Assignment - 1 To 4

Task 1:

1. How many ways are there to call a function in R?

There are 3 ways to call a function in R.

I. ***Calling a Function without an Argument:***

Example:

```
# Create a function without an argument.
new.function <- function() {
  for(i in 1:5) {
    print(i^2)
  }
}

# Call the function without supplying an argument.
new.function()
```

II. ***Calling a Function with Argument Values (by position and by name):***

The arguments to a function call can be supplied in the same sequence as defined in the function or they can be supplied in a different sequence but assigned to the names of the arguments.

Example:

```
# Create a function with arguments.
new.function <- function(a,b,c) {
  result <- a * b + c
  print(result)
}

# Call the function by position of arguments.
new.function(5,3,11)

# Call the function by names of the arguments.
new.function(a = 11, b = 5, c = 3)
```

III. Calling a Function with Default Argument:

We can define the value of the arguments in the function definition and call the function without supplying any argument to get the default result. But we can also call such functions by supplying new values of the argument and get non default result.

Example:

```
# Create a function with arguments.
new.function <- function(a = 3, b = 6) {
  result <- a * b
  print(result)
}

# Call the function without giving any argument.
new.function()

# Call the function with giving new values of the argument.
new.function(9,5)
```

2. What is the Recycling of elements in a vector?

Solution: *Recycling of elements in a vector in R is, how R automatically recycles, or repeats, elements of the shorter Vector when applying an operation to two vectors that requires them to be the same length.*
R automatically recycles, or repeats, elements of the shorter one, until it is long enough to match the longer Vector.

3. Give an example of recycling of elements.

Example:
Suppose we have two Vectors c (1,2,4), c (6,0,9,10,13), where the first one is shorter with only 3 elements. Now if we sum these two, we will get a warning message as follows.

```
> c(1,2,4) + c(6,0,9,10,13)
[1]  7  2 13 11 15
```

```
Warning message:
In c(1, 2, 4) + c(6, 0, 9, 10, 13) :  longer object length is not a multiple of shorter object length
```

Here R , Sum those Vectors by Recycling or repeating the elements in shorter one, until it is long enough to match the longer one as follows..

```
> c(1,2,4,1,2) + c(6,0,9,10,13)
[1]  7  2 13 11 15
```

Task 2:

1. What should be the output of the following Script?

v <- c (2,5.5,6)
t <- c (8, 3, 4)
print (v% / %t)

Output:

```
> v <- c( 2,5.5,6)
> t <- c(8, 3, 4)
> print(v%/%t)
[1] 0 1 1
```

2. You have 25 excel files with names as xx_1.xlsx, xx_2.xlsx, ...xx_25.xlsx in a dir. Write a program to extract the contents of each excel sheet and make it one df.

Solution1:

```
library(xlsx)
files=list.files(pattern="_1.csv")
df_total=data.frame()
for(i in 1:length(files))
{
 filename=files[i]
 data=read.csv(file = filename,header = T)
 df_total=rbind(df_total,data)
}
df_total
```

Output:

```
> df_total
     Company Ranking
1        HCL       5
2  JP Morgan       6
3   Facebook       3
4  Microsoft       4
5     Google       1
6     Amazon       2
```

or

Solution2:

```
files=list.files(pattern="_1.csv")
for(i in 1:length(files))
{filename=files[i]
data=read.csv(file = filename,header = T)
```

```
assign(x = filename,value = data)}
data
abc=read.csv("ey_1.csv")
abc
library(dplyr)
bind_rows(data,abc)
```

Output:

```
> files=list.files(pattern="_1.csv")
>
> for(i in 1:length(files))
+
+ {filename=files[i]
+ data=read.csv(file = filename,header = T)
+
+ assign(x = filename,value = data)}
> data
  Company Ranking
1  Google       1
2  Amazon       2
> abc=read.csv("ey_1.csv")
> abc
    Company Ranking
1  Facebook       3
2 Microsoft       4
> library(dplyr)
> bind_rows(data,abc)
    Company Ranking
1    Google       1
2    Amazon       2
3  Facebook       3
4 Microsoft       4
```

Task 3:

1. Create an m x n matrix with replicate (m, rnorm(n)) with m=10 column vectors of n=10 elements each, constructed with rnorm(n), which creates random normal numbers.

