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

Ans: # Function to print sequence with given differences

print\_sequence <- function(start, end, differences) {

sequence <- start

current <- start

for (diff in differences) {

while (current < end) {

current <- current + diff

if (current <= end) {

sequence <- c(sequence, current)

}

}

current <- start

}

print(sequence)

}

# Define the range and differences

start <- 1

end <- 50

differences <- c(3, 5, 10)

# Print the sequence

print\_sequence(start, end, differences)

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

Ans:

In R, there are several different data objects that you can use. Here are some commonly used data objects along with their syntax and examples:

1. Vectors:
   * Syntax: **vector\_name <- c(element1, element2, ...)**
   * Example: numbers <- c(1, 2, 3, 4, 5)
   * fruits <- c("apple", "banana", "orange")
2. Matrices:

* Syntax: **matrix\_name <- matrix(data, nrow, ncol)**
* Example: my\_matrix <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)

1. Arrays:

* Syntax: **array\_name <- array(data, dim = c(dim1, dim2, ...))**
* Example: my\_array <- array(c(1, 2, 3, 4, 5, 6), dim = c(2, 3))

1. Lists:

* Syntax: **list\_name <- list(element1, element2, ...)**
* Example: my\_list <- list(name = "John", age = 25, city = "New York")

1. Data Frames:

* Syntax: **data\_frame\_name <- data.frame(column1, column2, ...)**
* Example: my\_data <- data.frame(name = c("John", "Alice", "Bob"), age = c(25, 30, 35))

1. Factors:

* Syntax: **factor\_name <- factor(vector, levels = c(level1, level2, ...))**
* Example: gender <- factor(c("Male", "Female", "Male"), levels = c("Male", "Female"))

1. Characters:

* Syntax: **character\_name <- "string"**
* Example: name <- "John Doe"

1. Numbers:

* Syntax: **number\_name <- numeric\_value**
* Example: age <- 30

1. Logical:

* Syntax: **logical\_name <- TRUE/FALSE**
* Example: is\_student <- TRUE

**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**

# Create a data frame

my\_df <- data.frame(

Name = c("John", "Alice", "Bob", "Emma", "Michael"),

Age = c(25, 30, 35, 28, 32),

City = c("New York", "London", "Paris", "Sydney", "Tokyo")

)

# Print the initial data frame

print(my\_df)

# Fetch a specific row using index

row\_index <- 3

row\_3 <- my\_df[row\_index, ]

print(row\_3)

# Delete a specific row using index

my\_df <- my\_df[-row\_index, ]

print(my\_df)

# Fetch a specific column using index

col\_index <- 2

col\_2 <- my\_df[, col\_index]

print(col\_2)

# Delete a specific column using index

my\_df <- my\_df[, -col\_index]

print(my\_df)

# Add a new column

new\_column <- c(1, 2, 3, 4, 5)

my\_df$NewColumn <- new\_column

print(my\_df)

# Add a new row

new\_row <- data.frame(Name = "Sara", Age = 27, City = "Berlin")

my\_df <- rbind(my\_df, new\_row)

print(my\_df)

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

Num = 45

If (num > 0) {

print('Positive')

} else if (num < 0) {

print('Negative')

} else {

print('Zero')

}

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

# Function to check the type of input

check\_input\_type <- function(input) {

if (is.character(input)) {

print("The input is a character.")

} else if (is.numeric(input)) {

print("The input is numeric.")

} else {

print("The input is a special character.")

}

}

# Take user input

user\_input <- readline(prompt = "Enter a value: ")

# Check the type of input

check\_input\_type(user\_input)

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

Ans: Break and Next are control flow statements used in loops in R. Here's an explanation of the difference between break and next, along with examples:  
  
Break statement:  
  
The break statement is used to exit a loop prematurely. When encountered, it terminates the loop and immediately transfers the program control to the statement following the loop.  
It is commonly used when a certain condition is met within the loop, and you want to exit the loop early.  
Example:

# Example of break statement  
for (i in 1:10) {  
if (i == 5) {  
break # Exit the loop when i equals 5  
}  
print(i)  
}

Output :

[1] 1  
[2] 2  
[3] 3  
[4] 4

In this example, the loop is terminated when the value of i becomes 5 due to the break statement. The numbers 1 to 4 are printed before the loop is exited.

