# RWorksheet_BIBIT#4a

John Benedict Bibit

2024-10-22

1. The table below shows the data about shoe size and height. Create a data frame.

a. Describe the data.

```
shoes_data <- data.frame(
  Gender = c("F", "F", "F", "F", "M", "F", "F", "F", "M", "F", "M", "F", "M", "M", "M", "M", "F", "F",
  Shoe_Size = c( 6.5, 9.0, 8.5, 8.5, 10.5, 7.0, 9.5, 9.0, 13.0, 7.5, 10.5, 8.5, 12.0, 10.5, 13.0, 11.5,
  Height = c(66.0, 68.0, 64.5, 65.0, 70.0, 64.0, 70.0, 71.0, 72.0, 64.0, 74.5, 67.0, 71.0, 71.0, 77.0,
)
```

b. Create a subset by males and females with their corresponding shoe size and height. What its result? Show the R scripts.

```
# b. Subset by gender
males <- subset(shoes_data, Gender == "M")
females <- subset(shoes_data, Gender == "F")
```

c. Find the mean of shoe size and height of the respondents. Write the R scripts and its result.

```
mean_shoe_size <- mean(shoes_data$Shoe_Size)
mean_height <- mean(shoes_data$Height)
mean_shoe_size
```

```
## [1] 9.410714
```

```
mean_height
```

```
## [1] 68.57143
```

d. Is there a relationship between shoe size and height? Why? Yes, typically there's a positive correlation between shoe size and height

Factors A nominal variable is a categorical variable without an implied order. This means that it is impossible to say that 'one is worth more than the other'. In contrast, ordinal variables do have a natural ordering. Example: Gender <- c("M","F","F","M") factor_Gender <- factor(Gender) factor_Gender

## [1] M F F M

## Levels: F M

2. Construct character vector months to a factor with factor() and assign the result to factor_months_vector. Print out factor_months_vector and assert that R prints out the factor levels below the actual values. Consider data consisting of the names of months: "March","April","January","November","January", "September","October","September","November","August", "January","November","November","February","May","August", "July","December","August","August","September","N
April")

```r
months_vector <- c("March","April","January","November","January",
"September","October","September","November","August",
"January","November","November","February","May","August",
"July","December","August","August","September","November","February","April")

factor_months_vector <- factor(months_vector)
print(factor_months_vector)
```

```
##  [1] March     April     January   November  January   September October  
##  [8] September November  August    January   November  November  February 
## [15] May       August    July      December  August    August    September
## [22] November  February  April    
## 11 Levels: April August December February January July March May ... September
```

3. Then check the summary() of the months_vector and factor_months_vector. | Inter- pret the results of both vectors. Are they both equally useful in this case?

```r
summary(months_vector)
```

```
##    Length     Class      Mode 
##        24 character character 
```

```r
summary(factor_months_vector)
```

```
##     April    August  December  February   January      July     March       May 
##         2         4         1         2         3         1         1         1 
##  November   October September 
##         5         1         3 
```

```r
# The factor summary is more useful as it shows the frequency of each month
```

4. Create a vector and factor for the table below. Note: Apply the factor function with required order of the level. new_order_data <- factor(factor_data,levels = c(“East”,“West”,“North”)) print(new_order_data)

```r
directions <- c("East", "West", "West", "West", "West", "North", "North", "North")
factor_data <- factor(directions)
new_order_data <- factor(factor_data, levels = c("East","West","North"))
print(new_order_data)
```

```
## [1] East  West  West  West  West  North North North
## Levels: East West North
```

5. Enter the data below in Excel with file name = import_march.csv

a. Import the excel file into the Environment Pane using read.table() function. Write the code.

```r
data <- read.table("import_march.csv")
```

b. View the dataset. Write the R scripts and its result.

```r
print(data)
```

```
##                              V1
## 1 students,strat1,strat2,strat3
## 2                   male,8,10,8
## 3                        ,4,8,6
## 4                        ,0,6,4
## 5                           ,,,
## 6                female,14,4,15
## 7                      ,10,2,12
## 8                        ,6,0,9
```

