**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

|  |  |  |
| --- | --- | --- |
| **Course Code: ITA04** | **Course Name: Statistics with R Programming** | |
| **Branch: CSE** |  | **Academic Year: 2019-2020 (EVEN)** |
| **Date of Exam: 20/1/20** | **Max. Marks: 20M** | **Time: 1 hour** |

**ANSWER ALL THE QUESTIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CLASS TEST 1** | **MARKS** | **COs** | **BT LEVEL** |
| 1 | Write in Detail about various Advanced-Data structures with example | 10 | CO1 | K3 |
| 2 | 1. Write a function called kelvin\_to\_celsius () that takes a temperature in Kelvin and returns that temperature in Celsius (Hint: To convert from Kelvin to Celsius you subtract 273.15) 2. Write suitable R code to compute the mean, median, mode of the following values c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20) | 10 | CO1 | K3 |

**CLASS TEST 1-ANSWERS**

1.

**LIST:**

# Create a list

my\_list <- list(name = "John", age = 25, scores = c(80, 85, 90))

# Access elements in the list

print(my\_list$name)

print(my\_list$age)

print(my\_list$scores)

**Factor:**

# Create a factor

gender <- factor(c("Male", "Female", "Male", "Male", "Female"))

# Print the levels of the factor

print(levels(gender))

# Convert the factor to numeric

gender\_numeric <- as.numeric(gender)

print(gender\_numeric)

**Array:**

# Create an array

my\_array <- array(c(1, 2, 3, 4, 5, 6), dim = c(2, 3))

# Access elements in the array

print(my\_array[1, 2])

print(my\_array[, 3])

**Matrices with Row and Column Names:**

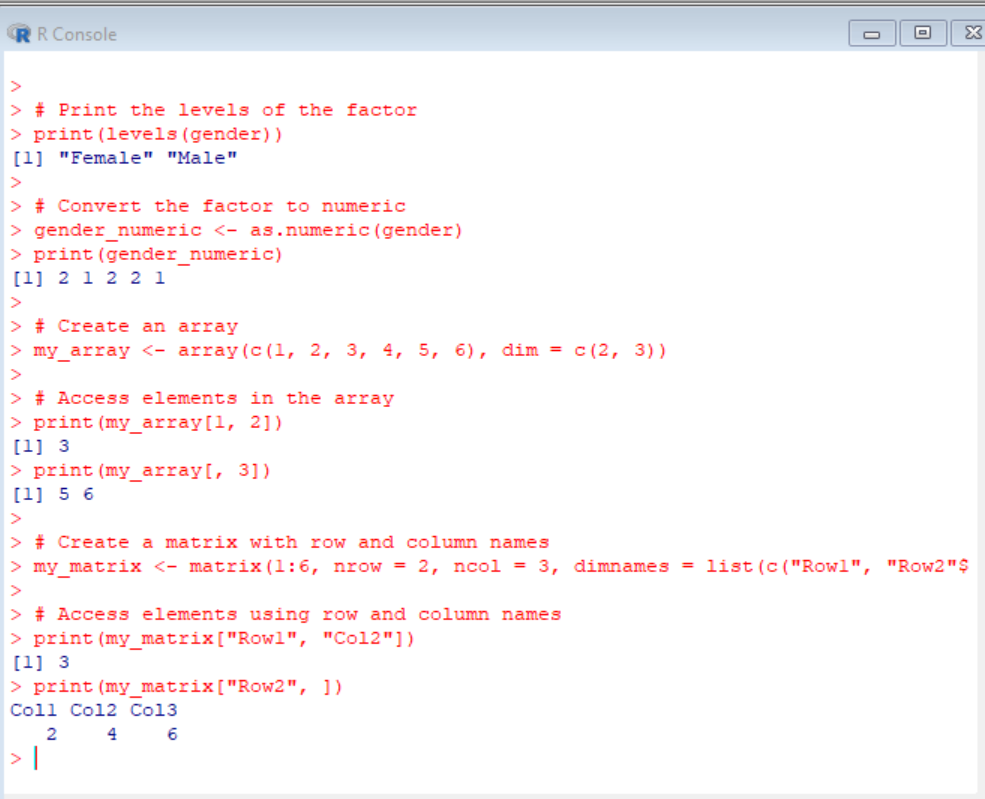
# Create a matrix with row and column names

my\_matrix <- matrix(1:6, nrow = 2, ncol = 3, dimnames = list(c("Row1", "Row2"), c("Col1", "Col2", "Col3")))

# Access elements using row and column names

print(my\_matrix["Row1", "Col2"])

print(my\_matrix["Row2", ])



**2.(i)**

kelvin\_to\_celsius <- function(temperature\_kelvin) {

temperature\_celsius <- temperature\_kelvin - 273.15

return(temperature\_celsius)

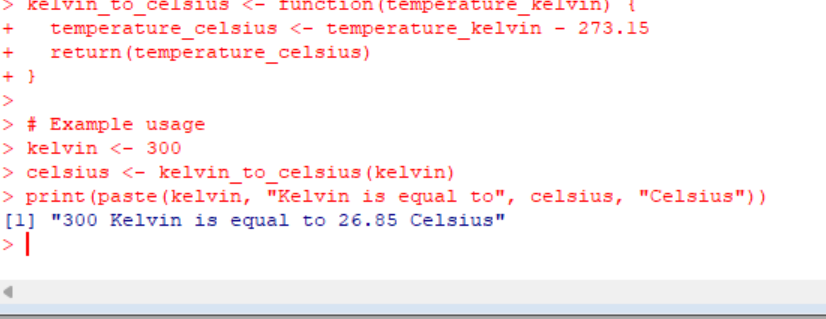
}

# Example usage

kelvin <- 300

celsius <- kelvin\_to\_celsius(kelvin)

print(paste(kelvin, "Kelvin is equal to", celsius, "Celsius"))



**(ii)**

values <- c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

# Compute mean

mean\_value <- mean(values)

print(paste("Mean:", mean\_value))

