UNIT_II SOLVED QB_R

2-mark questions

1. What are Table format Files? List its features.

able format files are structured data files where data is organized into rows and columns. They are often used for storing and exchanging data. Features of table format files include:

- Tabular structure with rows and columns.
- Data organized in a structured format.
- Common file formats include CSV, TSV, Excel, and more.

2. How can you read a CSV file in R? Give Example

You can read a CSV file in R using the **read.csv()** function. Here's an example:
data <- read.csv("data.csv")

3. What is the purpose of write.table command? Give an example.

The **write.table()** command in R is used to write data frames or matrices to external files, such as text or CSV files. Here's an example:
data <- data.frame(Name = c("Alice", "Bob", "Charlie"), Age = c(25, 30, 22))
write.table(data, "data.txt", sep = "\t", row.names = FALSE)

4**. How to read Web based files? Give example**

You can read web-based files in R using functions like **read.table()**, **read.csv()**, or by downloading the file with libraries like **download.file()** and then reading it. Here's an example using **read.csv()**:
url <- "https://example.com/data.csv"
data <- read.csv(url)

5. How to save plots and graphics directly to file? Give example

You can save plots and graphics to a file in R using the **pdf()**, **png()**, **jpeg()**, or other graphical device functions. Here's an example using **pdf()**:

```
pdf("plot.pdf")
plot(1:10, 1:10)
dev.off()  # Save the plot to plot.pdf and close the device
```

6. Differentiate dput() and dget() functions in R

- **dput()**: This function is used to serialize an R object into a character representation. It is often used to store and share R objects in a human-readable format.
- **dget()**: This function is used to deserialize an R object from its character representation created with **dput()**. It reconstructs the R object from the character string.

7. Differentiate global and local environment in R

- Global Environment: It is the main workspace in R where you interact with objects. It's the top-level environment where you create and modify variables, functions, and data.
- Local Environment: These are temporary environments created when functions are executed. They contain local variables specific to that function's scope and do not affect the global environment.

8. What the "search path" in R refers to.? How can you view the current search path in R?

The "search path" in R is the order in which R looks for objects, variables, and functions. You can view the current search path using the **search()** function:

```
search()
```

9. What is purpose of switch () function in R? Give example

The **switch()** function is used to select one of several alternatives based on a numeric or character index. It's often used for conditional branching. Here's an example:

```
choice <- 2
result <- switch(choice, "Option 1", "Option 2", "Option 3")
```

10.What is use of apply function in R? Give example

The **apply()** function in R is used to apply a function to the rows or columns of a matrix or data frame. It is a convenient way to perform operations on rows or columns. Here's an example:

```
mat <- matrix(1:6, nrow = 2)
row_sums <- apply(mat, 1, sum)  # Calculate row sums
```

11.Differentiate break and next statements in R

- **break**: The **break** statement is used to exit a loop prematurely. When encountered, it terminates the current loop, and control passes to the code following the loop.
- **next**: The **next** statement is used within a loop, typically a **for** or **while** loop. When encountered, it skips the current iteration of the loop and continues with the next iteration

12.What is repeat statement in R? Give an example.

The repeat statement is used to create an infinite loop, which continues indefinitely until it is explicitly terminated using break. It is not commonly used but can be useful in specific situations where you want to continuously run code until a certain condition is met. Here's an example:

```
count <- 1
repeat {
  if (count > 5) {
    break  # Terminate the loop when count is greater than 5
  }
  cat("Count:", count, "\n")
```

```
  count <- count + 1
}
```

13.What do you mean by lazy evaluation? Give an example.

Lazy evaluation means that an expression is not evaluated or computed until its value is actually needed. In R, this allows for efficient memory usage and can improve performance in some cases. Here's an example:

```
x <- 1:10
y <- x * 2  # At this point, the multiplication is not performed, and y is a promise.
print(y)    # When you print or use y, the multiplication is evaluated.
```

14.How to check for Missing Arguments of function in R? Give example

You can check for missing arguments in a function using the **missing()** function. Here's an example:

```
my_function <- function(x, y) {
  if (missing(x) || missing(y)) {
    stop("Both x and y are required arguments.")
  }
  # Rest of the function's code
}

my_function(1)  # This will throw an error because y is missing.
```

15.Write an R code snippet that demonstrates the use of a tryCatch block to handle an exception.

A **tryCatch** block allows you to handle exceptions or errors in R. Here's an example:

```
result <- tryCatch({
  x <- 10 / 0  # This will throw a division by zero error
  x
}, error = function(e) {
  cat("An error occurred:", conditionMessage(e), "\n")
  0  # Return a default value
```

```
})
```

16.How can you measure the execution time of a specific piece of code in R?

you can measure the execution time of a code block in R using the **system.time()** function. Wrap the code you want to time with this function.

