 Charlie  35    Paris                             4
8 8   David  40    Tokyo                             4
```

## Basic Visualization

1. **Line plot**

```
> x <- 1:10
> y <- x^2
> plot(x, y, type = "o", col = "blue")
>
```

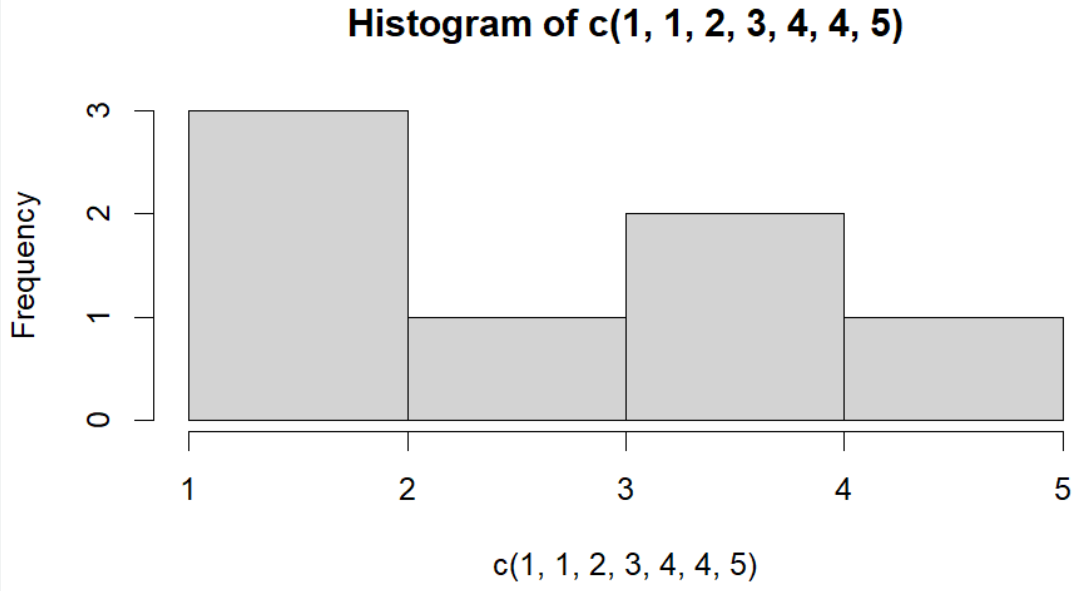


2. **Histogram**