Then we transform it into a dataframe (thus 10 observations of 10 variables) and perform an algebraic operation on each element using a nested for loop: at each iteration, every element referred by the two indexes is incremented by a sinusoidal function, compare the vectorized and non-vectorized form of creating the solution and report the system time differences.

Solution:

```
m=10; n=10;
mymat<-replicate(m, rnorm(n)) # create matrix of normal random numbers

mydframe=data.frame(mymat)# transform into data frame

for (i in 1:m) {
  for (j in 1:n) {
```

```
   mydframe[i,j]<-mydframe[i,j] + 10*sin(0.75*pi)
  }
}
mydfsin = mydframe
print(mydfsin)
```

Output:

```
> print(mydfsin)
          X1       X2       X3       X4       X5       X6       X7       X8
1  13.39354 13.24283 13.69747 13.95793 14.58475 14.27874 15.94850 13.64435
2  12.76774 13.35745 15.11878 13.78519 13.87093 15.20532 13.04631 13.03821
3  11.85668 13.70572 13.86343 14.19366 14.50514 13.84825 14.44735 13.76269
4  16.18751 12.94968 14.37052 15.39534 13.31033 14.11087 15.17301 14.93887
5  14.20621 12.67041 15.00352 14.68083 16.08200 13.17299 11.84720 13.07611
6  14.95676 14.95621 15.30184 15.11522 13.63548 13.46231 14.29492 15.29254
7  13.37664 14.68827 13.98946 14.02742 13.50911 12.97950 14.30424 14.66569
8  13.59583 15.98624 14.13182 13.26460 13.52081 14.74984 13.51209 14.65057
9  14.75442 14.04125 12.38548 13.39404 14.12290 11.59378 15.00260 14.44387
10 14.38700 14.04020 14.48446 12.83653 13.57364 13.44730 14.97641 16.56565
          X9       X10
1  14.40610 15.99758
2  13.02671 13.42029
3  14.98964 14.63240
4  14.47810 13.07676
5  14.22789 14.96754
6  14.29541 14.43974
7  12.61910 12.63015
8  15.15950 14.12712
9  16.20425 15.87064
10 14.96498 13.62949
```

System time differences:

Solution:

```
TS = system.time(for (i in 1:m) {
 for (j in 1:n) {
   mydframe[i,j]<-mydframe[i,j] + 10*sin(0.75*pi)
 }
})

print(TS[3])
```

Output:

```
user  system elapsed
0.02    0.00    0.02
```

Task 4:

1. Define matrix mymat by replicating the sequence 1:5 for 4 times and transforming into a matrix, sum over rows and columns.

replicate(4,1:5,simplify = T)
mymat<-(replicate(4,1:5,simplify = T))
apply(mymat,1,sum)
apply(mymat,2,sum)

Output:

```
> replicate(4,1:5,simplify = T)
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    2    2    2    2
[3,]    3    3    3    3
[4,]    4    4    4    4
[5,]    5    5    5    5
> mymat<-(replicate(4,1:5,simplify = T))
> apply(mymat,1,sum)
[1]  4  8 12 16 20
> apply(mymat,2,sum)
[1] 15 15 15 15
```

Task 5:

1. States = rownames(USArrests)
Get states names with 'w'.
Get states names with 'W'.

Solution:
Get states names with 'w'.