# Compute median

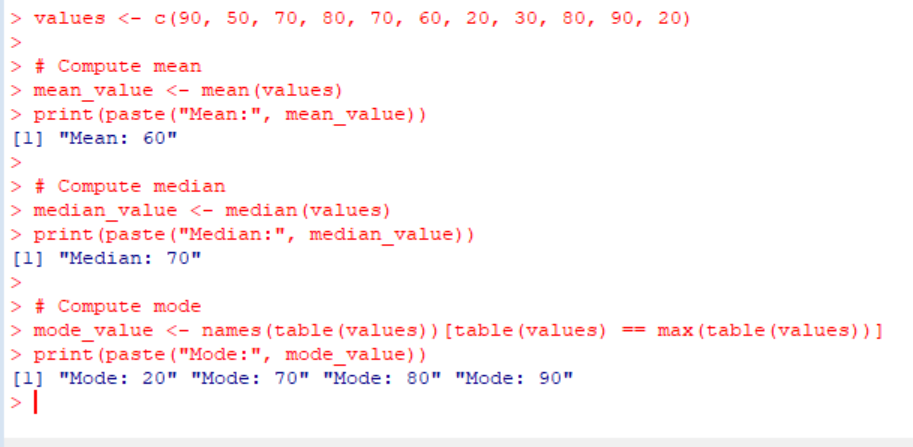
median\_value <- median(values)

print(paste("Median:", median\_value))

# Compute mode

mode\_value <- names(table(values))[table(values) == max(table(values))]

print(paste("Mode:", mode\_value))



**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

|  |  |  |
| --- | --- | --- |
| **Course Code: ITA04** | **Course Name: Statistics with R Programming** | |
| **Branch: CSE** |  | **Academic Year: 2019-2020 (EVEN)** |
| **Date of Exam: 11/2/20** | **Max. Marks: 20M** | **Time: 1 hour** |

**ANSWER ALL THE QUESTIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CLASS TEST 2** | **MARKS** | **COs** | **BT LEVEL** |
| 1 | Write a function to find the factorial of a given number using “for” Loop | 10 | CO2 | K3 |
| 2 | Explain the functioning of apply (), lapply () and sapply () in R program with one example each | 10 | CO2 | K3 |

**CLASS TEST 2-ANSWERS**

**1.** factorial <- function(n) {

result <- 1

if (n < 0) {

stop("Factorial is not defined for negative numbers.")

}

for (i in 1:n) {

result <- result \* i

}

return(result)

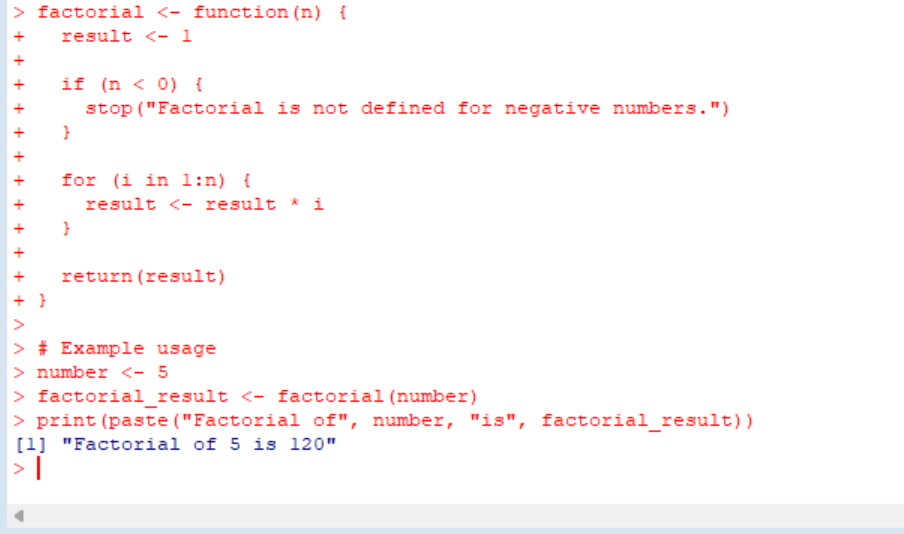
}

# Example usage

number <- 5

factorial\_result <- factorial(number)

print(paste("Factorial of", number, "is", factorial\_result))



2. apply(X, MARGIN, FUN)

# Create a matrix

my\_matrix <- matrix(1:9, nrow = 3)

# Apply column-wise sum using apply()

column\_sum <- apply(my\_matrix, 2, sum)

print(column\_sum)

lapply(X, FUN)

# Create a vector

my\_vector <- c(1, 2, 3, 4, 5)

# Apply squaring using lapply()

squared\_list <- lapply(my\_vector, function(x) x^2)

print(squared\_list)

sapply(X, FUN)