```
Console  Terminal  Background Jobs
R 4.3.3 · ~/
> hist(x, col = "green", main = "Histogram")
> hist(c(1,1,2,3,4,4,5))
>
```

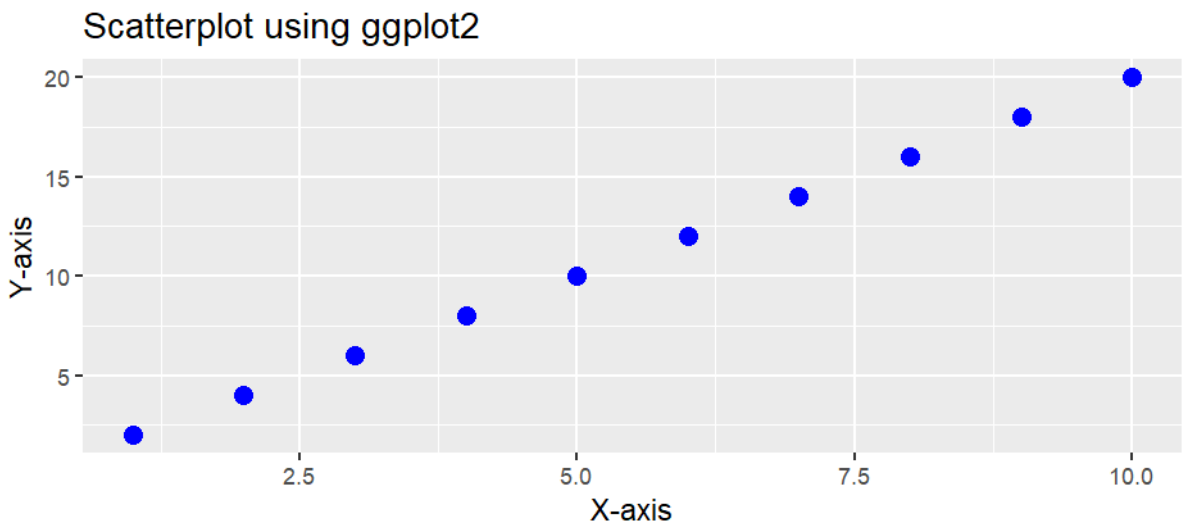
Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Publish



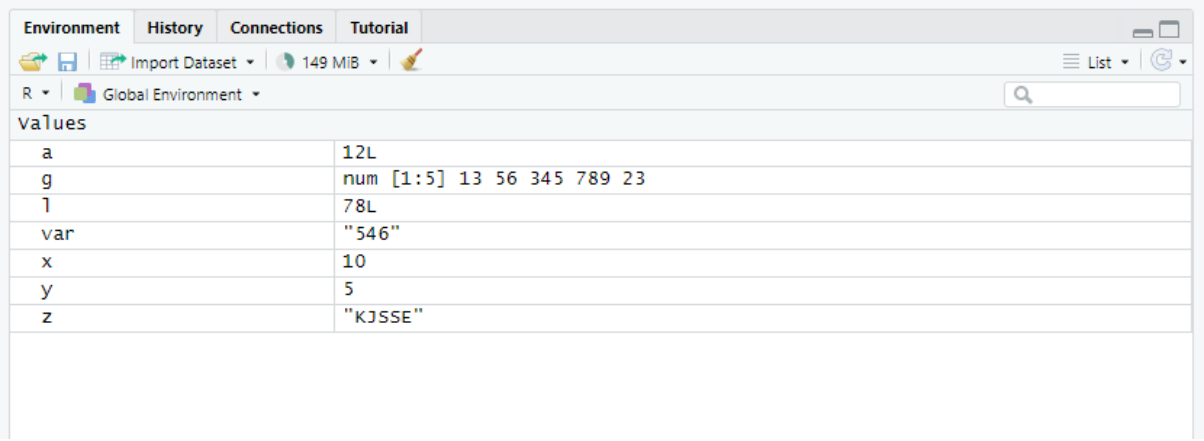
3. **Scatterplot, using ggplot2**

```
Console   Terminal ×   Background Jobs ×
R 4.3.3 · ~/
> library(ggplot2)
Learn more about the underlying theory at
https://ggplot2-book.org/
> df <- data.frame(
+     x = 1:10,
+     y = c(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
+ )
> ggplot(df, aes(x = x, y = y)) +
+     geom_point(color = "blue", size = 3) +
+     ggtitle("Scatterplot using ggplot2") +
+     xlab("X-axis") +
+     ylab("Y-axis")
```




**Students have to perform the above tasks and add their code and screenshots of the output here :**

```
> print("Hello, World!")
[1] "Hello, World!"
> x = 10
> y = 5
> class(x)
[1] "numeric"
> typeof(y)
[1] "double"
> z = 'KJSSE'
Error: unexpected input in "z = '"
> z = 'KJSSE'
> print(z)
[1] "KJSSE"
> class(z)
[1] "character"
> is.complex(z)
[1] FALSE
> print(ls())
[1] "x" "y" "z"
> a = readline()
12
> a = as.integer(a)
> 24
[1] 24
> l = as.integer(readline())
78
> var = readline(prompt = "Enter random number: ")
Enter random number: 546
> g = 0
> g = scan()
1: 13
2: 56
3: 345
4: 789
5: 23
6:
Read 5 items
>
```



```
> #numeric data type
> x<-10.5
> class(x)
[1] "numeric"
> x<-1000L
> class(x)
[1] "integer"
> x<-5i+3
> class(x)
[1] "complex"
> x<-"R is exciting"
> class(x)
[1] "character"
> x<-FALSE
> class(x)
[1] "logical"
```

