Challenge-3

Tang Ching Xian

27/08/2023

I. Questions

Question 1: Emoji Expressions

Imagine you're analyzing social media posts for sentiment analysis. If you were to create a variable named "postSentiment" to store the sentiment of a post using emojis (😄 for positive, 😐 for neutral, 😢 for negative), what data type would you assign to this variable? Why? (*narrative type question, no code required*)

Solution: Since the emojis represent positive, neutral and negative, they represents catergories with order, thus they are ordinal categoric data which are normally character. Since we are using "postSentiment", it would likely to be assigned the data type of "string".

Question 2: Hashtag Havoc

In a study on trending hashtags, you want to store the list of hashtags associated with a post. What data type would you choose for the variable "postHashtags"? How might this data type help you analyze and categorize the hashtags later? (*narrative type question, no code required*)

Solution: postHashtags would likely to be datatype "list" or character string. Element in a list can be named and list are useful for storing hashtags since they can vary in number and their order of appearance may be important. Therefore, with list, one can categories the data based on criteria and iterate through the list, tally up the occurrence of each hashtag and identify trends and patterns.

Question 3: Time Traveler's Log

You're examining the timing of user interactions on a website. Would you use a numeric or non-numeric data type to represent the timestamp of each interaction? Explain your choice (*narrative type question, no code required*)

Solution: I would use non-numeric data type since "-",month="april",2.2.2023 all these would be classified as string and continuous double therefore non-numeric.

Question 4: Event Elegance

You're managing an event database that includes the date and time of each session. What data type(s) would you use to represent the session date and time? (*narrative type question, no code required*)

Solution: I would use a continuous numeric datatype because time and date are numerical values, it is continuous and has order of progression of time. Using numeric datatype, one will be able to store and manipulate time-related data easily.

Question 5: Nominee Nominations

You're analyzing nominations for an online award. Each participant can nominate multiple candidates. What data type would be suitable for storing the list of nominated candidates for each participant? (*narrative type question, no code required*)

Solution: Since each participants can nominate multiple candidates, using list data type can elicit names in the list, retrieve the element specifically named and truncated list.

Question 6: Communication Channels

In a survey about preferred communication channels, respondents choose from options like “email,” “phone,” or “social media.” What data type would you assign to the variable “preferredChannel”? (*narrative type question, no code required*)

Solution: I woudl assign categoric data type for “preferredChannel” since categorical dat type is used to represent data that has distinct categories without any inherent order. Since “email”,“phone”, “social media” are discrete, “preferredChannel” variable is suitable because it accurately reflects the data without implying any sort of numeric relationship.

Question 7: Colorful Commentary

In a design feedback survey, participants are asked to describe their feelings about a website using color names (e.g., “warm red,” “cool blue”). What data type would you choose for the variable “feedbackColor”? (*narrative type question, no code required*)

Solution: Since the color names are “red” and “blue”, they are Categoric variables and they are nominal as they do not have natural ordering. Therefore their type is character or logical. But in this case, it should be string of characters.

Question 8: Variable Exploration

Imagine you’re conducting a study on social media usage. Identify three variables related to this study, and specify their data types in R. Classify each variable as either numeric or non-numeric.

Solution:
Variable 1: Number of followers Data type: Numeric (integer; whole number)

Variable 2: Time spent on social media Data type: Numeric (Double; decimal values)

Variable 3: Favourite social media platform Data type: Catgoric character (String; non-numeric)

Question 9: Vector Variety

Create a numeric vector named “ages” containing the ages of five people: 25, 30, 22, 28, and 33. Print the vector.

Solution:

```
ages<- c(25L,30L,22L,28L,33L)
print(ages)
```

```
## [1] 25 30 22 28 33
```

Question 10: List Logic

Construct a list named “student_info” that contains the following elements:

- A character vector of student names: “Alice,” “Bob,” “Catherine”
- A numeric vector of their respective scores: 85, 92, 78
- A logical vector indicating if they passed the exam: TRUE, TRUE, FALSE

Print the list.

Solution:

```
student_info<-list (name=c("Alice","Bob","Catherine"), scores=c(85,92,78),passed_exam=c(TR
UE,TRUE,FALSE))
print(student_info)
```

```
## $name
## [1] "Alice"     "Bob"       "Catherine"
##
## $scores
## [1] 85 92 78
##
## $passed_exam
## [1]  TRUE  TRUE FALSE
```

Question 11: Type Tracking

You have a vector “data” containing the values 10, 15.5, “20”, and TRUE. Determine the data types of each element using the typeof() function.

