**R Assignment:**

**1.Try to write a code for printing sequence of numbers from 1 to 50 with the differences of 3, 5, 10**

x<-seq(1,50,3)

print(x)

y<-seq(1,50,5)

print(y)

z<-seq(1,50,10)

print(z)

Output:

> x<-seq(1,50,3)

> print(x)

[1] 1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49

> y<-seq(1,50,5)

> print(y)

[1] 1 6 11 16 21 26 31 36 41 46

> z<-seq(1,50,10)

> print(z)

[1] 1 11 21 31 41

--------------------------------------------------------------------------------------------------------------------------------

**2. What are the different data objects in R? and write syntax and example for each and every object**

Everything in R is an Object, which has 6 basic datatypes.

1)Character

2)Numeric(real or decimal)

3)Integer

4)Logical

5)Complex

6)DataFrame

Example:

1)Character : "a","swc"

2)numeric: 2,15.5

3)integer: 2L(the L tells R to store this as an integer)

4)logical: TRUE,FALSE

5)Complex: 1+4i(Complex numbers with real and imaginary parts)

6)Data Frame:Data sets like IRIS,mtcar etc.

----------------------------------------------------------------------------------------------------------------------------------

**3. Create Data frame with 3 columns and 5 rows and write a code to fetch and delete row and a column using index and add new column and row to the existed data frame**

df1=data.frame(CustomerID=(1:5),Company=c("Samsung","Samsung","Haier","Haier","Sony"),Product=c(rep("Oven",3),(rep("TV",2))))

#To fetch a row

fetr<-df1[1,]

print(fetr)

#To fetch a column

fetc<-df1[,1]

print(fetc)

#To delete a row

delr<-df1[-c(1),]

print(delr)

#To delete a column

delc<-df1[,-c(1)]

print(delc)

#To add a column

Price<-c(55000,56000,72000,74000,20000)

df\_new<-cbind(df1,Price)

df\_new

#To add a row

df1=rbind(df1,data.frame("CustomerID"="6","Company"="Prestige","Product"="Mixer"))

print(df1)

Output:

> df1=data.frame(CustomerID=(1:5),Company=c("Samsung","Samsung","Haier","Haier","Sony"),Product=c(rep("Oven",3),(rep("TV",2))))

>

> #To fetch a row

> fetr<-df1[1,]

> print(fetr)

CustomerID Company Product

1 1 Samsung Oven

> #To fetch a column

> fetc<-df1[,1]

> print(fetc)

[1] 1 2 3 4 5

> #To delete a row

> delr<-df1[-c(1),]

> print(delr)

CustomerID Company Product

2 2 Samsung Oven

3 3 Haier Oven

4 4 Haier TV

5 5 Sony TV

> #To delete a column

> delc<-df1[,-c(1)]

> print(delc)

Company Product

1 Samsung Oven

2 Samsung Oven

3 Haier Oven

4 Haier TV

5 Sony TV

> #To add a column

> Price<-c(55000,56000,72000,74000,20000)

> df\_new<-cbind(df1,Price)

> df\_new

CustomerID Company Product Price

1 1 Samsung Oven 55000

2 2 Samsung Oven 56000

3 3 Haier Oven 72000

4 4 Haier TV 74000

5 5 Sony TV 20000

> #To add a row

> df1=rbind(df1,data.frame("CustomerID"="6","Company"="Prestige","Product"="Mixer"))

> print(df1)

CustomerID Company Product

1 1 Samsung Oven

2 2 Samsung Oven

3 3 Haier Oven

4 4 Haier TV

5 5 Sony TV

6 6 Prestige Mixer

----------------------------------------------------------------------------------------------------------------------------------

**4.Write nested if else statements to print whether the given number is negative, positive or Zero**

**num = as.double(readline(prompt="Enter a number: "))**

if(num > 0) {

print("Positive number")

} else {

if(num == 0) {

print("Zero")

