Challenge-3

Jaren Ong

2023-08-30

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 (positive, neutral, negative), what data type would you assign to this variable? Why? (*narrative type question, no code required*)

Solution: Character. I would assign a key word to the emoji.

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: Character. Being a nominal categorical variable, the hashtags serve as a tool to gather key words so that data can be collected, categorized and analysed.

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 data type. It is easier to manipulate and perform calculations with numeric data type.

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: Numeric. Using the unix timestamp, date and time can be represented as the number of seconds that have elapsed since a specific reference point in 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: I would use a list to store the list of nominated candidates for each participant, these elements would be characters.

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 would store the options as characters in a factor.

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: I would store the color name options as characters in a factor.

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: Age (numeric), Number of posts (numeric), Preferred platform (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:

```
# Create a numeric vertor "ages"
ages <- c(25,30,22,28,33)

# Print "ages"
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:

```
# Create a list named "student info"
student_info <- list(
  names = c("Alice", "Bob", "Catherine"),
  scores = c(85, 92, 78),
  passed_exam = c(TRUE, TRUE, FALSE)
)

# Print "student info"
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:

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

# Show data types of each element
for (element in data) {
  type <- typeof(element)
  print(paste("Element:",element," | Data Type:", type))
}
```

```
## [1] "Element: 10  | Data Type: character"
## [1] "Element: 15.5  | Data Type: character"
## [1] "Element: 20  | Data Type: character"
## [1] "Element: TRUE  | Data Type: 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:

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

# Convert "prices" to numeric data type
prices <- as.numeric(prices)

# Check the type of "prices"
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:

```
#
numericvector <- c(5,10,15)
charactervector <- c("apple","banana","cherry")

# Combine both vectors
mixed <- c(numericvector, charactervector)

# Print combined vector
mixed
```

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

```r
# Check data types
typeof(mixed)
```

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

```r
# Explanation: R prioritises character over integer
```

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
# Create vector "numbers"
numbers <- c(7,12.5,"15.7")

# Convert "numbers" vector to numeric so that R can sum them
numbers <- as.numeric(numbers)

# Print "numbers" to check output
numbers
```

```
## [1]  7.0 12.5 15.7
```

```r
# Calculate sum of numbers
sum(numbers)
```

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

```r
# R will not automatically handle data type conversion. I had to use explicit coercion to change the da
```

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:

```r
# Create vector "grades"
grades <- c(85,90.5,"75.2")

# Explicit coercion of "grades" into numeric data types
grades <- as.numeric(grades)

# Find mean of "grades"
mean(grades)
```

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

```
# I tried to find mean using the mean() function before explicit coercion and it returned "NA" Hence, I
```

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 "mixed_data" vector
mixed_data <- list(
  numvec = c(10, 20, 30),
  charvec = c("red", "green", "blue"),
  logvec = c(TRUE, FALSE, TRUE)
)

# Print "mixed_data" to check
mixed_data
```

```
## $numvec
## [1] 10 20 30
##
## $charvec
## [1] "red"   "green" "blue"
##
## $logvec
## [1]  TRUE FALSE  TRUE
```

```
# Calculate the mean of the numeric vector within the list
mean_numeric <- mean(mixed_data$numvec)

# Print the mean
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:

```
# Find the index of "Bob" in the "names" vector
bob_index <- which(student_info$names == "Bob")

# Access and print Bob's score using the index
print(student_info$scores[bob_index])
```

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

```
# I used the which function which helps to find the index of "Bob" in the "names" vector which allowed
```

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 values
values <- runif(10)  # Example: creates a vector of 10 random values between 0 and 1

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

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

```
## [1] 0.469212
```

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:

```
# Create the character vector
words <- c("apple", "banana", "cherry", "apple")

# Find and print the indices of all occurrences of "apple"
indices_apple <- which(words == "apple")
print(indices_apple)
```

```
## [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:

```
# Sample vector of ages
ages <- c(25, 42, 18, 37, 29, 50)

# Extract and print ages of individuals older than 30
older_than_30 <- ages[ages > 30]
print(older_than_30)
```

```
## [1] 42 37 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:

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

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

```
# Numeric vector from 1 to 10
numbers <- 1:10

# Extract and print values between the fourth and eighth elements
between_4_and_8 <- numbers[4:8]
print(between_4_and_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:

```
# Numeric vector with missing value (NA)
data <- c(10, NA, 15, 20)

# Check if the second element is missing (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:

```r
# Sample numeric vector of temperatures
temperatures <- c(85, 92, 88, 91, 89, 95, 86, 93, 90, 88, 94)

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

# Calculate and print the total number of hot days
total_hot_days <- sum(hot_days)
print(total_hot_days)
```

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

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:

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

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

# Extract and print the long fruit names
long_fruit_names <- fruits[long_names]
print(long_fruit_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:

```r
# Create vector "numbers"
numbers <- c(3,5,9,15,39,55,59)

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

# Extract and print the numbers divisible by 5
numbers_divisible_by_5 <- numbers[divisible_by_5]
print(numbers_divisible_by_5)
```

```
## [1]  5 15 55
```

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(12,15,19,20)
vector2 <- c(10,16,18,22)

# Check if each element in vector1 is greater than the corresponding element in vector2
comparevecs <- vector1 > vector2

# Print answer
comparevecs
```

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