```
> num<-as.numeric("124")
> char<-as.character(123)
> class(num)
[1] "numeric"
> typeof(num)
[1] "double"
> class(char)
[1] "character"
> typeof(char)
[1] "character"
>
```

```
> vec<-c(1,3,2,4,5)
> vec
[1] 1 3 2 4 5
```

```
> seq_vec=seq(1,10,by=1)
> seq_vec
 [1]  1  2  3  4  5  6  7  8  9 10
> rep_vec=rep(1,times=4)
> rep_vec
[1] 1 1 1 1
```

```
> x<-c(1,2,3)
> y<-c(4,5,6)
> length(x)
[1] 3
> length(y)
[1] 3
> x+y
[1] 5 7 9
```

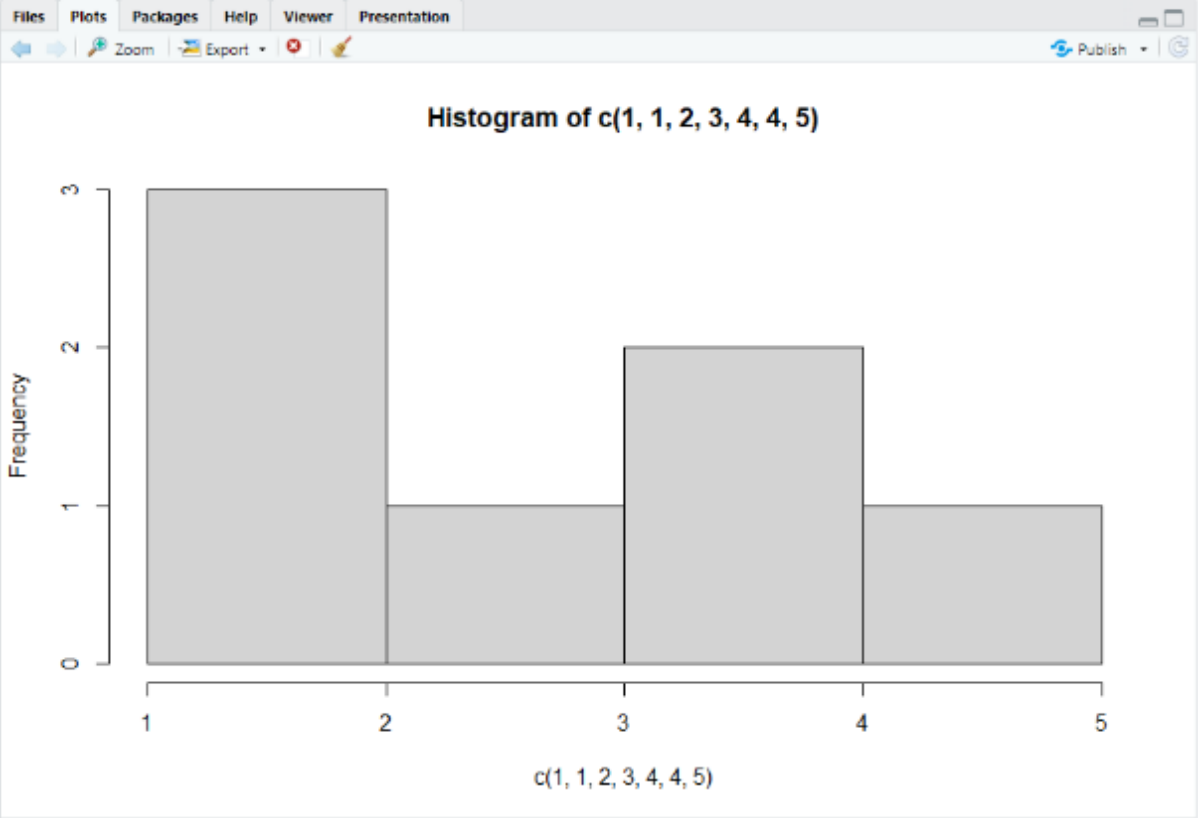
```
> x<-c(1,3,2,5)
> y<-c(1,4,3)
> ls()
[1] "a" "x" "y"
> rm(a)
```