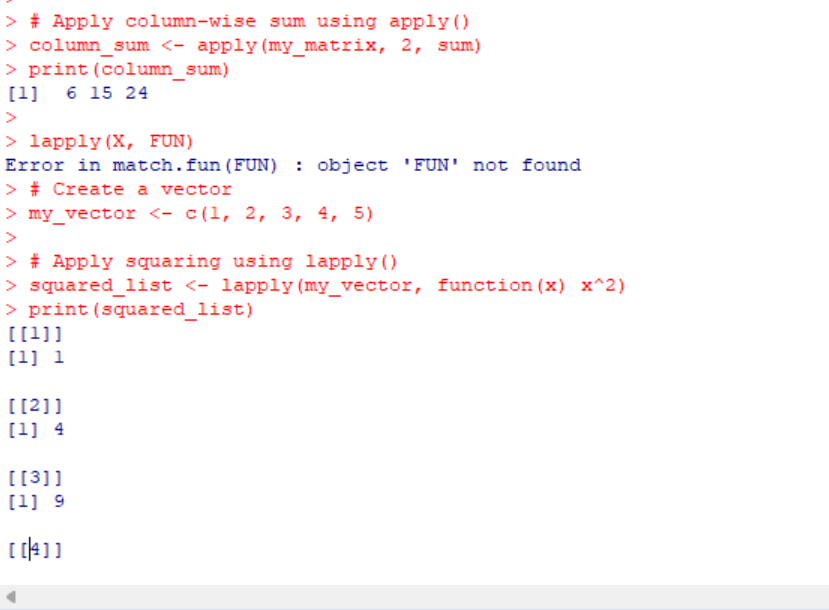
# Create a character vector

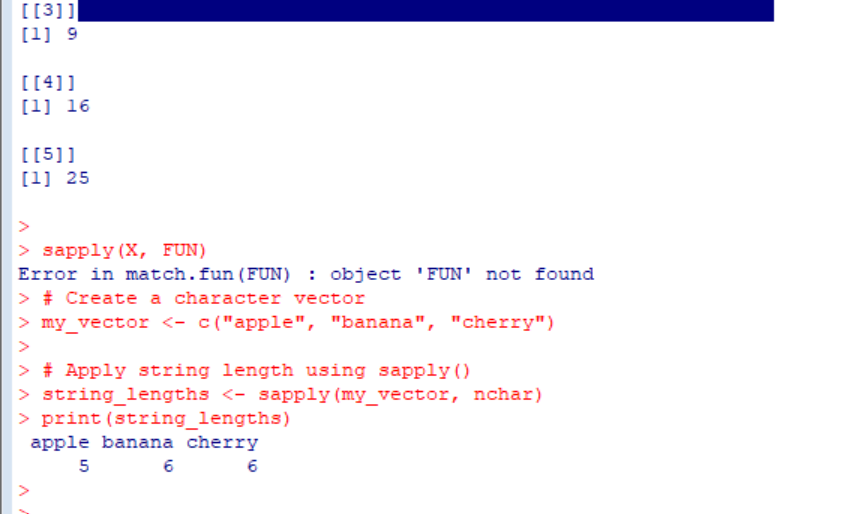
my\_vector <- c("apple", "banana", "cherry")

# Apply string length using sapply()

string\_lengths <- sapply(my\_vector, nchar)

print(string\_lengths)





**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

|  |  |  |
| --- | --- | --- |
| **Course Code: ITA04** | **Course Name: Statistics with R Programming** | |
| **Branch: CSE** |  | **Academic Year: 2019-2020 (EVEN)** |
| **Date of Exam: 2/3/20** | **Max. Marks: 20M** | **Time: 1 hour** |

**ANSWER ALL THE QUESTIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CLASS TEST 3** | **MARKS** | **COs** | **BT LEVEL** |
| 1 | What is data frame and how to create a data frame using the following data:  Height GPA  66 3.80  62 3.78  63 3.88  70 3.72  74 3.69  Write a R program to prepare the inventory. find the mean of height and GPA | 10 | CO3 | K3 |
| 2 | Explain melting and casting data in R with examples | 10 | CO3 | K3 |

**CLASS TEST 3-ANSWERS**

**1**. # Create the data frame

data <- data.frame(

Height = c(66, 62, 63, 70, 74),

GPA = c(3.80, 3.78, 3.88, 3.72, 3.69)

)

# Print the data frame

print(data)

# Calculate the mean of Height

height\_mean <- mean(data$Height)

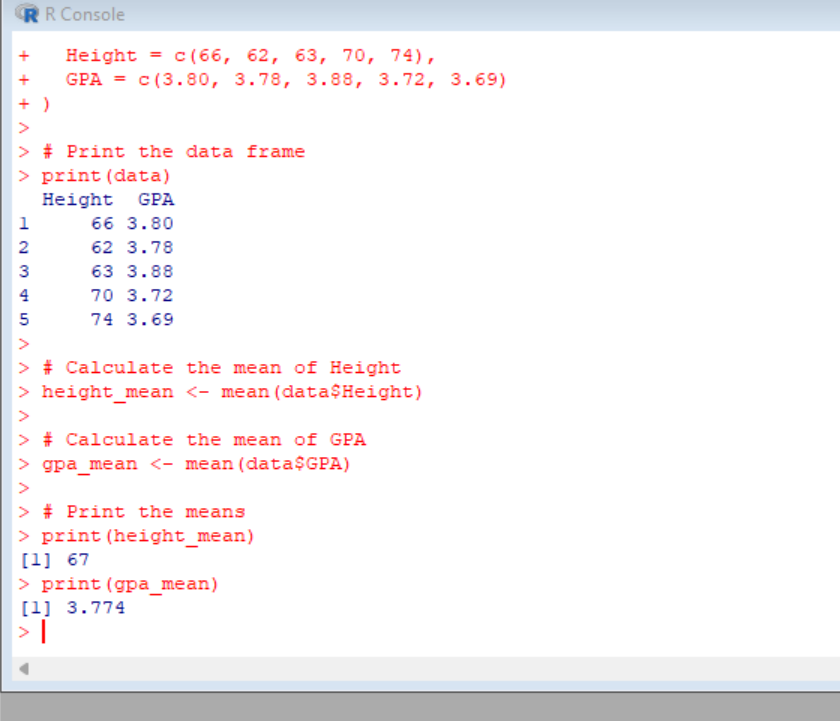
# Calculate the mean of GPA

gpa\_mean <- mean(data$GPA)

# Print the means

print(height\_mean)

print(gpa\_mean)



2. **MELTING DATA**

# Example data frame

data <- data.frame(

ID = c(1, 2, 3),

Height = c(66, 62, 63),

GPA = c(3.80, 3.78, 3.88)