} else {

print("Negative number")

}

}

Output:

Enter a number: -9.6

[1] "Negative number"

Enter a number: 2

[1] "Positive number"

--------------------------------------------------------------------------------------------------------------------------------------

**5.write a program to input any value and check whether it is character, numeric or special character.**

**write difference between break and next also write examples for both**

x<-"$"

is.character(x)

is.numeric(x)

grepl("^[A-Za-z]+$",x)

Output:

> x<-"$"

> is.character(x)

[1] TRUE

> is.numeric(x)

[1] FALSE

> grepl("^[A-Za-z]+$",x)

[1] FALSE

**Break Statement**

The break keyword is a jump statement that is used to terminate the loop at a particular iteration.

**Syntax:**

if (test\_expression) {

break

}

**Example 1: Using break in For-loop**

# R program for break statement in For-loop

no <**-** 1:10

**for** (val **in** no)

{

**if** (val **==** 5)

    {

**print**(paste("Coming out from for loop Where i = ", val))

**break**

    }

**print**(paste("Values are: ", val))

}

**Output:**

[1] "Values are: 1"

[1] "Values are: 2"

[1] "Values are: 3"

[1] "Values are: 4"

[1] "Coming out from for loop Where i = 5"

**Example 2: Using break statement in While-loop**

# R Break Statement Example

a<**-**1

**while** (a < 10)

{

    print(a)

**if**(a**==**5)

**break**

    a **=** a **+** 1

}

**Output:**

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

#### Next Statement

The next statement is used to skip the current iteration in the loop and move to the next iteration without exiting from the loop itself.  
**Syntax:**

if (test\_condition)

{

next

}

**Example 1: Using next statement in For-loop**

# R Next Statement Example

no <**-** 1:10

**for** (val **in** no)

{

**if** (val **==** 6)

    {

**print**(paste("Skipping for loop Where i =  ", val))

|  |
| --- |
|  |

**Output:**

[1] "Values are: 1"

[1] "Values are: 2"

[1] "Values are: 3"

[1] "Values are: 4"

[1] "Values are: 5"

[1] "Skipping for loop Where i = 6"

[1] "Values are: 7"

[1] "Values are: 8"

[1] "Values are: 9"

[1] "Values are: 10"

**Example 2: Using next statement in While-loop**

# R Next Statement Example

x <**-** 1

**while**(x < 5)

{

    x <**-** x **+** 1;

**if** (x **==** 3)

        next;

    print(x);

}

**Output:**

[1] 2

[1] 4

[1] 5

--------------------------------------------------------------------------------------------------------------------------------------

**7.write a program to print a given vector in reverse format**

x= c(1,5.6,3,10,3.5,5)

print("Original vector-1: ")

print(v)

rv=rev(v)

print("The said vector in Reverse Order: ")

print(rv)

Output:

> v=c(1,5.6,3,10,3.5,5)

> print("Original vector-1: ")

[1] "Original vector-1: "

> print(v)

[1] 1.0 5.6 3.0 10.0 3.5 5.0

> rv=rev(v)

> print("The said vector in Reverse Order: ")

[1] "The said vector in Reverse Order: "

> print(rv)

[1] 5.0 3.5 10.0 3.0 5.6 1.0

--------------------------------------------------------------------------------------------------------------------------------------

**8.write a program to get the mode value of the given vector (‘a’,’b’,’c’,’t’,’a’,’c’,’r’,’a’,’c’,’t’,’z’,’r’,’v’,’t’,’u’,’e’,’t’)**

#Create the function

getmode<-function(v){

uniqr<-unique(v)

uniqr[which.max(tabulate(match(v,uniqr)))]

}

#Create the vector with characters

charv<-c("a","b","c","t","a","c","r","a","c","t","z","r","v","t","u","e","t")

#Calculate the mode using the user function

result<-getmode(charv)

print(result)

Output:

> #Create the function

> getmode<-function(v){

+ uniqr<-unique(v)