Solution:

```
# Create a vector
x <- c(10,15.5,"20",TRUE)
# Check the type of x
typeof(x[1])
```

```
## [1] "character"
```

```
typeof(x[2])
```

```
## [1] "character"
```

```
typeof(x[3])
```

```
## [1] "character"
```

```
typeof(x[4])
```

```
## [1] "character"
```

Question 12: Coercion Chronicles

You have a numeric vector “prices” with values 20.5, 15, and “25”. Use explicit coercion to convert the last element to a numeric data type. Print the updated vector.

Solution:

```
prices <- c(20.5, 15, "25")

# Convert the last element to a numeric data type
prices[3] <- as.numeric(prices[3])
```

Question 13: Implicit Intuition

Combine the numeric vector c(5, 10, 15) with the character vector c("apple", "banana", "cherry"). What happens to the data types of the combined vector? Explain the concept of implicit coercion.

Solution:

```
numeric_vector <- c(5, 10, 15)
character_vector <- c("apple", "banana", "cherry")
combined_vector <- c(numeric_vector, character_vector)

print(combined_vector)
```

```
## [1] "5"      "10"     "15"     "apple"  "banana" "cherry"
```

```
print(typeof(combined_vector))
```

```
## [1] "character"
```

The data types of the combined vector will be implicitly coerced to a common data type. In this case, the resulting vector will be a characater data type due to the presence of character vector, hence R convert the numeric data type to character data type based on its content.

Question 14: Coercion Challenges

You have a vector "numbers" with values 7, 12.5, and "15.7". Calculate the sum of these numbers. Will R automatically handle the data type conversion? If not, how would you handle it?

Solution: Mixing different data types (numeric and character) in arithmetic operations, R will not automatically handle the data type conversion. Since R will not be able to automatically handle the data type conversion, I need to use explicit coercion to convert the character into numeric using as.numeric.

```
 numbers <- c(7, 12.5, "15.7")

# Convert character elements to numeric
numbers <- as.numeric(numbers)

# Calculate the sum
sum_result <- sum(numbers)

#simplify
#sum(as.numeric(numbers))

print(sum_result)
```

```
## [1] 35.2
```

Question 15: Coercion Consequences

Suppose you want to calculate the average of a vector "grades" with values 85, 90.5, and "75.2". If you directly calculate the mean using the mean() function, what result do you expect? How might you ensure accurate calculation?

Solution: If I directly calculate the mean of the vector "grades" with values 85, 90.5, and "75.2" using the mean() function without any data type conversion,can lead to incorrect results such as 85. As shown here, mean(85,90.5,“75.2”) [1] 85 OR grades <- c(85, 90.5, “75.2”) mean(grades) [1] NA Mean() function requires the input vector to contain only numeric values.

```
grades <- c(85, 90.5, "75.2")

# Convert character element to numeric
grades <- as.numeric(grades)

# Calculate the mean
mean_result <- mean(grades)

# Simplify
#mean_result <- mean(as.numeric(grades))
print(mean_result)
```

```
## [1] 83.56667
```

Question 16: Data Diversity in Lists

Create a list named “mixed_data” with the following components:

- A numeric vector: 10, 20, 30
- A character vector: “red”, “green”, “blue”
- A logical vector: TRUE, FALSE, TRUE

Calculate the mean of the numeric vector within the list.

Solution:

```
# Creating the list with different components
mixed_data <- list(numeric_vector = c(10, 20, 30),character_vector = c("red", "green", "bl
ue"),logical_vector = c(TRUE, FALSE, TRUE))

# Calculate the mean of the numeric vector within the list
mean_numeric <- mean(mixed_data$numeric_vector)
#Another way to calculate the mean of the numeric vector within the list
#mean_numeric <- mean(mixed_data[["numeric_vector"]])
print(mean_numeric)
```

```
## [1] 20
```

Question 17: List Logic Follow-up

Using the “student_info” list from Question 10, extract and print the score of the student named “Bob.”

Solution:

```
# Assuming you have the "student_info" list defined from Question 10
student_info<-list(names = c("Alice", "Bob", "Catherine"),scores = c(85, 92, 78),passed_ex
am = c(TRUE, TRUE, FALSE))

# Extract and print the score of the student named "Bob"
bob_score<-student_info$scores[student_info$names == "Bob"]

print(bob_score)
```

```
## [1] 92
```

Question 18: Dynamic Access

Create a numeric vector values with random values. Write R code to dynamically access and print the last element of the vector, regardless of its length.

Solution:

```
# Create a numeric vector with random random values
# values <- runif(10, min = 0, max = 100)

# Create a numeric vector with random fixed values
values <- c(10, 20, 30, 40, 50)

# Get the length of the vector
vector_length <- length(values)

# Access and print the last element dynamically
last_element <- values[vector_length]
print(last_element)
```

```
## [1] 50
```

```
##or values[length(values)]
```

Question 19: Multiple Matches

You have a character vector words <- c(“apple”, “banana”, “cherry”, “apple”). Write R code to find and print the indices of all occurrences of the word “apple.”

