Name: Xian Hui B. Cheng
Section: BSIT 3-5

Activity 3: Working with Vectors and Data Frames

Instructions

Perform the tasks below using R programming. Write the corresponding R code for each question and explain the output briefly. Ensure your code is properly formatted and test it in RStudio or an equivalent IDE.

1. Create a numeric vector x with values from 1 to 50. Extract all even numbers and calculate their sum using a combination of indexing and functions.

```
> x <- 1:50
> even_num <- x[x %% 2 == 0]
> sum <- sum(even_num)
> print(sum)
[1] 650
>
```

Explanation:

In order to get the sum of all even numbers from 1 to 50, we used:

Step 1: **colon operator** (:) - this is used to assign 1 to 50 to x because it's often used to create a sequence of numbers.

Step 2:

To assign all the even numbers from 1 to 50, we used the variable x which contained the sequence of numbers from 1 to 50. Then we used **logical indexing**.

x %% 2 == 0 is used to get all the even numbers and creates a **logical vector**.

Then x[ ] uses the logical vector to extract all the numbers of x where the condition was TRUE and assigns it to even_num.

Step 3: using the function sum() to calculate the sum of even_num without the use of loop.

Bonus: Another way

```
> even_num <- seq(0,50,2)
> sum <- sum(even_num)
> print(sum)
[1] 650
```

2. Construct a data frame employees with columns:
EmployeeID: 101 to 105
Name: "Alice", "Bob", "Charlie", "Diana", "Eve"
Department: "HR", "IT", "Finance", "Marketing", "IT"
Salary: 50000, 60000, 70000, 55000, 65000
Write code to:
Calculate and add a new column Tax as 10% of Salary.
Filter out employees in the IT department.

```
employeeDf <- data.frame(
  employeeId = 101:105,
  name = c("Alice", "Bob", "Charlie", "Diana", "Eve"),
  department = c("HR", "IT", "Finance", "Marketing", "IT"),
  salary = c(50000, 60000, 70000, 55000, 65000)
)

employeeDf$tax <- employeeDf$salary * .10
print(employeeDf)
nonITDept <-employeeDf[employeeDf$department != "IT", ]
print(nonITDept)
```

Output:

```
> print(employeeDf)
  employeeId    name department salary  tax
1        101   Alice         HR  50000 5000
2        102     Bob         IT  60000 6000
3        103 Charlie    Finance  70000 7000
4        104   Diana  Marketing  55000 5500
5        105     Eve         IT  65000 6500
> nonITDept <-employeeDf[employeeDf$department != "IT", ]
> print(nonITDept)
  employeeId    name department salary  tax
1        101   Alice         HR  50000 5000
3        103 Charlie    Finance  70000 7000
4        104   Diana  Marketing  55000 5500
```

Explanation:

employeeDf
- This was created through the data.frame, and it got the columns employeeId, name, department, and salary. These columns were initialize using the **c()** / concatenate which creates a sequence of data into a single vector and forms the rows of the data frame.

- We also used an = operator instead of <- inside the function. Although they both have the same use cases, We used = for function arguments since <- is only used for assignment operator.

tax column

- To get the tax column that calculates 10% of the salary, we used **$** -> this operator is used to directly access the columns of the employeeDf. We both used it to create a column for tax, and also access the column of salary.

nonITDept (filtering out IT department)

- All data frames in R uses [rows, columns]. We accessed the row values in R using this. That why we used employeeDf[employeeDf$department != IT,]. This checks every row in the employeeDf and assigns every value that is not equal to “IT” into nonITDept.

- We used the empty comma [,] to separate the rows from the columns. A blank column means that we are including all the original column from employeeDf.

3. Write an R function that takes a column name as input and returns all unique values of that column from a data frame. Demonstrate this using the employees data frame.