States = rownames(USArrests)
rownames(USArrests)

grep("w",rownames(USArrests))
States_with_w <-grep("w",States)

for (i in 1:length(States_with_w))
{
print(States[States_with_w[i]])
}

Output:

```
> States = rownames(USArrests)
> grep("w",rownames(USArrests))
[1]  8 11 15 29 30 31 32
> States_with_w <-grep("w",States)
> for (i in 1:length(States_with_w)){
+   print(States[States_with_w[i]])
+
+ }
[1] "Delaware"
[1] "Hawaii"
[1] "Iowa"
[1] "New Hampshire"
[1] "New Jersey"
[1] "New Mexico"
[1] "New York"
```

Get states names with 'W'.

States = rownames(USArrests)
rownames(USArrests)

grep("W",rownames(USArrests))
States_with_W <-grep("W",States)

for (i in 1:length(States_with_W))
{
print(States[States_with_W[i]])
}

Output:

```
> States = rownames(USArrests)
> grep("w",rownames(USArrests))
[1] 47 48 49 50
> States_with_w <-grep("w",States)
> for (i in 1:length(States_with_w)){
+   print(States[States_with_w[i]])
+
+ }
[1] "Washington"
[1] "West Virginia"
[1] "Wisconsin"
[1] "Wyoming"
```

2. Prepare a Histogram of the number of characters in each US state.

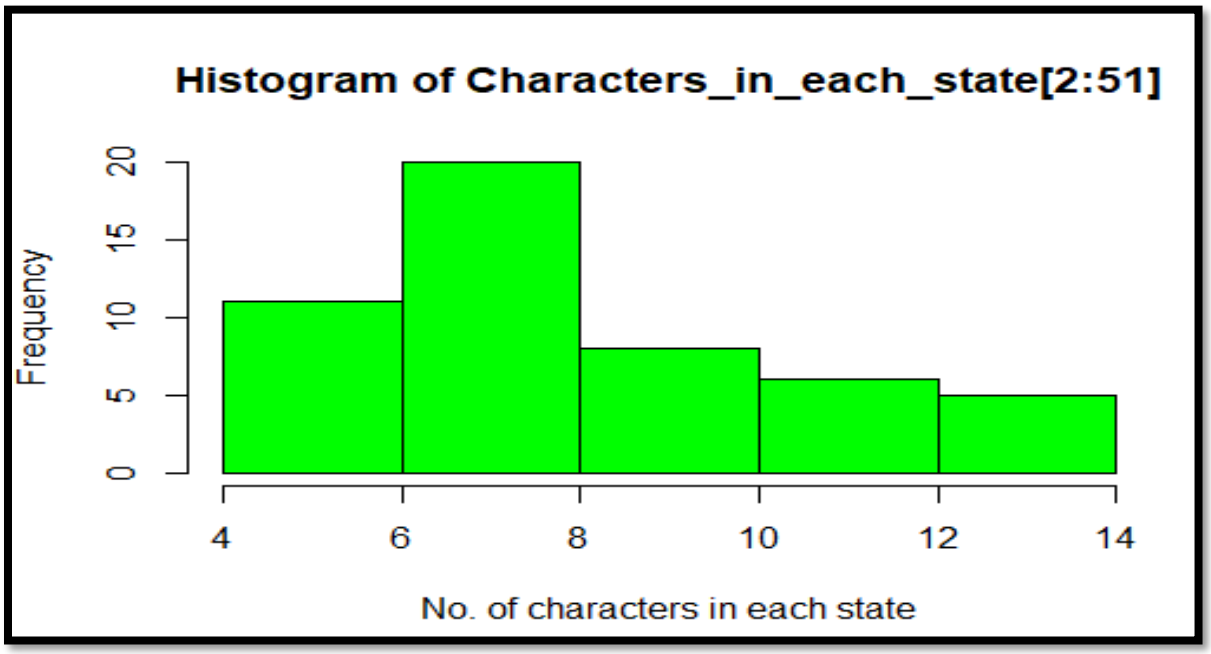
Solution:
Characters_in_each_state <- c(0)
for(i in 1:50){

```
  temp <- States[i]
  len <- nchar(temp)
  Characters_in_each_state <- c(Characters_in_each_state,len)

}

hist(Characters_in_each_state[2:51],xlab="No. of characters in each state",col = " green")
```

