Challenge-3

Wu Xingyi

2023-08-28

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: *They will likely be stored as a character datatype. This is because in R, emojis are typically represented by multiple characters through their Unicodes.*

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: *In this case, it should be stored as a vector. This is because all the data vector would be of the same type, similar to how all the hashtags in a post.*

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: *A numeric data type should be used to represent the timestamp of each interaction. Numeric data types can hold numerical values that correspond to specific points in time, such as Unix timestamps or milliseconds since a certain reference point.*

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: *Dates are represented by the Date class and times are represented by the POSIXct or the POSIXlt class.*

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: *A suitable data structure would be a list. Lists in R can hold diverse elements, including other lists, making them adaptable to the variable number of nominations each participant can have.*

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: *This should be stored as categorical factors. In this case, the preferred communication channels ("email," "phone," "social media") are categories that respondents choose from, and the order of these categories is not meaningful.*

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: *Character. Character vectors are well-suited for storing text-based responses such as color names.*

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: *hours_per_day -> Numeric, platform_used -> Non-Numeric, followers_count -> 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:

```
#Create the vector
ages <- c('25','30','22','28','33')
#Print the vector
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:

```
# Constructing the list
student_info = list(names = c("Alice", "Bob", "Catherine"),
  scores = c(85, 92, 78),
  passed_exam = c(TRUE, TRUE, FALSE))
# Printing the list
print(student_info)
```

```
## $names
## [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:

```r
# Define the vector
data <- c(10, 15.5, "20", TRUE)

# Determine the data types using typeof()
print(typeof(data[1]))
```

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

```r
print(typeof(data[2]))
```

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

```r
print(typeof(data[3]))
```

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

```r
print(typeof(data[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:

```r
# Numeric vector with mixed data types
prices <- c(20.5, 15, "25")

# Convert the last element to numeric using explicit coercion
prices <- as.numeric(prices)

# Print the updated vector
print(prices)
```

```
## [1] 20.5 15.0 25.0
```

```r
typeof(prices)
```

```
## [1] "double"
```

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:

```r
# Creating the vectors
numeric_vector <- c(5, 10, 15)
character_vector <- c("apple", "banana", "cherry")

#Combining the vectors
combined_vector <- c(numeric_vector, character_vector)

#Print combined vector
print(combined_vector)
```

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

```r
typeof(combined_vector)
```

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

The concept of implicit coercion: *When we input numeric and character data in an object, R converts numeric data to character data by itself.*

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:

```r
# Define the vector
numbers <- c(7, 12.5, "15.7")

# Try to calculate the sum
#print(sum(numbers))
```

R will not handle the data type conversion. The error message reads: Error in sum(numbers) : invalid 'type' (character) of argument.

```r
# Convert character values to numeric
numbers <- as.numeric(numbers)

# Calculate the sum
print(sum(numbers))
```

```
## [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:

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

# Directly calculating the mean
#mean(grades)
```

Expected result: Warning: argument is not numeric or logical: returning NA[1] NA

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

# Convert vector to numeric (ignoring non-numeric values)
grades <- as.numeric(grades)

# Calculate the mean
print(mean(grades))
```

```
## [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:

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

# Calculate the mean of the numeric vector
print(mean(mixed_data$numeric_vector))
```

```
## [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:

```r
# Extracting and printing Bob's score
print(student_info$scores[student_info$names == "Bob"])
```

```
## [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:

```r
# Create a numeric vector with random values
random_values <- c(3.14, 2.71, 1.618, 0.577, 4.669)

print(tail(random_values, n = 1))
```

```
## [1] 4.669
```

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:

```r
# Character vector
words <- c("apple", "banana", "cherry", "apple")

# Find indices of 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
# Sample vector of ages
ages <- c(25, 40, 18, 32, 55, 28, 42)

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

# Print the extracted ages
print(older_than_30)
```

```
## [1] 40 32 55 42
```

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
sequence <- 1:20

# Extract every third element from 1:20
every_third <- sequence[seq(1, length(sequence), by = 3)]

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

```
## [1]  1  4  7 10 13 16 19
```

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 numeric vector
numbers <- 1:10

# Extract values between the fourth and eighth elements
between <- numbers[5:7]

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

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

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:

```r
# Creating numeric data to check
missing_data <- c(10, NA, 15, 20)

# Check for msising data (NA)

is.na(missing_data)
```

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

```r
# Print the answer
print(is.na(missing_data))
```

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

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 temperatures
temperatures <- c(88, 92, 87, 95, 89, 91, 67, 86, 93, 98, 84, 89, 97, 101, 130)

# Create logical vector for hot days
hot_days <- temperatures > 90

# Print the total number of hot days
print(sum(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", "pineapple", "watermelon", "mango", "jackfruit", "
# Create logical vector for long fruit names
long_names <- nchar(fruits) > 6

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

```
## [1] "strawberry" "pineapple"  "watermelon" "jackfruit"
```

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, 15, 8, 25, 26, 70, 80, 84, 13, 30, 12, 7, 20)

# Create logical vector for 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 15 25 70 80 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
# Two numeric vectors
vector1 <- c(10, 15, 8, 25, 30, 24, 212, 2)
vector2 <- c(12, 10, 6, 20, 25, 90, 100, 74)

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

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

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