Note: For this demonstration, I added another “Bob” name to the data frame.

```
employeeDf <- data.frame(
  employeeId = 101:105,
  name = c("Alice", "Bob", "Charlie", "Diana", "Eve","Bob"),
  department = c("HR", "IT", "Finance", "Marketing", "IT"),
  salary = c(50000, 60000, 70000, 55000, 65000)
)
```

Code:

```
unique_values <- function (df, columnName){
  if (columnName %in% colnames(df)){ #check if column exist
    return(unique(df[[columnName]]))
  }else{
    print("Column name not found")
  }
}

unique_name <- unique_values(employeeDf, "name")
print(unique_name)
```

Output:

```
> unique_name <- unique_values(employeeDf, "name")
> print(unique_name)
[1] "Alice"   "Bob"     "Charlie" "Diana"   "Eve"
```

Explanation:
Unique_values

- This checks if the column name exists in the data frame using columnName %in% colnames (df)
- If the condition evaluates TRUE (the column exists) the it uses the unique() function. This function finds and returns the unique values from the data that was passed in its parameters
- If the condition evaluates FALSE (the column doesn't exist), it will print an error message

Tracing:

employeeDf$name	unique(df[[columnName]])	unique_name (returned)
Alice	TRUE	c("Alice")
Bob	TRUE	c("Alice", "Bob")
Charlie	TRUE	c("Alice", "Bob", "Charlie")
Diana	TRUE	c("Alice", "Bob", "Charlie", "Diana")
Eve	TRUE	c("Alice", "Bob", "Charlie", "Diana", "Eve")
Bob	FALSE	c("Alice", "Bob", "Charlie", "Diana", "Eve")

4. Write an R script to check if any Salary in the employees data frame is above 70,000. If found, print "High Salary Detected"; otherwise, print "All Salaries are Within Range".

```
check_salary <- function(data) {
  if (any(data$salary > 70000)) {
    print("High Salary Detected")
  } else {
    print("All Salaries are Within Range")
  }
}

check_salary(employeeDf)
```

Output:

```
> check_salary(employeeDf)
[1] "All Salaries are Within Range"
```

Modified data frame: (changed to 80,000)

```
salary = c(50000, 60000, 70000, 55000, 65000, 80000)
```

```
> check_salary(employeeDf)
[1] "High Salary Detected"
```

Explanation:

check_salary

- The condition any(data$salary > 70000):
    - Evaluates if any salaries in salary column exceeds 70,000
    - If any() returns TRUE, it means at least one value exceed 70,000.
- If the condition evaluates TRUE, it prints “High Salary Detected”
- If the condition evaluates FALSE, it prints “All Salaries are Within Range”

Example Tracing w/ Modified Data Frame (one salary is 80,000):

```
salary = c(50000, 60000, 70000, 55000, 65000, 80000)
```

Salary	any(data$salary > 70000)	
50,000	FALSE	
60,000	FALSE	
70,000	FALSE	
55,000	FALSE	
65,000	FALSE	
80,000	TRUE	“High Salary Detected”

5. Modify the employees data frame so that one Salary value is set to NA. Write code to calculate the total salary excluding the NA values and explain the output.

```
employeeDf$salary[4] <- NA
calculate_salary <- function(df) {
  totalSalary <- 0

for (i in seq_along(df$salary)) {
    if (!is.na(df$salary[i])) {  # Check if the salary is not NA
      totalSalary <- totalSalary + df$salary[i]
    }
}
  return(totalSalary)
}

total <- calculate_salary(employeeDf)
print(total)
```

Output:

```
> total <- calculate_salary(employeeDf)
> print(total)
[1] 240000
```

Explanation:

- Updating the 4th employee of employeeDf to make it NA. Which makes the 55,000 into NA. This means 50000 + 60000 + 70000 + 60000 = 240000

```
  salary = c(50000, 60000, 70000, 55000, 60000)
)
```

Process:

calculate_salary - this takes any data frame and calculates the total salary while skipping NA values

Loop

- Using for loop to iterate through salary column.
- seq_along(df$salary) is used to generate the right length for the column size of salary (is used as i).
- Each iteration of the loop checks if it's not NA through !is.na(df$salary[i]) based on the current iteration number.

totalSalary

- Each time a condition passes, the salary value is added to the totalSalary and returned once the loop finishes

totalSalary	employeeDf$salary	!is.na(df$salary[i])
0	50000	TRUE

50000	60000	TRUE
110000	70000	TRUE
180000	NA	FALSE
180000	60000	TRUE
240000	end	

Another solution without using loop:

```
employeeDf$salary[4] <- NA
total <- sum(Employee_DF$Salary, na.rm = TRUE)
print(total)
```