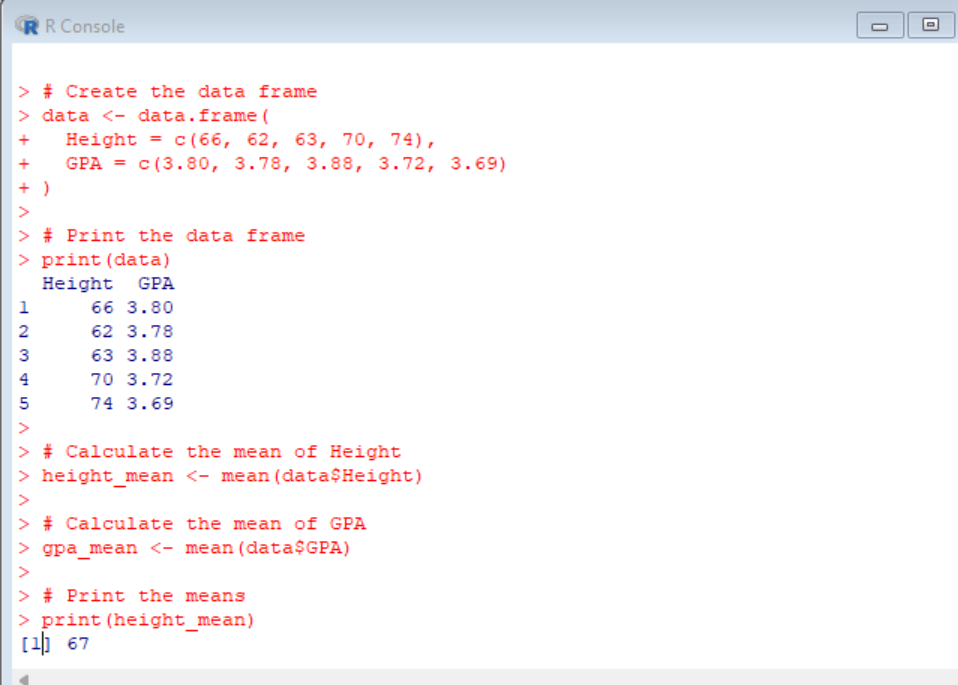
)

# Melting the data frame

melted\_data <- reshape2::melt(data, id.vars = "ID")

# Printing the melted data

print(melted\_data)

****

**CASTING DATA**

# Example melted data frame

melted\_data <- data.frame(

ID = c(1, 2, 3),

variable = c("Height", "Height", "GPA"),

value = c(66, 62, 3.80)

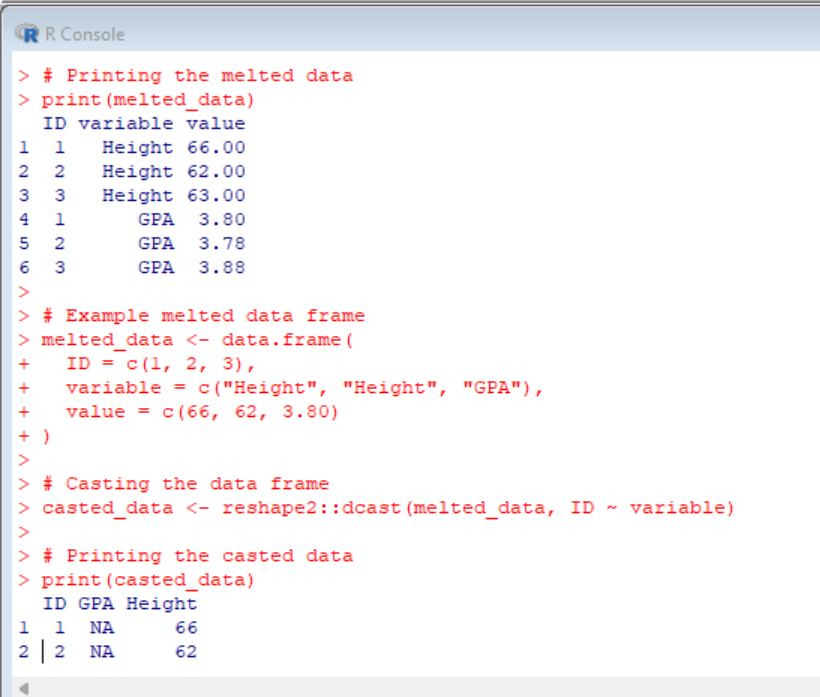
)

# Casting the data frame

casted\_data <- reshape2::dcast(melted\_data, ID ~ variable)

# Printing the casted data

print(casted\_data)



**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

|  |  |  |
| --- | --- | --- |
| **Course Code: ITA04** | **Course Name: Statistics with R Programming** | |
| **Branch: CSE** |  | **Academic Year: 2019-2020 (EVEN)** |
| **Date of Exam: 23/3/20** | **Max. Marks: 20M** | **Time: 1 hour** |

**ANSWER ALL THE QUESTIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CLASS TEST 4** | **MARKS** | **COs** | **BT LEVEL** |
| 1 | Compute the correlation coefficient for the following data and write R code for same  X<-68 ,64 ,75, 50, 64, 80, 75, 40, 55, 64  Y<-62, 58, 68, 45, 81, 60, 68 ,48, 58, 70 | 10 | CO4 | K3 |
| 2 | Explain Skewness and Kurtosis and its types | 10 | CO4 | K3 |

**CLASS TEST 4-ANSWERS**

**1.**

# Given data

X <- c(68, 64, 75, 50, 64, 80, 75, 40, 55, 64)

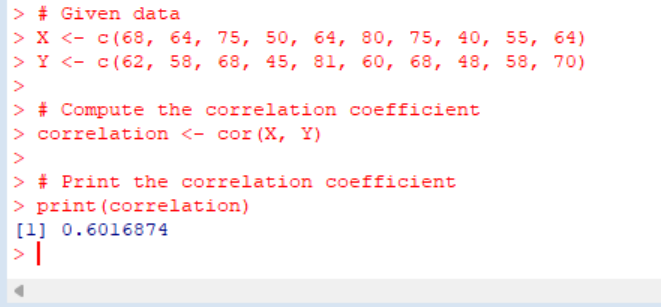
Y <- c(62, 58, 68, 45, 81, 60, 68, 48, 58, 70)

# Compute the correlation coefficient

correlation <- cor(X, Y)

# Print the correlation coefficient

print(correlation)



**2.SKEWNESS:**

Skewness is a statistical numerical method to measure the asymmetry of the distribution or data set. It tells about the position of the majority of data values in the distribution around the mean value.

**example**

# Load the required package

library(moments)

# Example data

data <- c(68, 64, 75, 50, 64, 80, 75, 40, 55, 64)

# Calculate skewness using moments package

skewness\_result <- skewness(data)

# Print the skewness

print(skewness\_result)

**KURTOSIS:**

Kurtosis is a numerical method in statistics that measures the sharpness of the peak in the data distribution.