17.Give an example of timing a code block using the system.time function.

```
start_time <- system.time({
  for (i in 1:1000000) {
    sqrt(i)
  }
})
end_time <- system.time()
elapsed_time <- end_time - start_time
```

18.How do you unmount packages in R? Give an example.

You can detach or "unmount" packages in R using the **detach()** function. Here's an example:

```
detach(package:ggplot2, unload = TRUE)  # Unmount the ggplot2 package and remove it from the search path
```

19.How the :: operator help to prevent masking when calling functions from specific packages?

The **::** operator allows you to specify which package's function you want to use, preventing function masking or conflicts. It ensures that the function is called from the specified package. For example:

```
base::mean(x)  # Calls the mean function from the base package
stats::mean(x)  # Calls the mean function from the stats package
```

20.Why is data visualization important in data analysis with R?

Data visualization is important in data analysis with R for several reasons:

- It allows you to understand the data's patterns, trends, and outliers.

- Visualizations make complex data more understandable.
- They help in communicating findings and results effectively.
- Visualizations assist in making data-driven decisions.

21.Differentiate boxplot and scatter plot

- **Boxplot**: A boxplot is a graphical representation of the distribution of data, showing the median, quartiles, and potential outliers. It provides a summary of the data's central tendency and spread. Boxplots are useful for comparing multiple datasets and identifying outliers.
- **Scatter Plot**: A scatter plot displays individual data points as dots on a two-dimensional graph. It is used to visualize the relationship between two variables. Scatter plots are excellent for detecting patterns, trends, and relationships in data points.

4 or 6 marks questions

1. How do you read external data files into R? Explain any three types of files with necessary commands to read their characters into R, with example.

you can read external data files into R using functions like **read.table()**, **read.csv()**, and **readLines()**. Here are examples for three file types:

```
data <- read.csv("data.csv")
```

```
text <- readLines("text.txt")
```

```
library(readxl)
```

```
data <- read_excel("data.xlsx")
```

2. What do you mean by argument matching to function in R programming? Explain any three of them.

Argument matching in R refers to how function arguments are matched with their corresponding parameter names in the function definition. Three types of argument matching are:

- Exact Matching: Arguments are matched in the order they are defined in the function. This is the default behavior.
- Partial Matching: Arguments are matched partially based on unique prefixes of parameter names.

- Positional Matching: Arguments are matched by position, ignoring parameter names.

3. Explain if …. else and ifelse statements with syntax and example.

if...else statement:

Syntax:

```
if (condition) {
  # Code to execute if the condition is TRUE
} else {
  # Code to execute if the condition is FALSE
}
```

Example:

```
x <- 5
if (x > 0) {
  result <- "Positive"
} else {
  result <- "Non-positive"
}
```

ifelse() function:

Syntax:

```
ifelse(test, yes, no)
```

Example:

```
x <- c(2, -3, 5, -1)
result <- ifelse(x > 0, "Positive", "Non-positive")
```

4. What do you mean by nesting and stacking in R? Explain with an example.

Nesting: Nesting in R refers to placing one data structure or function inside another. For example, placing a vector inside a list**.**

Example:

```
nested_list <- list(names = c("Alice", "Bob"), ages = c(25, 30))
```

Stacking: Stacking in R typically refers to combining multiple data frames or matrices into a single data frame or matrix. This can be achieved using functions like rbind() for rows and cbind() for columns.

Example:

```
df1 <- data.frame(Name = c("Alice", "Bob"), Age = c(25, 30))

df2 <- data.frame(Name = c("Charlie", "David"), Age = c(22, 28))

stacked_df <- rbind(df1, df2)
```

5. Explain for loop with its varieties in R with syntax and example.

A for loop in R is used to iterate over a sequence (e.g., vector, list) and execute a block of code for each element. Varieties of for loops include:

- Basic for loop:

Syntax:

```
for (variable in sequence) {
  # Code to execute for each element of the sequence
}
```

Example:

```
for (i in 1:5) {
  print(i)
}
```

for loop with seq_along():

Syntax:

```
for (i in seq_along(sequence)) {
  # Code to execute for each element of the sequence
}
```

Example:

```
names <- c("Alice", "Bob", "Charlie")
for (i in seq_along(names)) {
  print(names[i])
}
```

for loop with seq_len():

Syntax:

```
for (i in seq_len(n)) {
  # Code to execute for each element from 1 to n
}
```

Example:

```
n <- 3
for (i in seq_len(n)) {
  print(i)
}
```

6. Explain while loop in R with syntax and example.

A while loop in R is used to repeatedly execute a block of code as long as a specified condition is TRUE.