Output:



Task 6:

1. Test whether two vectors are exactly equal (element by element).

vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,])

Solution1:

```
> vec1 = c(rownames(mtcars[1:15,]))
> vec2 = c(rownames(mtcars[11:25,]))
> vec1 == vec2
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[14] FALSE FALSE
```

Solution2:

```
> vec1 = c(rownames(mtcars[1:15,]))
> vec2 = c(rownames(mtcars[11:25,]))
> setequal(vec1,vec2)
[1] FALSE
```

2. Sort the character vector in ascending order and descending order.
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))

Solution:
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))
sort(vec1, decreasing = FALSE)
sort(vec2, decreasing = TRUE)

Output:

```
> vec1 = c(rownames(mtcars[1:15,]))
> vec2 = c(rownames(mtcars[11:25,]))
> sort(vec1, decreasing = FALSE)
 [1] "Cadillac Fleetwood" "Datsun 710"         "Duster 360"
 [4] "Hornet 4 Drive"     "Hornet Sportabout"  "Mazda RX4"
 [7] "Mazda RX4 Wag"      "Merc 230"           "Merc 240D"
[10] "Merc 280"           "Merc 280C"          "Merc 450SE"
[13] "Merc 450SL"         "Merc 450SLC"        "Valiant"
> sort(vec2, decreasing = TRUE)
 [1] "Toyota Corona"       "Toyota Corolla"      "Pontiac Firebird"
 [4] "Merc 450SLC"         "Merc 450SL"          "Merc 450SE"
 [7] "Merc 280C"           "Lincoln Continental" "Honda Civic"
[10] "Fiat 128"            "Dodge Challenger"    "Chrysler Imperial"
[13] "Camaro Z28"          "Cadillac Fleetwood"  "AMC Javelin"
```

3. What is the major difference between str() and paste() show an example?

Solution:
paste(): *This is a function which is used to* ***concatenate the strings****.*

Example:

```
> v="Virat"
> x="Kohli"
> paste(v,x)
[1] "Virat Kohli"
```

str(): *This is a function which converts* ***data frame*** *into* ***list*** *format.*

Example:

```
> L1<-list(mtcars[,1:5])
> str(L1)
List of 1
 $ :'data.frame':       32 obs. of  5 variables:
  ..$ mpg : num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
  ..$ cyl : num [1:32] 6 6 4 6 8 6 8 4 4 6 ...
  ..$ disp: num [1:32] 160 160 108 258 360 ...
  ..$ hp  : num [1:32] 110 110 93 110 175 105 245 62 95 123 ...
  ..$ drat: num [1:32] 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
```

4. Introduce a separator when concatenating the strings.

Solution:

```
> V="Asif"
> X="Faiza"
> paste(V,X, sep = ",")
[1] "Asif,Faiza"
```

Task 7:

1. Import the Titanic Dataset from the link => Titanic Data Set.

Perform the following:

a. Is there any difference in fares by a different class of tickets?