+ uniqr[which.max(tabulate(match(v,uniqr)))]

+ }

>

> #Create the vector with characters

> charv<-c("a","b","c","t","a","c","r","a","c","t","z","r","v","t","u","e","t")

>

> #Calculate the mode using the user function

> result<-getmode(charv)

> print(result)

[1] "t"

-----------------------------------------------------------------------------------------------------

**9.Write a function to filter only data belongs to ‘setosa’ in species of Iris dataset.( using dplyr package)**

install.packages("dplyr")

library(dplyr)

filter\_function <- function(dataset, filter\_text) {

eval(rlang::parse\_expr(filter\_text)) %>%

nrow() %>%

paste0( "Number of rows satisfying the filter: ", .)

}

filter\_function(iris, "filter(dataset, Sepal.Length > 5 & Species == 'setosa' )" )

Output:

> library(dplyr)

> filter\_function <- function(dataset, filter\_text) {

+ eval(rlang::parse\_expr(filter\_text)) %>%

+ nrow() %>%

+ paste0( "Number of rows satisfying the filter: ", .)

+ }

>

> filter\_function(iris, "filter(dataset, Sepal.Length > 5 & Species == 'setosa' )" )

[1] "Number of rows satisfying the filter: 22"

-------------------------------------------------------------------------------------------------------

**10.Create a new column for iris dataset with the name of Means\_of\_obs, which contains mean value of each row.( using dplyr package)**

data("iris")

z=apply(iris[,1:4],1,mean)

z

iris$Means\_of\_obs <-apply(iris[,1:4],1,mean)

iris

Output:

> z=apply(iris[,1:4],1,mean)

