Challenge-3

Ho Wei Ni

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

Solution: This would be an ordinal categorical variable so I would assign data type character as strings can be used to represent the category.

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, as I can search for common terms for each topic in the string to categorize the hashtags. For example, to categorise design-related hashtags, I can search for common terms like "design", "uiux" etc. to analyse the data easily.

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: Non-numeric data type as calculations does not to be performed on the time stamp of each interaction in this analysis.

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: Date will be represented by character strings while time will be represented by double as it is a continuous numeric variable.

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: Character as the candidates' names are strings of texts.

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: Character as the options given are nominal categorical variables.

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: Integer type, as we can assign each color name to a value ranking.

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: Screen-time: numeric, Application: non-numeric, Notifications pick-up: 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(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:

```
student_info <- list(student_names = c("Alice","Bob","Catherine"),
                     scores = c(85,92,78),
                     pass = c(TRUE,TRUE,FALSE) )
print(student_info)
```

```
## $student_names
## [1] "Alice"     "Bob"       "Catherine"
##
## $scores
## [1] 85 92 78
##
## $pass
## [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
data <- c(10, 15.5,"20", TRUE)
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
prices <- c(20.5,15,"25")
prices_updated <- as.numeric(prices)
print(prices_updated)
```

```
## [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: The data type of the combined vector follows that of the last element in the vector, which means that all previous elements are converted to match it.

```r
x <- c(5, 10, 15)
x <- c(x, "apple", "banana", "cherry")
print(typeof(x))
```

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

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: Since R will not automatically handle the data type conversion, we have to perform explicit coercion of the vector from character to integer in order to calculate the sum of these numbers.

```r
numbers <- c(7,12.5,"15.7")
sum(as.numeric(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: Due to the mixed data types, "grades" is not a vector but instead is a list. To directly calculate the mean, we should ensure that all elements are numeric. This indicates that we should perform explicit coercion from character to numeric.

```r
grades <- list(85, 90.5, "75.2")
mean(as.numeric(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:

```r
mixed_data <- list(c(10,20,30),c("red","green","blue"),c(TRUE,FALSE,TRUE))
names(mixed_data) <- c("Numeric","Character","Logical")
mean(mixed_data[["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:

```r
bob_score <- student_info$scores[student_info$student_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:

```r
x <- c(1,34,56,6,542,1,489,567,2,4)
print(tail(x,1))
```

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

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
fruits <- c("apple", "banana", "cherry", "apple")
which(fruits == "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:

```r
indivage <- c(12,34,5,2,54,45)
print(indivage[indivage>30])
```

```
## [1] 34 54 45
```

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
seq21 <- 1:20
third <- seq(1, length(seq21), 3)
print(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
vec22 <- 1:10
btwn4n8 <- vec22[5]:vec22[7]
print(btwn4n8)
```

```
## [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
vec23 <- c(10, NA, 15, 20)
is.na(vec23)
```

```
## [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:

```r
daily_temp <- c(65,56,30,102,34,90,99,95,12)
hot_days <- daily_temp > 90
print(sum(hot_days))
```

```
## [1] 3
```

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
fruits25 <- c("apple","banana","dragonfruit","mango","peach","watermelon")
long_names <- nchar(fruits25)>6
print(fruits25[long_names])
```

```
## [1] "dragonfruit" "watermelon"
```

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
vec26 <- 1:92
divisible_by_5 <- vec26 %% 5 == 0
print(vec26[divisible_by_5])
```

```
##  [1]  5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90
```

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
vector1 <- c(5,10,15,20,25,20)
vector2 <- c(-10,0,10,20,30,40)
comparison_result <- vector1 > vector2
print(comparison_result)
```

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