**Example:**

# Load the required package

library(moments)

# Example data

data <- c(68, 64, 75, 50, 64, 80, 75, 40, 55, 64)

# Calculate kurtosis using moments package

kurtosis\_result <- kurtosis(data)

# Print the kurtosis

print(kurtosis\_result)

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

|  |  |  |
| --- | --- | --- |
| **Course Code: ITA04** | **Course Name: Statistics with R Programming** | |
| **Branch: CSE** |  | **Academic Year: 2019-2020 (EVEN)** |
| **Date of Exam: 16/4/20** | **Max. Marks: 20M** | **Time: 1 hour** |

**ANSWER ALL THE QUESTIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CLASS TEST 5** | **MARKS** | **COs** | **BT LEVEL** |
| 1 | 1. The maximum temperature in Celsius in a week is given as T(35,42,38,25,28,36,40). Draw the bar plot for the given data. Also use legend function to describe hists. 2. Explain the differences between stacked bar plot and bar plot. | 10 | CO5 | K3 |
| 2 | 1. What is Box plot? Explain importance of boxplot with example? 2. Draw a pie chart for the following data Section: I, II, III , IV, V   No.of workers:220,370, 190, 70, 250 | 10 | CO5 | K3 |

**CLASS TEST 5-ANSWERS**

**1.** # Data

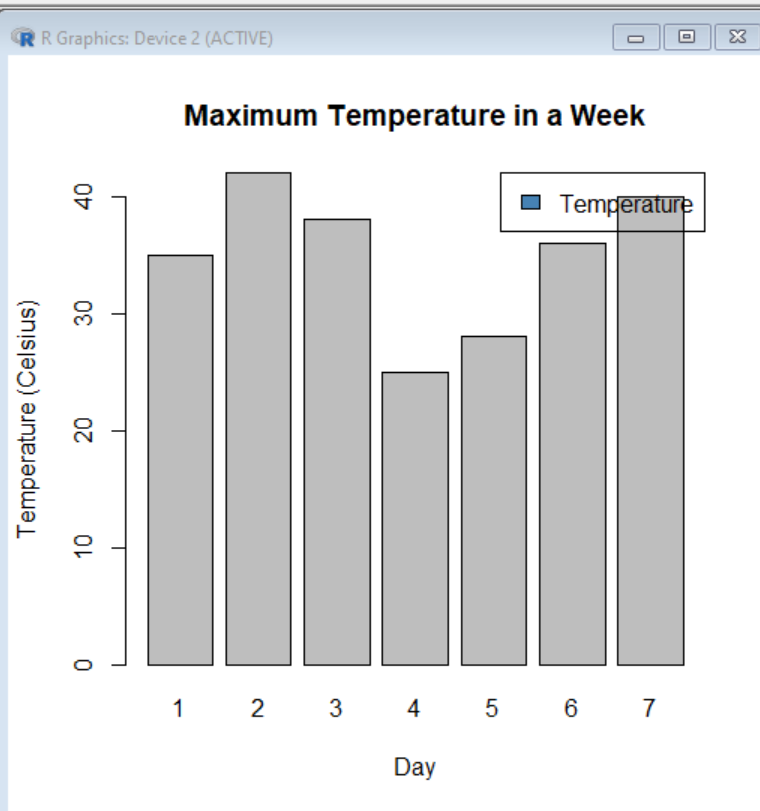
T <- c(35, 42, 38, 25, 28, 36, 40)

# Bar plot

barplot(T, ylim = c(0, max(T)), names.arg = 1:7, xlab = "Day", ylab = "Temperature (Celsius)", main = "Maximum Temperature in a Week")

# Add legend

legend("topright", legend = "Temperature", fill = "steelblue")



(ii)

Differences between stacked bar plot and bar plot:

* Bar Plot: In a bar plot, each bar represents a single category or variable, and the height of the bar corresponds to the magnitude or value of that category. Bar plots are commonly used to display and compare categorical data. Each bar is plotted side by side, and they do not overlap or stack on top of each other.
* Stacked Bar Plot: In a stacked bar plot, each bar represents a single category or variable, similar to a bar plot. However, the height of each bar is divided into segments that represent sub-categories or sub-groups within the main category. Each segment of the bar represents the contribution or proportion of a sub-category to the total value of the main category. Stacked bar plots are useful for showing both the composition of a whole and the individual contributions of sub-categories.

**2. i) Box plot**

also known as a box-and-whisker plot, is a graphical representation of a dataset's summary statistics. It displays the distribution of a continuous variable and provides a visual summary of its central tendency, dispersion, and skewness. A box plot consists of a box, whiskers, and possibly outliers.

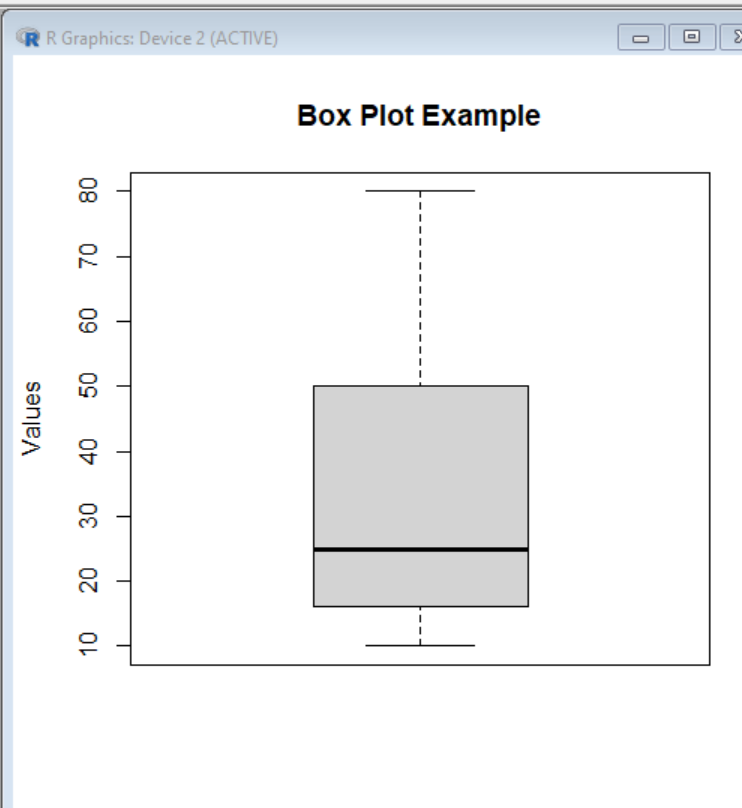
**Example of box plot:**

# Example data

data <- c(10, 12, 15, 16, 20, 22, 25, 30, 40, 50, 60, 70, 80)