> z

[1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400

[11] 2.700 2.500 2.325 2.125 2.800 3.000 2.750 2.575 2.875 2.675

[21] 2.675 2.675 2.350 2.650 2.575 2.450 2.600 2.600 2.550 2.425

[31] 2.425 2.675 2.725 2.825 2.425 2.400 2.625 2.500 2.225 2.550

[41] 2.525 2.100 2.275 2.675 2.800 2.375 2.675 2.350 2.675 2.475

[51] 4.075 3.900 4.100 3.275 3.850 3.575 3.975 2.900 3.850 3.300

[61] 2.875 3.650 3.300 3.775 3.350 3.900 3.650 3.400 3.600 3.275

[71] 3.925 3.550 3.800 3.700 3.725 3.850 3.950 4.100 3.725 3.200

[81] 3.200 3.150 3.400 3.850 3.600 3.875 4.000 3.575 3.500 3.325

[91] 3.425 3.775 3.400 2.900 3.450 3.525 3.525 3.675 2.925 3.475

[101] 4.525 3.875 4.525 4.150 4.375 4.825 3.400 4.575 4.200 4.850

[111] 4.200 4.075 4.350 3.800 4.025 4.300 4.200 5.100 4.875 3.675

[121] 4.525 3.825 4.800 3.925 4.450 4.550 3.900 3.950 4.225 4.400

[131] 4.550 5.025 4.250 3.925 3.925 4.775 4.425 4.200 3.900 4.375

[141] 4.450 4.350 3.875 4.550 4.550 4.300 3.925 4.175 4.325 3.950

Means\_of\_obs

1 2.550

2 2.375

3 2.350

4 2.350

5 2.550

6 2.850

7 2.425

8 2.525

9 2.225

10 2.400

11 2.700

12 2.500

13 2.325

14 2.125

15 2.800

16 3.000

17 2.750

18 2.575

19 2.875

20 2.675

21 2.675

22 2.675

23 2.350

24 2.650

25 2.575

26 2.450

27 2.600

28 2.600

29 2.550

30 2.425

31 2.425

32 2.675

33 2.725

34 2.825

35 2.425

36 2.400

37 2.625

38 2.500

39 2.225

40 2.550

41 2.525

42 2.100

43 2.275

44 2.675

45 2.800

46 2.375

47 2.675

48 2.350

49 2.675

50 2.475

51 4.075

52 3.900

53 4.100

54 3.275

55 3.850

56 3.575

57 3.975

58 2.900

59 3.850

60 3.300

61 2.875

62 3.650

63 3.300

64 3.775

65 3.350

66 3.900

67 3.650

68 3.400

69 3.600

70 3.275

71 3.925

72 3.550

73 3.800

74 3.700

75 3.725

76 3.850

77 3.950

78 4.100

79 3.725

80 3.200

81 3.200

82 3.150

83 3.400

84 3.850

85 3.600

86 3.875

87 4.000

88 3.575

89 3.500

90 3.325

91 3.425

92 3.775

93 3.400

94 2.900

95 3.450

96 3.525

97 3.525

98 3.675

99 2.925

100 3.475

101 4.525

102 3.875

103 4.525

104 4.150

105 4.375

106 4.825

107 3.400

108 4.575

109 4.200

110 4.850

111 4.200

112 4.075

113 4.350

114 3.800

115 4.025

116 4.300

117 4.200

118 5.100

119 4.875

120 3.675

121 4.525

122 3.825

123 4.800

124 3.925

125 4.450

126 4.550

127 3.900

128 3.950

129 4.225

130 4.400

131 4.550

132 5.025

133 4.250

134 3.925

135 3.925

136 4.775

137 4.425

138 4.200

139 3.900

140 4.375

141 4.450

142 4.350

143 3.875

144 4.550

145 4.550

146 4.300

147 3.925

148 4.175

149 4.325

150 3.950

--------------------------------------------------------------------------------------------------------------------

**11.Filter data for the “versicolor” and get only ‘sepel\_length’ and Sepel \_width’ columns.( using dplyr package)**

library(dplyr)

iris %>%

select(Sepal.Width,Sepal.Length,Species) %>%

filter(Species=="versicolor")

Output:

> library(dplyr)

> iris %>%

+ select(Sepal.Width,Sepal.Length,Species) %>%

+ filter(Species=="versicolor")

Sepal.Width Sepal.Length Species

1 3.2 7.0 versicolor

2 3.2 6.4 versicolor

3 3.1 6.9 versicolor

4 2.3 5.5 versicolor

5 2.8 6.5 versicolor

6 2.8 5.7 versicolor

7 3.3 6.3 versicolor

8 2.4 4.9 versicolor

9 2.9 6.6 versicolor

10 2.7 5.2 versicolor

11 2.0 5.0 versicolor

12 3.0 5.9 versicolor

13 2.2 6.0 versicolor

14 2.9 6.1 versicolor

15 2.9 5.6 versicolor

16 3.1 6.7 versicolor

17 3.0 5.6 versicolor

18 2.7 5.8 versicolor

19 2.2 6.2 versicolor

20 2.5 5.6 versicolor

21 3.2 5.9 versicolor

22 2.8 6.1 versicolor

23 2.5 6.3 versicolor

24 2.8 6.1 versicolor

25 2.9 6.4 versicolor

26 3.0 6.6 versicolor

27 2.8 6.8 versicolor

28 3.0 6.7 versicolor

29 2.9 6.0 versicolor

30 2.6 5.7 versicolor

31 2.4 5.5 versicolor

32 2.4 5.5 versicolor

33 2.7 5.8 versicolor

34 2.7 6.0 versicolor

35 3.0 5.4 versicolor

36 3.4 6.0 versicolor

37 3.1 6.7 versicolor

38 2.3 6.3 versicolor

39 3.0 5.6 versicolor

40 2.5 5.5 versicolor

41 2.6 5.5 versicolor

42 3.0 6.1 versicolor

43 2.6 5.8 versicolor

44 2.3 5.0 versicolor

45 2.7 5.6 versicolor

46 3.0 5.7 versicolor

47 2.9 5.7 versicolor

48 2.9 6.2 versicolor

49 2.5 5.1 versicolor

50 2.8 5.7 versicolor

---------------------------------------------------------------------------------------------------------