Using Conditional Statements (IF-ELSE) 6. Full Search Exhaustive search is a methodology for finding an answer by exploring all possible cases. When trying to find a desired number in a set of given numbers, the method of finding the corresponding number by checking all elements in the set one by one can be called an exhaustive search. Implement an exhaustive search function that meets the input/output conditions below.

a. Create an R Program that allows the User to randomly select numbers from 1 to 50. Then display the chosen number. If the number is beyond the range of the selected choice, it will have to display a string "The number selected is beyond the range of 1 to 50". If number 20 is inputted by the User, it will have to display "TRUE", otherwise display the input number.

```r
user_number <- 20

if(user_number < 1 || user_number > 50) {
  print("The number selected is beyond the range of 1 to 50")
} else if(user_number == 20) {
  print("TRUE")
} else {
  print(user_number)
}
```

```
## [1] "TRUE"
```

7. Change At ISATU University's traditional cafeteria, snacks can only be purchased with bills. A long-standing rule at the concession stand is that snacks must be purchased with as few coins as possible. There are three types of bills: 50 pesos, 100 pesos, 200 pesos, 500 pesos, 1000 pesos.

a. Write a function that prints the minimum number of bills that must be paid, given the price of the snack. Input: Price of snack (a random number divisible by 50) Output: Minimum number of bills needed to purchase a snack.

```r
calculate_bills <- function(price) {
  bills <- c(1000, 500, 200, 100, 50)
  total_bills <- 0
  remaining <- price

  for(bill in bills) {
    count <- floor(remaining/bill)
    total_bills <- total_bills + count
    remaining <- remaining - (count * bill)
  }
  return(total_bills)
}
```

8. The following is each student's math score for one semester. Based on this, answer the following questions.

Name Grade1 Grade2 Grade3 Grade4 Annie 85 65 85 100 Thea 65 75 90 90 Steve 75 55 80 85 Hanna 95 75 100 90

a. Create a dataframe from the above table. Write the R codes and its output.

```r
grades_df <- data.frame(
  Name = c("Annie", "Thea", "Steve", "Hanna"),
  Grade1 = c(85, 65, 75, 95),
  Grade2 = c(65, 75, 55, 75),
  Grade3 = c(85, 90, 80, 100),
  Grade4 = c(100, 90, 85, 90)
)
```

b. Without using the rowMean function, output the average score of students whose average math score over 90 points during the semester. write R code and its output. Example Output: Annie's average grade this semester is 88.75.

```
for(i in 1:nrow(grades_df)) {
  avg <- sum(grades_df[i,2:5])/4
  if(avg > 90) {
    print(paste(grades_df$Name[i], "'s average grade this semester is", avg))
  }
}
```

c. Without using the mean function, output as follows for the tests in which the average score was less than 80 out of 4 tests. Example output: The nth test was difficult.

```
for(j in 2:5) {
  test_avg <- sum(grades_df[,j])/nrow(grades_df)
  if(test_avg < 80) {
    print(paste("The", j-1, "th test was difficult"))
  }
}
```

```
## [1] "The 2 th test was difficult"
```

d. Without using the max function, output as follows for students whose highest score for a semester exceeds 90 points. Example Output: Annie's highest grade this semester is 95.

```
for(k in 1:nrow(grades_df)) {
  max_grade <- -Inf

  for (j in 2:5) {
    if (grades_df[k, j] > max_grade) {
      max_grade <- grades_df[k, j]
    }
  }

  if(max_grade > 90) {
    print(paste(grades_df$Name[k], "'s highest grade this semester is 95"))
  }
}
```

```
## [1] "Annie 's highest grade this semester is 95"
## [1] "Hanna 's highest grade this semester is 95"
```