# Box plot

boxplot(data, ylab = "Values", main = "Box Plot Example")



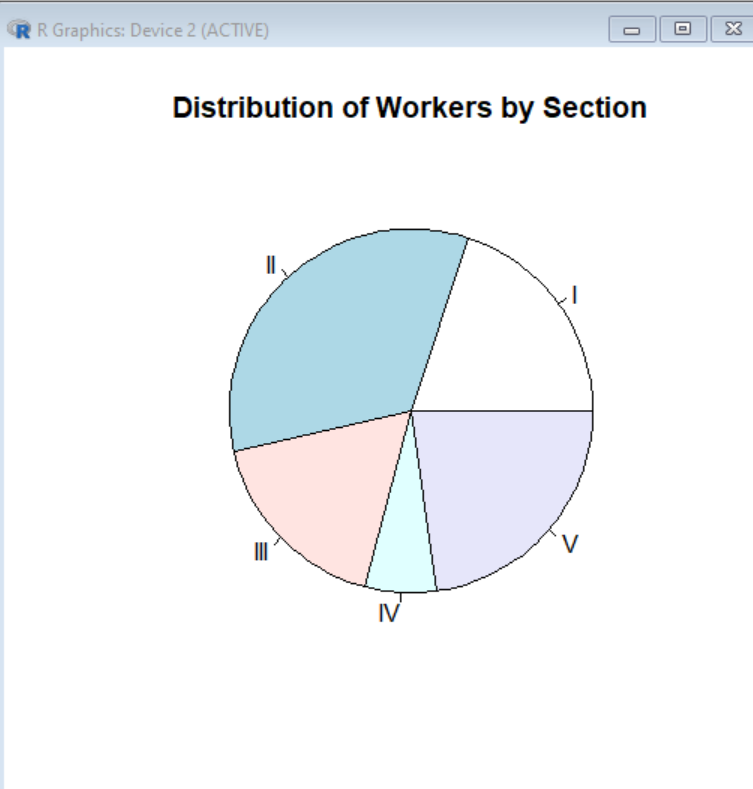
(ii)# Data

section <- c("I", "II", "III", "IV", "V")

workers <- c(220, 370, 190, 70, 250)

# Pie chart

pie(workers, labels = section, main = "Distribution of Workers by Section")



**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

|  |  |  |
| --- | --- | --- |
| **Course Code: ITA04** | **Course Name: Statistics with R Programming** | |
| **Branch: CSE** |  | **Academic Year: 2019-2020 (EVEN)** |
| **Date of Exam: 4/5/2020** | **Max. Marks: 100 M** | **Time: 3 hours** |

**ANSWER ALL THE QUESTIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **MODEL EXAMINATION** | **MARKS** | **COs** | **BT LEVEL** |
| 1 | Explain in detail about the various Advanced Data Structures in R. | 10 | CO1 | K3 |
| 2 | Write a function in R programming to find a factorial of a given number. | 10 | CO1 | K3 |
| 3 | Create First DataFrame with variables   * surname * nationality   Create Second DataFrame with variables   * surname * movies   The common key variable is surname. How to merge both data and check if the dimensionality is 7x3 | 10 | CO2 | K3 |
| 4 | How to edit and read from a file in R? Explain in detail. | 10 | CO2 | K3 |
| 5 | What is function and recursion? Write R code function to generate first n terms of a Fibonacci series | 10 | CO3 | K3 |
| 6 | Explain melting and casting with example | 10 | CO3 | K3 |
| 7 | Explain the various high level plotting functions with suitable example | 10 | CO4 | K3 |
| 8 | Compute the correlation coefficient for the following data and write R code for same  X<-68 ,64 ,75, 50, 64, 80, 75, 40, 55, 64  Y<-62, 58, 68, 45, 81, 60, 68 ,48, 58, 70 | 10 | CO4 | K3 |
| 9 | Explain Skewness and Kurtosis and its types | 10 | CO5 | K3 |
| 10 | What is Box plot? Explain importance of boxplot with example? b) Draw a pie chart for the following data  Section: I, II, III , IV, V  No.of workers:220,370, 190, 70, 250 | 10 | CO5 | K3 |

**MODEL EXAMINATION KEY**

1A) data.table:

The data.table package provides an enhanced version of R's built-in data.frame object. It is designed to handle large datasets efficiently and is known for its speed and memory optimization.

It offers fast aggregation, joins, and indexing operations, making it suitable for working with large datasets.

data.table also provides additional features such as key-based subsetting, updates by reference, and optimized memory usage.

dplyr:

Although dplyr is primarily a data manipulation package, it introduces a few advanced data structures for data analysis tasks.

The tbl\_df (tibble data frame) is an alternative to R's default data.frame. It provides additional features like improved printing, strict data typing, and enhanced compatibility with other dplyr functions.

dplyr also introduces the concept of grouped data frames (grouped\_df), which allows efficient computation by group using functions like group\_by, summarize, and mutate.

igraph:

The igraph package is widely used for analyzing and visualizing graphs and networks in R.

It provides efficient data structures and algorithms for creating, manipulating, and analyzing graphs.

The core data structure in igraph is the graph object, which can represent directed or undirected graphs with attributes and edge weights.

igraph offers various graph-related functions like shortest path calculations, centrality measures, community detection, and visualization capabilities.

tidyverse:

The tidyverse is a collection of R packages, including dplyr, ggplot2, tidyr, and more.

While not specifically focused on advanced data structures, tidyverse provides a consistent and integrated set of tools for data manipulation, visualization, and analysis.