**12.create below plots for the mtcars also write your inferences for each and every plot (use ggplot package) Use Different ( Size , Colour )**

scatter plot

boxplot

histogram

line graph

bar graph

install.packages ("ggplot2")

library(ggplot2)

#Load the mtcars Dataset

data(mtcars)

#view first six rows of mtcars dataset

head(mtcars)

#summarize mtcars dataset

summary(mtcars)

#display rows and columns

dim(mtcars)

#display column names

names(mtcars)

#create histogram of values for mpg

hist(mtcars$mpg,col='steelblue',main='Histogram',

xlab='mpg',ylab='Frequency')

#create boxplot of values for mpg

boxplot(mtcars$mpg,main='Distribution of mpg values',

ylab='mpg',col='steelblue',border='black')

#create scatterplot of mpg vs. wt

plot(mtcars$mpg, mtcars$wt,

col='steelblue',

main='Scatterplot',

xlab='mpg',

ylab='wt',

pch=19)

#Create Bar Plotting

barplot(table(mtcars$carb), col="green")

#Create Line Plotting

v<-c(17,25,38,13,41)

plot(v,type="o",

col="red",xlab="mgp",ylab="wt",main="Line Graph")

Output:

> data(mtcars)

> head(mtcars)

mpg cyl disp hp

Mazda RX4 21.0 6 160 110

Mazda RX4 Wag 21.0 6 160 110

Datsun 710 22.8 4 108 93

Hornet 4 Drive 21.4 6 258 110

Hornet Sportabout 18.7 8 360 175

Valiant 18.1 6 225 105

drat wt qsec vs

Mazda RX4 3.90 2.620 16.46 0

Mazda RX4 Wag 3.90 2.875 17.02 0

Datsun 710 3.85 2.320 18.61 1

Hornet 4 Drive 3.08 3.215 19.44 1

Hornet Sportabout 3.15 3.440 17.02 0

Valiant 2.76 3.460 20.22 1

am gear carb

Mazda RX4 1 4 4

Mazda RX4 Wag 1 4 4

Datsun 710 1 4 1

Hornet 4 Drive 0 3 1

Hornet Sportabout 0 3 2

Valiant 0 3 1

> summary(mtcars)

mpg cyl

Min. :10.40 Min. :4.000

1st Qu.:15.43 1st Qu.:4.000

Median :19.20 Median :6.000

Mean :20.09 Mean :6.188

3rd Qu.:22.80 3rd Qu.:8.000

Max. :33.90 Max. :8.000

disp hp

Min. : 71.1 Min. : 52.0

1st Qu.:120.8 1st Qu.: 96.5

Median :196.3 Median :123.0

Mean :230.7 Mean :146.7

3rd Qu.:326.0 3rd Qu.:180.0

Max. :472.0 Max. :335.0

drat wt

Min. :2.760 Min. :1.513

1st Qu.:3.080 1st Qu.:2.581

Median :3.695 Median :3.325

Mean :3.597 Mean :3.217

3rd Qu.:3.920 3rd Qu.:3.610

Max. :4.930 Max. :5.424

qsec vs

Min. :14.50 Min. :0.0000

1st Qu.:16.89 1st Qu.:0.0000

Median :17.71 Median :0.0000

Mean :17.85 Mean :0.4375

3rd Qu.:18.90 3rd Qu.:1.0000

Max. :22.90 Max. :1.0000

am gear

Min. :0.0000 Min. :3.000

1st Qu.:0.0000 1st Qu.:3.000

Median :0.0000 Median :4.000

Mean :0.4062 Mean :3.688

3rd Qu.:1.0000 3rd Qu.:4.000

Max. :1.0000 Max. :5.000

carb

Min. :1.000

1st Qu.:2.000

Median :2.000

Mean :2.812

3rd Qu.:4.000

Max. :8.000

> dim(mtcars)