```
> x<-matrix(data=c(1,2,3,4),nrow=2,ncol=2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> mat<-matrix(1:9,nrow=3,byrow=TRUE)
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

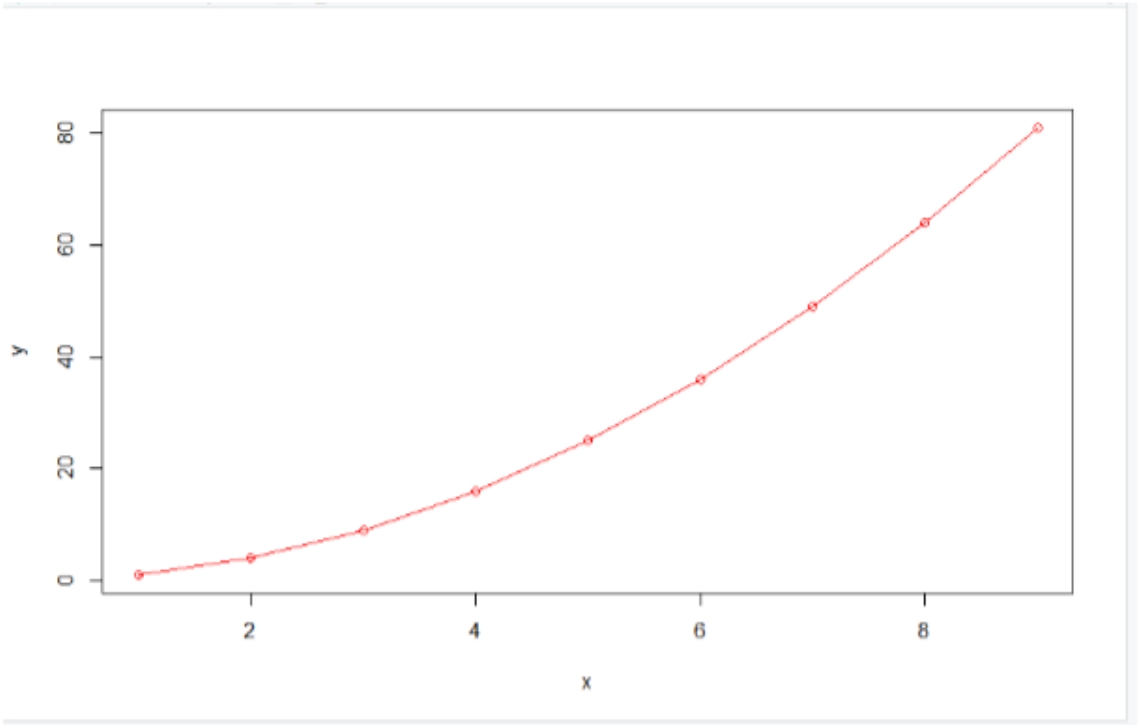
```
> sqrt(x)
         [,1]     [,2]
[1,] 1.000000 1.732051
[2,] 1.414214 2.000000
> x^2
     [,1] [,2]
[1,]    1    9
[2,]    4   16
```