Next statement:  
The next statement is used to skip the remaining code within the current iteration of a loop and proceed to the next iteration.  
It is commonly used when you want to skip certain iterations based on a condition, without terminating the loop entirely.  
Example:  
# Example of next statement  
for (i in 1:5) {  
if (i %% 2 == 0) {  
next # Skip the even numbers  
}  
print(i)  
}

Output:  
[1] 1  
[2] 3  
[3] 5  
In this example, the next statement is used to skip the even numbers (2 and 4) within the loop. Only the odd numbers (1, 3, and 5) are printed.

To summarize, break is used to prematurely exit a loop, while next is used to skip the remaining code within the current iteration and proceed to the next iteration of a loop.

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

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

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

rv = rev(v)

print (rv)

**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’)**

vet = c("a","b", "c","t","a", "c","r","a","c", "t","z","r","v","t","u","e","t")

my\_mode <- function(x) { # Create mode function

unique\_x <- unique(x)

tabulate\_x <- tabulate(match(x, unique\_x))

unique\_x[tabulate\_x == max(tabulate\_x)]

}

my\_mode(vet)

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

# Install and load the dplyr package

install.packages("dplyr")

library(dplyr)

# Function to filter data by species

filter\_species <- function(dataset) {

filtered\_data <- dataset %>%

filter(Species == "setosa")

return(filtered\_data)

}

# Load the Iris dataset from the datasets package

data(iris)

# Filter the data by 'setosa' species

setosa\_data <- filter\_species(iris)

# View the filtered data

print(setosa\_data)

**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)**

# Install and load the dplyr package

install.packages("dplyr")

library(dplyr)

# Load the Iris dataset from the datasets package

data(iris)

# Add a new column with mean values

iris\_with\_means <- iris %>%

mutate(Means\_of\_obs = rowMeans(.[1:4]))

# View the updated dataset

print(iris\_with\_means)

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

# Install and load the dplyr package

install.packages("dplyr")

library(dplyr)

# Load the Iris dataset from the datasets package

data(iris)

# Filter data for "versicolor" species and select columns

versicolor\_data <- iris %>%

filter(Species == "versicolor") %>%

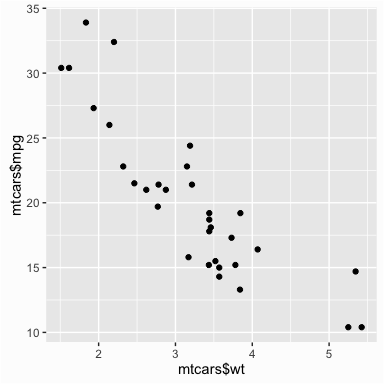
select(Sepal\_Length, Sepal\_Width)

# View the filtered and selected data

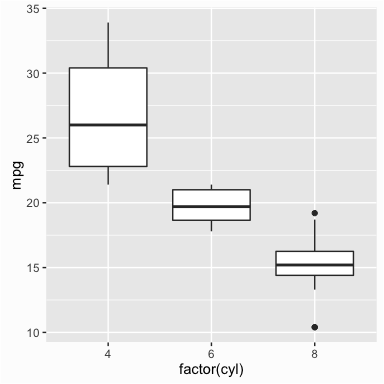
print(versicolor\_data)

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

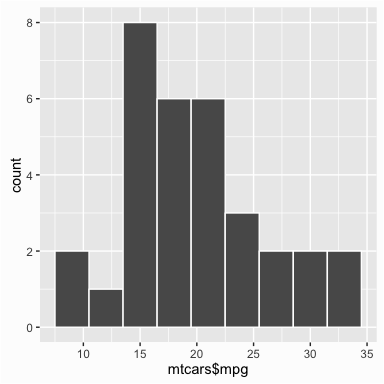
* scatter plot



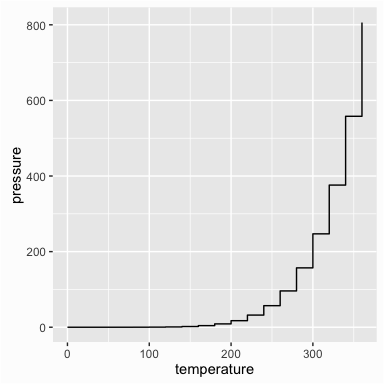
* boxplot



* histogram



* line graph



* bar graph