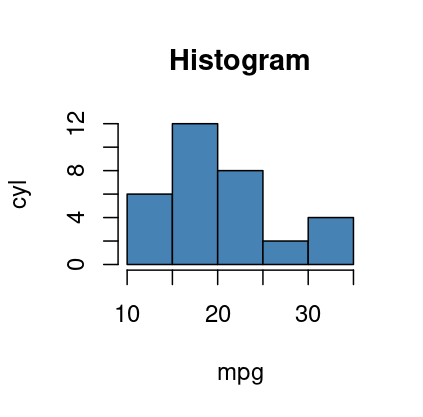
[1] 32 11

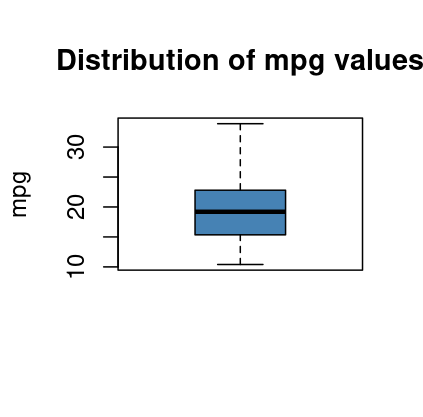
> names(mtcars)

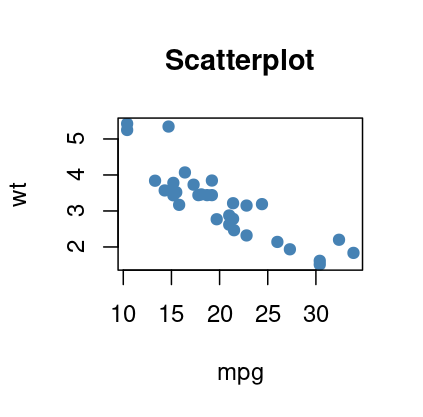
[1] "mpg" "cyl" "disp" "hp"

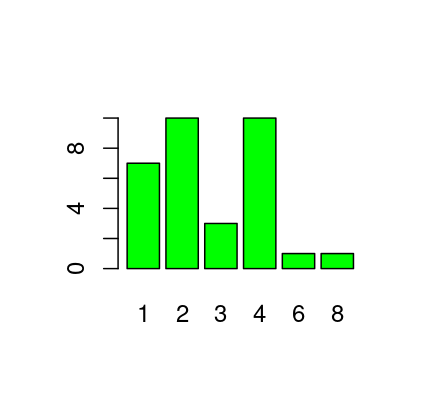
[5] "drat" "wt" "qsec" "vs"

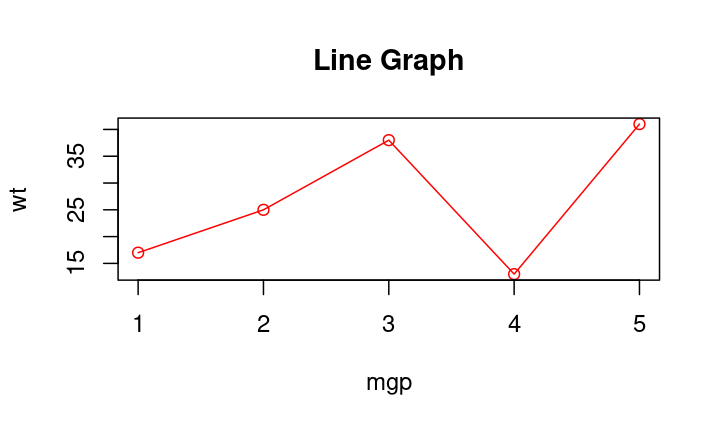
[9] "am" "gear" "carb"











-----------------------------------------------------------------------------------------------------------------------------