Challenge-3

Janelle Tan

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:

Data type: Character

Reason:

Emojis are essentially represented as sequences of characters. In R, a character data type is used to store individual characters, strings, or sequences of text. Emojis are a form of textual representation, and they are treated as strings of characters. Therefore, by assigning the "postSentiment" variable the character data type, you can accurately store and represent the sentiment of a post using emojis. The sentiments are also naturally ordered, from positive to negative, hence, the character data type should be used since it is an ordinal variable.

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:

Data type: Character

Reason:

Hashtags are strings of characters.

How will this data type help in analysis and categorisation: Individual Hashtag Access: You can easily access individual hashtags by indexing the vector. For example, you could retrieve the first hashtag using postHashtags[1].

Visualization: You can create visualizations like word clouds or bar plots to showcase the most common hashtags and their frequencies.

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:
Numeric.

Time is a continuous numeric variable. Between any two time points, there are infinitely many other possible time points. Hence, I would consider using the numeric double data type. However, there may be imitations in measurement precision or computational representation. In such cases, I would use the numeric integer data type for time.

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:
Either integer or double data type.

I would use the integer data type for dates, since dates are discrete numeric variables without decimals and are finite.

I would use the double data type for time, since time is a continuous numeric variable. Between any two time points, there are infinitely many other possible time points. However, there may be imitations in measurement precision or computational representation. In such cases, I would use the integer data type for time.

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:

Data type: Character (nominal)

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:
Data type: Character (nominal)

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:
Data type: Character (nominal)

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:

Total usage timing: numeric, double.

User traffic: numeric, integer.

Rating of how much they like the social media platform they're using (very pleasant, pleasant, neutral, unpleasant, very unpleasant): non-numeric, character.

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:

```
# Enter code here
ages <- c(25, 30, 22, 28, 33)

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:

```
# Enter code here

names <- c("Alice", "Bob", "Catherine")
scores <- c(85, 92, 78)
pass_exam <- c(TRUE, TRUE, FALSE)

student_info <- list(names, scores, pass_exam)

print(student_info)
```

```
## [[1]]
## [1] "Alice"     "Bob"       "Catherine"
##
## [[2]]
## [1] 85 92 78
##
## [[3]]
## [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:

```
# Enter code here
data <- c(10, 15.5, "20", TRUE)

typeof(data[1])
```

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

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

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

```
typeof("20")
```

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

```
typeof(TRUE)
```

```
## [1] "logical"
```

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:

```
# Enter code here
prices <- c(20.5, 15, "25")

prices <- as.numeric(prices)

print(prices)
```

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

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:

```
# Enter code here
numbers <- c(5, 10, 15)
fruits <- c("apple", "banana", "cherry")
typeof(numbers)
```

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

```
typeof(fruits)
```

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

```
combined <- c(numbers, fruits)
typeof(combined)
```

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

```
##The data types of the combined vector became of the character data type. Implicit coercion is when R R automatically converts values from one data type to another to perform operations or co mparisons.
```

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:

```
# Enter code here
vector <- c(7, 12.5, "15.7")

## R does not automatically handle the data type conversion.

vector <- as.numeric(vector)
sum(vector)
```

```
## [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:

```
# Enter code here
grades <- c(85, 90.5, "75.2")

##I would expect an error message

grades <- as.numeric(grades)
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:

```
# Enter code here
numbers <- c(10, 20, 30)
colours <- c("red", "green", "blue")
logical <- c(TRUE, FALSE, FALSE)

mixed_data <- list(numbers, colours, logical)
mixed_data
```

```
## [[1]]
## [1] 10 20 30
##
## [[2]]
## [1] "red"   "green" "blue"
##
## [[3]]
## [1]  TRUE FALSE FALSE
```

```
numeric_vector <- mixed_data[[1]]
mean_numeric <- mean(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:

```
# Enter code here
print(student_info)
```

```
## [[1]]
## [1] "Alice"     "Bob"       "Catherine"
##
## [[2]]
## [1] 85 92 78
##
## [[3]]
## [1]  TRUE  TRUE FALSE
```

```
names(student_info) <- c("students", "scores", "passed")

print(student_info)
```

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

```
student_info$scores[2]
```

```
## [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:

```
# Enter code here
vector <- c(1, 2, 3, 4, 5)
last_element <- vector[length(vector)]
print(last_element)
```

```
## [1] 5
```

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
# Enter code here
words <- c("apple", "banana", "cherry", "apple")
apple_indices <- which(words == "apple")
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
# Enter code here
ages <- c(10, 20, 30, 40, 50)
vector <- which(ages > 30)
ages_greater_than_30 <- ages[vector]
print(ages_greater_than_30)
```

```
## [1] 40 50
```

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
# Enter code here
sequence <- 1:20
every_third <- sequence[seq(3, length(sequence), by = 3)]
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
# Enter code here
vector <- c(1:10)
print(vector[4:8])
```

```
## [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:

```
# Enter code here
data <- c(10, NA, 15, 20)
missing <- is.na(data[2])
print(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:

```
# Enter code here
daily_temperatures <- c(70, 80, 90, 100, 110)
hot_days <- c(daily_temperatures > 90)
print(sum(hot_days))
```

```
## [1] 2
```

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:

```
# Enter code here

fruit_names <- c("banana", "pineapple", "strawberry", "blackberry")

long_names <- nchar(fruit_names) > 6

print(fruit_names[long_names])
```

```
## [1] "pineapple"  "strawberry" "blackberry"
```

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:

```r
# Enter code here
numbers <- 1:20
divisible_by_5 <- numbers %% 5 == 0
print(numbers[divisible_by_5])
```

```
## [1]  5 10 15 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
# Enter code here
vector1 <- c(1, 7, 3, 4, 65, 45, 30)
vector2 <- c(3, 6, 5, 47, 89, 90, 54)

comparison <- vector1 > vector2

print(comparison)
```

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