Solution:

```
words <- c("apple", "banana", "cherry", "apple")

# Find indices of all occurrences of "apple"
apple_indices <- which(words == "apple")

# Print the indices#
print(apple_indices)
```

```
## [1] 1 4
```

Question 20: Conditional Capture

Assume you have a vector ages containing the ages of individuals. Write R code to extract and print the ages of individuals who are older than 30.

Solution:

```r
ages <- c(25, 45, 32, 28, 55, 40)

# Extract ages of individuals older than 30
older_than_30 <- ages[ages > 30]

# Print the ages
print(older_than_30)
```

```
## [1] 45 32 55 40
```

Question 21: Extract Every Nth

Given a numeric vector sequence <- 1:20, write R code to extract and print every third element of the vector.

Solution:

```r
# Numeric vector from 1 to 20
sequence<-1:20

# Extract and print every third element
every_third <- sequence[seq(from = 3, to = length(sequence), by = 3)]

# Print the extracted elements
print(every_third)
```

```
## [1]  3  6  9 12 15 18
```

Question 22: Range Retrieval

Create a numeric vector numbers with values from 1 to 10. Write R code to extract and print the values between the fourth and eighth elements.

Solution:

```r
# Create a numeric vector from 1 to 10
numbers <- 1:10

# Extract and print values between the fourth and eighth elements
between_fourth_and_eighth <- numbers[4:8]

# Print the extracted values
print(between_fourth_and_eighth)
```

```
## [1] 4 5 6 7 8
```

Question 23: Missing Matters

Suppose you have a numeric vector data <- c(10, NA, 15, 20). Write R code to check if the second element of the vector is missing (NA).

Solution:

```
# Numeric vector with NA
data <- c(10, NA, 15, 20)
data[2]
```

```
## [1] NA
```

```
# Check if the second element is NA
is_second_element_missing <- is.na(data[2])

# Print the result
print(is_second_element_missing)
```

```
## [1] TRUE
```

Question 24: Temperature Extremes

Assume you have a numeric vector temperatures with daily temperatures. Create a logical vector hot_days that flags days with temperatures above 90 degrees Fahrenheit. Print the total number of hot days.

Solution:

```
# Numeric vector of daily temperatures
temperatures <- c(88, 92, 87, 95, 89, 92, 91, 86, 93, 97, 88, 99, 84, 88, 92)

# Create a logical vector to flag hot days
hot_days <- temperatures > 90

# Count the total number of hot days
total_hot_days <- sum(hot_days, na.rm = TRUE)

# Print the total number of hot days
print(total_hot_days)
```

```
## [1] 8
```

```
#another method
daily_temp=c(88, 92, 87, 95, 89, 92, 91, 86, 93, 97, 88, 99, 84, 88, 92)
hot_days=vector("logical")
for (temp in daily_temp){#temp is all element
  if (temp>90){
    hot_days<-c(hot_days,TRUE)}
  else{
    hot_days<-c(hot_days,FALSE)}
}
length(hot_days[hot_days==TRUE])
```

```
## [1] 8
```

```
#another method
hot_days= as.logical(daily_temp[daily_temp>90])
length(hot_days)
```

```
## [1] 8
```

Question 25: String Selection

Given a character vector fruits containing fruit names, create a logical vector long_names that identifies fruits with names longer than 6 characters. Print the long fruit names.

Solution:

```
# Character vector of fruit names
fruits <- c("apple", "banana", "strawberry", "kiwi", "blueberry", "orange", "grape")

# Create a logical vector to identify long fruit names
long_names <- nchar(fruits) > 6

# Print the long fruit names
print(fruits[long_names])
```

```
## [1] "strawberry" "blueberry"
```

Question 26: Data Divisibility

Given a numeric vector numbers, create a logical vector divisible_by_5 to indicate numbers that are divisible by 5. Print the numbers that satisfy this condition.

Solution:

```
# Numeric vector of numbers
numbers <- c(10, 25, 14, 15, 30, 8, 20, 16)

# Create a logical vector to indicate numbers divisible by 5
divisible_by_5 <- numbers %% 5 == 0

# Print the numbers that are divisible by 5
print(numbers[divisible_by_5])
```

```
## [1] 10 25 15 30 20
```

Question 27: Bigger or Smaller?

You have two numeric vectors vector1 and vector2. Create a logical vector comparison to indicate whether each element in vector1 is greater than the corresponding element in vector2. Print the comparison results.

Solution:

```r
# Sample numeric vectors
vector1 <- c(10, 25, 14, 15, 30)
vector2 <- c(8, 20, 12, 10, 25)

# Create a logical vector for comparison
comparison <- vector1 > vector2

# Print the comparison results
print(comparison)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```