By combining the functionalities of different packages within tidyverse, you can effectively work with complex data structures and perform advanced data transformations.

Rcpp:

The Rcpp package allows you to integrate C++ code into R, enabling the use of advanced data structures available in C++.

With Rcpp, you can create and manipulate C++ data structures such as vectors, matrices, lists, and more within R.

This package is particularly useful when performance is a concern, as C++ implementations can often be faster than pure R implementations.

2A) factorial <- function(n) {

if (n < 0) {

stop("Factorial is not defined for negative numbers.")

} else if (n == 0 || n == 1) {

return(1)

} else {

result <- 1

for (i in 2:n) {

result <- result \* i

}

return(result)

}

}

output : 120

3 A )

# Create the first data frame

df1 <- data.frame(

surname = c("Smith", "Johnson", "Brown"),

nationality = c("USA", "UK", "Canada"),

stringsAsFactors = FALSE

)

# Create the second data frame

df2 <- data.frame(

surname = c("Smith", "Johnson", "Williams"),

movies = c(5, 3, 8),

stringsAsFactors = FALSE

)

# Merge the data frames based on the common key variable 'surname'

merged\_df <- merge(df1, df2, by = "surname", all = TRUE)

# Check the resulting dimensionality

dim(merged\_df)

output : 7 3

4A) To edit and read from a file in R, you can use various functions and techniques depending on the specific requirements. Here's a detailed explanation of how to accomplish these tasks in R:

Reading from a file:

R provides several functions to read data from different types of files, such as text files, CSV files, Excel files, etc.

The most commonly used function is read.table() or its variant read.csv() for reading data from text or CSV files.

Here's an example of how to read data from a text file named "data.txt" with comma-separated values:

R

Copy code

# Read data from a text file

data <- read.table("data.txt", header = TRUE, sep = ",")

In this example, read.table() reads the data from "data.txt" and stores it in the variable data. The header = TRUE argument indicates that the file contains a header row, and sep = "," specifies that the values are comma-separated.

Similarly, you can use read.csv() function to directly read data from a CSV file.

Editing a file:

To edit a file in R, you typically read the contents of the file into a data structure, make the necessary changes, and then write the updated data back to the file.

Here's a step-by-step process to edit a file:

Read the file into a data structure (e.g., data frame) using appropriate functions like read.table(), read.csv(), etc.

Make the necessary changes to the data structure. For example, you can modify values, add or remove rows/columns, or apply transformations.

Write the updated data back to the file using functions like write.table(), write.csv(), etc.

Here's an example of how to edit a CSV file by modifying values in R:

R

Copy code

# Read data from a CSV file

data <- read.csv("data.csv", header = TRUE)

# Make changes to the data

data$new\_column <- data$old\_column \* 2 # Multiply a column by 2

# Write the updated data back to the file

write.csv(data, "data.csv", row.names = FALSE)

In this example, the read.csv() function is used to read the data from "data.csv" into the variable data. The desired changes are made to the data by creating a new column (new\_column) based on an existing column (old\_column) multiplied by 2. Finally, the write.csv() function is used to write the updated data back to the file, overwriting the existing contents. The row.names = FALSE argument ensures that row numbers are not included in the output file.

You can use similar techniques to edit and update other types of files by appropriately choosing the read and write functions for those file formats.

5A)In programming, a function is a reusable block of code that performs a specific task. It helps in organizing code, promoting reusability, and modularizing the program. Functions take input parameters (arguments), perform some operations, and may return a value.

Recursion, on the other hand, is a programming technique where a function calls itself within its own body. This allows the function to solve a problem by breaking it down into smaller subproblems. Recursion involves two components: a base case (or termination condition) that stops the recursive calls, and a recursive case where the function calls itself with a modified input.

To generate the first n terms of a Fibonacci series using recursion in R, you can define a function that calls itself to calculate the Fibonacci numbers.

fibonacci <- function(n) {

if (n <= 0) {

stop("Invalid input. n must be a positive integer.")

}

if (n == 1) {

return(0) # Base case for the first Fibonacci number

} else if (n == 2) {

return(1) # Base case for the second Fibonacci number

} else {

# Recursive case: Calculate the nth Fibonacci number

return(fibonacci(n - 1) + fibonacci(n - 2))

}

}

# Generate the first n terms of the Fibonacci series

n <- 10 # Number of terms to generate

fib\_series <- numeric(n) # Initialize an empty vector

for (i in 1:n) {

fib\_series[i] <- fibonacci(i)

}

print(fib\_series)

output : [1] 0 1 1 2 3 5 8 13 21 34

6A) In R, melting and casting are operations commonly used for reshaping data frames. These operations are typically performed using the reshape2 package or the tidyverse ecosystem, which includes the tidyr package. Let's explore melting and casting in more detail, along with examples:

Melting:

Melting a data frame involves transforming it from a wide format to a long format by converting columns into rows.

The melt() function from the reshape2 package or the pivot\_longer() function from the tidyr package can be used to melt a data frame.

Melting is useful when you want to gather multiple variables into a single column while retaining identifiers or when you need to reshape data for further analysis.

Here's an example of melting a data frame using tidyr:

library(tidyverse)

# Create a sample data frame

data <- data.frame(

id = 1:3,

apples = c(5, 3, 8),

oranges = c(2, 4, 6)

)

# Melting the data frame

melted\_data <- data %>% pivot\_longer(-id, names\_to = "fruit", values\_to = "quantity")

print(melted\_data)

Casting:

Casting a data frame involves transforming it from a long format to a wide format by converting rows into columns.

The dcast() function from the reshape2 package or the pivot\_wider() function from the tidyr package can be used to cast a data frame.

Casting is useful when you want to reorganize data into a more compact form or when you need to reshape data for presentation purposes.