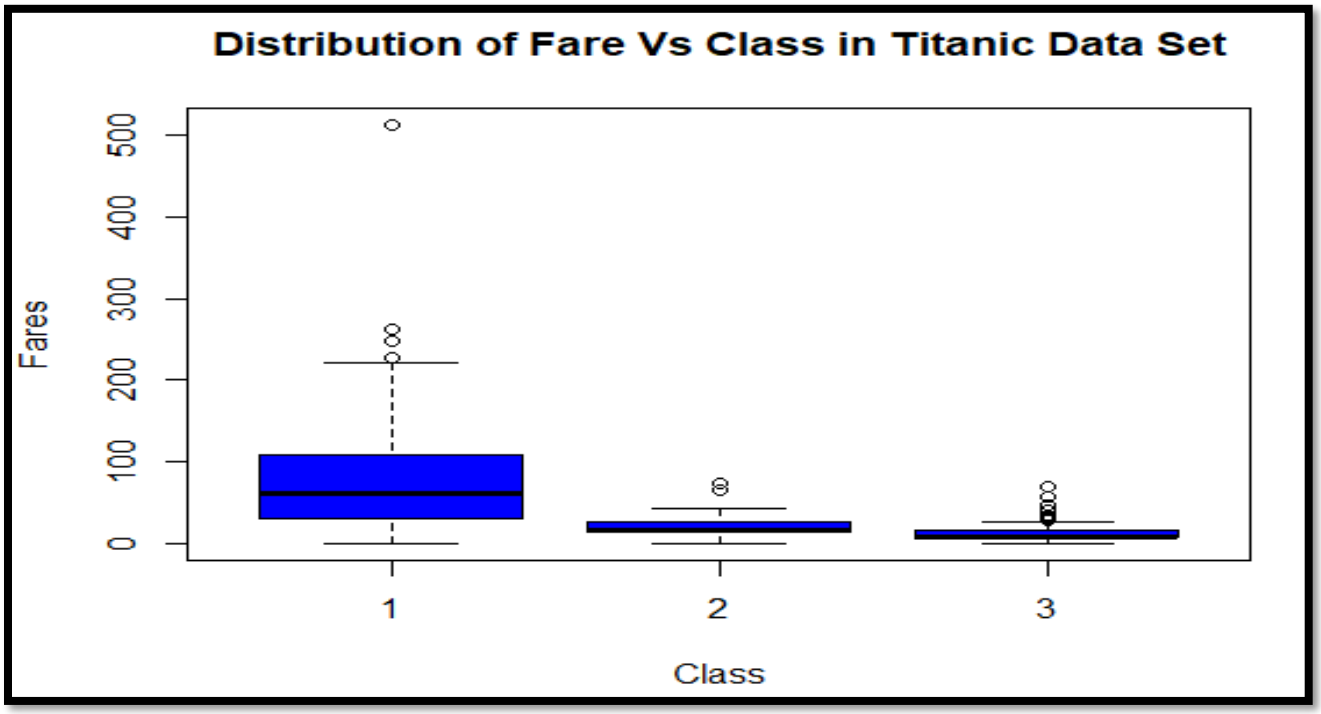
Note *- Show a boxplot displaying the distribution of fares by class.*

Solution:

Titanic1 = read.csv("Titanic.csv")

boxplot(fare ~ pclass, data = Titanic1, xlab = "Class",
ylab = "Fares", main = "Distribution of Fare Vs Class in Titanic Data Set", col="Blue")

Output:



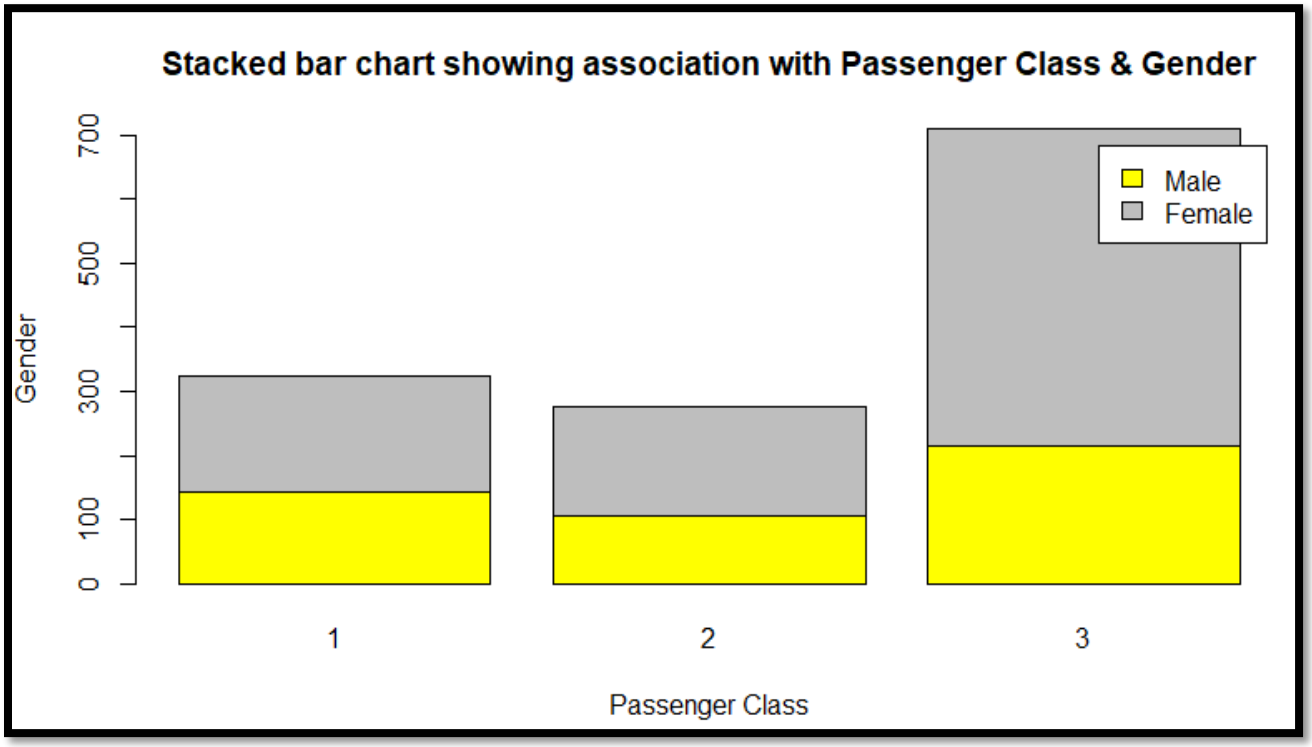
b. Is there any association with Passenger class and gender?
Note – *Show a stacked bar chart*

Solution:

```
Titanic1 = read.csv("Titanic.csv")

counts <- table(Titanic1$sex, Titanic1$pclass)
barplot(counts, main="Stacked bar chart showing association with Passenger Class &
Gender",  xlab="Passenger Class",ylab = "Gender",col=c("grey","yellow"),legend.text =
c("Female", "Male"))
```

Output:



Task 8:

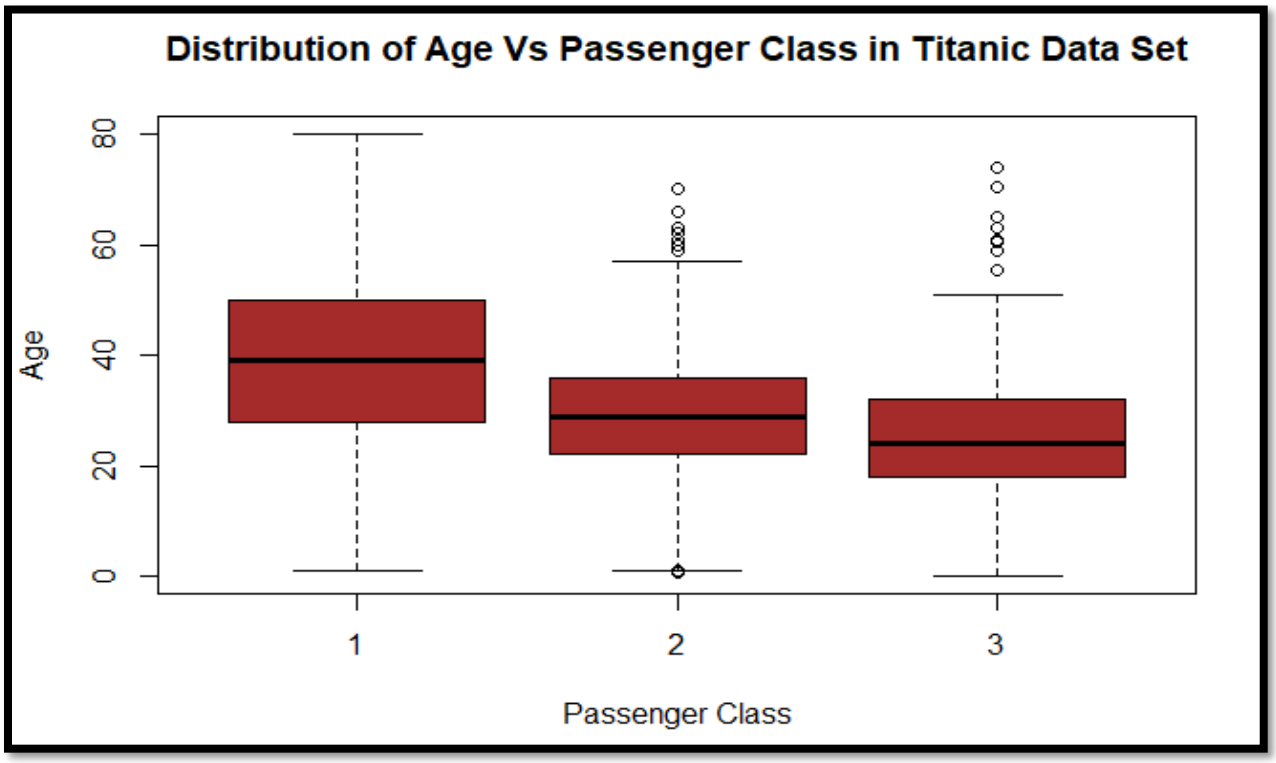
1. Create a box and whisker plot by class and count using Titanic dataset.

Solution:

```
Titanic1 = read.csv("Titanic.csv")

boxplot(age ~ pclass, data = Titanic1, xlab = "Passenger Class",
      ylab = "Age", main = "Distribution of Age Vs Passenger Class in Titanic Data Set",
col="Brown")
```

Output:



Task 9:

1. A recent national study showed that approximately 44.7% of college students have used Wikipedia as a source in at least one of their term papers. Let X equal the number of students in a random sample of size n = 31 who have used Wikipedia as a source.

Perform the below functions

a. Find the probability that X is equal to 17.

Solution:

dbinom(17, size = 31, prob = 0.447)

Output:

```
> dbinom(17, size = 31, prob = 0.447)
[1] 0.07532248
```

b. Find the probability that X is at most 13

Solution:

pbinom(13, size = 31, prob = 0.447)

Output:

```
> pbinom(13, size = 31, prob = 0.447)
[1] 0.451357
```

c. Find the probability that X is bigger than 11.

Solution:

pbinom(11, size = 31, prob = 0.447, lower.tail = FALSE)

Output:

```
> pbinom(11, size = 31, prob = 0.447, lower.tail = FALSE)
[1] 0.8020339
```

d. Find the probability that X is at least 15.

Solution:

pbinom(14, size = 31, prob = 0.447, lower.tail = FALSE)

Output:

```
> pbinom(14, size = 31, prob = 0.447, lower.tail = FALSE)
[1] 0.406024
```

e. Find the probability that X is between 16 and 19, inclusive

Solution-1:

sum(dbinom(16:19, size = 31, prob = 0.447))

Output:

```
> sum(dbinom(16:19, size = 31, prob = 0.447))
[1] 0.2544758
```

Or

Solution-2:

diff(pbinom(c(19, 15), size = 31, prob = 0.447, lower.tail = FALSE))

Output:

```
> diff(pbinom(c(19, 15), size = 31, prob = 0.447, lower.tail = FALSE))
[1] 0.2544758
```

6. Expected Output

Solution document/report shall be in PDF format. Submitted in GitHub.
The PDF Doc should have Code with its subsequent result screenshot.