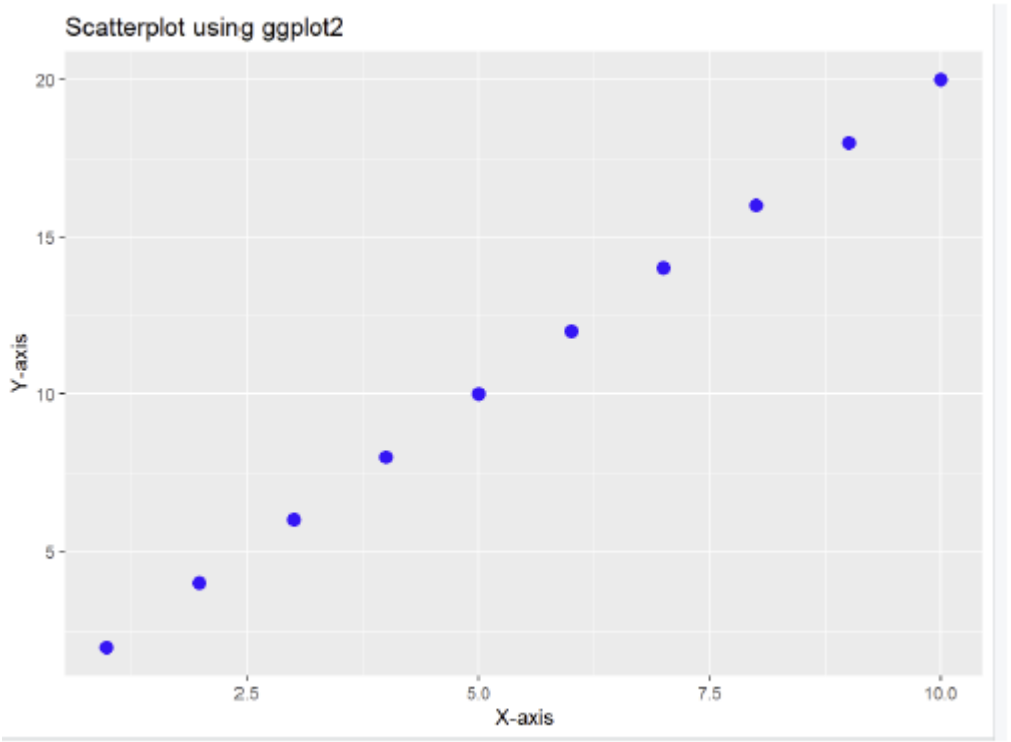
```
> df=data.frame(
+ Name=c("Alan","Bob","Cam","Dan"),
+ Age=c(20,23,25,27),
+ City=c("Mumbai","Delhi","Tokyo","Berlin")
+ )
> df
  Name Age   City
1 Alan  20 Mumbai
2  Bob  23  Delhi
3  Cam  25  Tokyo
4  Dan  27 Berlin
```

```
> hist(x, col="green", main="Histogram")
> hist(c(1,1,2,3,4,4,5))
```




```
> x<-1:9
> y<-x^2
> plot(x,y, type="o",col="red")
```


y
x
0
20
40
60
80
2
4
6
8


Scatterplot using ggplot2
Y-axis
X-axis
5
10
15
20
2.5
5.0
7.5
10.0

**Conclusion:**
I have successfully completed this experiment and learned about the basics of R programming.

**Post Lab Questions:**

1. **Explain the difference between vectors, lists, and data frames in R. Provide an example of when each would be used.**
2. **What are the differences between the plot() and ggplot() functions in R? When would you use each?**
3. **What parameters can you customize in the hist() function to change the appearance of the histogram?**
4. **Write an R command to create a vector of numbers from 1 to 100, but only include multiples of 5.**

Aaryan Sharma
16010123012

# IDS Exp 1

1) Vectors: Homogeneous. 1-D objects that contain elements of same type. eg: ageo = c(18,19,20) # Numeric vector
Used when same type of elements are needed to be stored.
Lists: Heterogenous objects that can contain elements of diff types. Used when related but diverse data types are stored together.
eg. list <- c("a", "b", 1, 2) # List
Dataframes: Two D tabular structure where each column is a vector & columns can have diff types. Used when we need to store & analyze structured data.
ex. df <- data.frame(name = c("A","B"), no. = c(1,2))

2) plot():
(i) Base R graphics fn
(ii) Offers quick, simple plotting with limited customisation.
(iii) Less flexible for complex visualisations
(iv) Used for simple exploratory plotting.
ggplot():
(i) Part of ggplot2 package
(ii) Highly customisable & supports layered, complex visualisations.
(iii) Allows for modular addition of components
(iv) Used for creating publication-quality or complex visualisations

3) (i) breaks: Number or method to determine number of bins
(ii) col: Color of the bars
(iii) main: Title of the histogram
(iv) xlab/ylab: Labels for X & Y axis
(v) xlim/ylim: Limits for X & Y axis
(vi) Border: Color of the bar borders

4) multi <- seq(5, 100, by = 5)