Here's an example of casting a data frame using tidyr:

library(tidyverse)

# Create a sample melted data frame

melted\_data <- data.frame(

id = c(1, 1, 2, 2, 3, 3),

fruit = c("apples", "oranges", "apples", "oranges", "apples", "oranges"),

quantity = c(5, 2, 3, 4, 8, 6)

)

# Casting the data frame

casted\_data <- melted\_data %>% pivot\_wider(names\_from = fruit, values\_from = quantity)

print(casted\_data)

7A) In R, there are several high-level plotting functions available that make it easy to create a wide range of visualizations. These functions are part of the base R graphics system and are often used for exploratory data analysis and data visualization. Let's explore some of these high-level plotting functions with suitable examples:

plot():

The plot() function is a versatile function that can create various types of plots, including scatter plots, line plots, bar plots, and more.

It is commonly used to explore relationships between variables or to visualize distributions.

Example:

R

Copy code

# Create a scatter plot

x <- 1:10

y <- c(3, 5, 6, 8, 9, 11, 13, 15, 16, 18)

plot(x, y, main = "Scatter Plot", xlab = "X", ylab = "Y")

hist():

The hist() function is used to create histograms, which provide a visual representation of the distribution of a continuous variable.

It divides the data into intervals (bins) and displays the frequency or density of observations in each bin.

Example:

R

Copy code

# Create a histogram

data <- c(5, 6, 7, 8, 9, 10, 10, 11, 11, 12, 13, 14, 15)

hist(data, main = "Histogram", xlab = "Values", ylab = "Frequency", col = "lightblue")

boxplot():

The boxplot() function creates box-and-whisker plots, which provide a visual summary of the distribution of a continuous variable across different categories or groups.

It displays the median, quartiles, and potential outliers of the data.

Example:

# Create a boxplot

data <- list(

Group1 = c(10, 12, 14, 16, 18),

Group2 = c(8, 10, 12, 14, 16),

Group3 = c(6, 8, 10, 12, 14)

)

boxplot(data, main = "Boxplot", xlab = "Groups", ylab = "Values", col = c("lightblue", "lightgreen", "lightyellow"))

barplot():

The barplot() function is used to create bar plots or bar charts, which are commonly used to display categorical or discrete data.

It represents the values of different categories as rectangular bars of varying

example :

# Create a bar plot

categories <- c("A", "B", "C", "D", "E")

values <- c(10, 15, 7, 12, 9)

barplot(values, names.arg = categories, main = "Bar Plot", xlab = "Categories", ylab = "Values", col = "lightblue")

8A) To compute the correlation coefficient between two variables, X and Y, you can use the cor() function in R. Here's the code to compute the correlation coefficient for the given data:

R

Copy code

X <- c(68, 64, 75, 50, 64, 80, 75, 40, 55, 64)

Y <- c(62, 58, 68, 45, 81, 60, 68, 48, 58, 70)

correlation <- cor(X, Y)

print(correlation)

In this code, we first define the two variables, X and Y, using the provided data. Then, we use the cor() function to calculate the correlation coefficient between X and Y. The resulting correlation coefficient is stored in the correlation variable.

output : [1] 0.6206415

9A)Skewness and kurtosis are statistical measures that provide insights into the shape and distribution of a dataset. They help in understanding the departure of a distribution from a normal or symmetric distribution. In R, you can calculate skewness and kurtosis using various functions and packages. Let's explore skewness and kurtosis in more detail:

Skewness:

Skewness measures the asymmetry of a distribution.

Positive skewness indicates a longer tail on the right side of the distribution, while negative skewness indicates a longer tail on the left side.

Skewness can help identify the presence of outliers and the departure from a normal distribution.

In R, you can calculate skewness using the skewness() function from the moments package or the e1071 package. Here's an example:

library(moments)

data <- c(10, 15, 12, 14, 8, 9, 18, 16, 11, 13)

skew <- skewness(data)

print(skew)

Kurtosis:

Kurtosis measures the tail-heaviness or the peakedness of a distribution.

High kurtosis indicates heavy tails or a more peaked distribution, while low kurtosis indicates light tails or a flatter distribution.

Kurtosis can provide insights into the presence of outliers and the departure from a normal distribution.

In R, you can calculate kurtosis using the kurtosis() function from the moments package or the e1071 package. Here's an example:

library(moments)

data <- c(10, 15, 12, 14, 8, 9, 18, 16, 11, 13)

kurt <- kurtosis(data)

print(kurt)

10A)a) Box plot:

A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset. It displays the summary statistics of the data, including the median, quartiles, and potential outliers. The box plot consists of a rectangle (the "box") and two lines (the "whiskers") that extend from the box. It is commonly used to compare the distribution of a variable across different categories or groups.

The importance of box plots includes:

Visualization of distribution: Box plots provide a visual summary of the distribution of data, allowing you to quickly assess the central tendency, spread, and skewness of the data.

Comparison between groups: Box plots are useful for comparing the distribution of a variable across different groups or categories. By plotting multiple box plots side by side, you can easily compare the medians, quartiles, and variability of the data between groups.

Identification of outliers: Box plots help in identifying potential outliers in the data. Outliers are data points that significantly deviate from the overall pattern and may indicate unusual observations or measurement errors.

Assessment of symmetry and skewness: Box plots provide insights into the symmetry or skewness of the data distribution. The length and position of the whiskers and the position of the median relative to the box can indicate the degree of skewness.

Compact representation: Box plots offer a compact representation of the data distribution, allowing you to convey essential information in a concise and visually appealing manner.

b) Drawing a pie chart in R:

To draw a pie chart in R, you can use the pie() function. Here's an example based on the provided data:

R

Copy code

# Create the data

section <- c("I", "II", "III", "IV", "V")

no\_of\_workers <- c(220, 370, 190, 70, 250)

# Create a pie chart

pie(no\_of\_workers, labels = section, main = "Number of Workers by Section")

In this example, the pie() function creates a pie chart based on the no\_of\_workers vector. The labels argument specifies the labels for each section of the pie chart. The main

argument sets the title of the chart.

When you run this code, it will generate a

# Create the data

section <- c("I", "II", "III", "IV", "V")

no\_of\_workers <- c(220, 370, 190, 70, 250)

# Create a pie chart

pie(no\_of\_workers, labels = section, main = "Number of Workers by Section")

In this example, the pie() function creates a pie chart based on the no\_of\_workers vector. The labels argument specifies the labels for each section of the pie chart. The main argument sets the title of the chart.