Syntax:

```
while (condition) {
  # Code to execute while the condition is TRUE
}
```

Example:

```
n <- 5
while (n > 0) {
  cat("Countdown: ", n, "\n")
  n <- n - 1
}
```

7. What is apply function? Explain variety of apply functions with an example.

The apply function in R is used to apply a function to the rows or columns of a matrix or data frame. It is a convenient way to perform operations on rows or columns of data. There are several variations of the apply function, including apply(), lapply(), and sapply(). Here's an example using apply() to calculate row sums:

Syntax:

- apply(X, MARGIN, FUN, ...): X is the data, MARGIN specifies rows (1) or columns (2), and FUN is the function to apply.

Example:

```
mat <- matrix(1:6, nrow = 2)
row_sums <- apply(mat, 1, sum)  # Calculate row sums
```

8. How do you define and call, user defined functions in R? Explain with an example.

You can define a user-defined function in R using the function() keyword. Here's an example:

```
# Define a function
my_function <- function(x, y) {
  result <- x + y
  return(result)
```

```
}


# Call the function
sum_result <- my_function(3, 4)
```

9. How do you set default arguments to a user defined function? Explain with an example.

You can set default arguments in a user-defined function by assigning default values within the function definition. Here's an example:

```
# Define a function with default arguments
greet <- function(name = "Guest") {
  cat("Hello, ", name, "!\n")
}


# Call the function with and without specifying the argument
greet("Alice")  # Outputs: Hello, Alice!
greet()         # Outputs: Hello, Guest!
```

10.Explain three kinds of specialized user defined functions in R, with example.

Three kinds of specialized user-defined functions in R include:

- Recursive Functions: These functions call themselves to solve a problem by breaking it down into smaller sub-problems. Example: Factorial function.
- Vectorized Functions: Functions that operate on entire vectors or data frames without needing explicit loops. Example: sqrt().
- Closure Functions: Functions that return other functions. Example: A function that generates a linear equation based on coefficients.

11.What is exception handling? How do you catch errors with try Statements? Explain with example.

Exception handling is a programming technique to gracefully handle errors or exceptions that may occur during the execution of a program. In R, you can catch errors using try() statements.

Example:

```
result <- try(log("abc"), silent = TRUE)
if (inherits(result, "try-error")) {
  cat("An error occurred: ", conditionMessage(result), "\n")
} else {
  cat("Result: ", result, "\n")
}
```

12.What is “masking" in R? Explain two most common masking situations in R, with example.

Masking in R occurs when a function or object with the same name as an existing one is loaded or defined, causing conflicts. Two common masking situations are:

- **Function Masking:** When a new function with the same name as an existing function is defined, it masks the original function.

Example:

```
# Define a custom mean function
mean <- function(x) {
  sum(x) / length(x)
}

# Now, the custom mean function masks the base R mean function
data <- c(1, 2, 3)
```

result <- mean(data) # Calls the custom mean function

- **Variable Masking**: When a variable with the same name as a function is defined, it masks the function.

Example:

Define a variable named "mean"

mean <- 42

Attempting to use the "mean" function

data <- c(1, 2, 3)

result <- mean(data) # Throws an error due to variable masking

13.How do you draw barplot and pie chart in R? Explain with example.

Barplots: Barplots are created using the barplot() function in R. Here's an example:

data <- data.frame(Category = c("A", "B", "C"), Value = c(10, 20, 15))

barplot(data$Value, names.arg = data$Category, main = "Barplot Example")

Pie Charts: Pie charts are created using the **pie()** function. Here's an example:

values <- c(30, 50, 20)

labels <- c("Category 1", "Category 2", "Category 3")

pie(values, labels = labels, main = "Pie Chart Example")

14.Explain histograms in R with an example

Histograms are created using the hist() function in R. They are used to visualize the distribution of numeric data. Here's an example:

data <- c(23, 28, 30, 32, 35, 35, 37, 39, 40, 45, 47, 49, 50)

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

15.Explain boxplot in R with an example.

Boxplots are created using the boxplot() function in R. They display the distribution of a dataset by showing the median, quartiles, and potential outliers. Here's an example:

```
data <- c(23, 28, 30, 32, 35, 35, 37, 39, 40, 45, 47, 49, 50)

boxplot(data, main = "Boxplot Example")
```

16.What is scatterplot? Explain single plot and matrix of plots, with an example.

A scatterplot is used to visualize the relationship between two numeric variables. It displays data points as dots on a two-dimensional graph. Here are two types of scatterplots:

- **Single Scatter Plot:** A single scatter plot visualizes the relationship between two variables.

Example:

```
x <- c(1, 2, 3, 4, 5)

y <- c(3, 6, 4, 9, 7)

plot(x, y, main = "Scatter Plot Example", xlab = "X-axis", ylab = "Y-axis")
```

Matrix of Scatter Plots: A matrix of scatter plots displays the relationships between multiple pairs of variables. It is useful for visualizing correlations in a dataset.

Example:

```
data <- data.frame(
  x1 = rnorm(100),
  x2 = rnorm(100),
  x3 = rnorm(100)
)

pairs(data, main = "Matrix of Scatter Plots")
```