RWorksheet_Calzado#4a.Rmd

Michael Angelo S. Calzado

2024-10-16

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

```
Table <- data.frame(
  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, 8
  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, 7
  Gender = c("F", "F", "F", "F", "M", "F", "F", "F", "M", "F", "M", "F", "M", "M", "M", "M", "F", "F", "
)
Table
```

```
##    Shoe_size Height Gender
## 1        6.5   66.0      F
## 2        9.0   68.0      F
## 3        8.5   64.5      F
## 4        8.5   65.0      F
## 5       10.5   70.0      M
## 6        7.0   64.0      F
## 7        9.5   70.0      F
## 8        9.0   71.0      F
## 9       13.0   72.0      M
## 10       7.5   64.0      F
## 11      10.5   74.5      M
## 12       8.5   67.0      F
## 13      12.0   71.0      M
## 14      10.5   71.0      M
## 15      13.0   77.0      M
## 16      11.5   72.0      M
## 17       8.5   59.0      F
## 18       5.0   62.0      F
## 19      10.0   72.0      M
## 20       6.5   66.0      F
## 21       7.5   64.0      F
## 22       8.5   67.0      M
## 23      10.5   73.0      M
## 24       8.5   69.0      F
## 25      10.5   72.0      M
## 26      11.0   70.0      M
## 27       9.0   69.0      M
## 28      13.0   70.0      M
```

a. Describe the data.

The data frame Table contains three columns: Shoe_size, Height, and Gender.

Shoe_size: A numeric vector representing the shoe sizes of individuals. The values range from 5.0 to 13.0.
Height: A numeric vector representing the height of individuals in inches. The values range from 59.0 to 77.0

inches. Gender: A categorical variable indicating the gender of each individual, with values "M" for male and "F" for female.

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

```
maledata <- subset(Table, Gender = "M", select = c(Shoe_size, Height))
```

```
## Warning: In subset.data.frame(Table, Gender = "M", select = c(Shoe_size,
##     Height)) :
##  extra argument 'Gender' will be disregarded
```

```
maledata
```

```
##    Shoe_size Height
## 1        6.5   66.0
## 2        9.0   68.0
## 3        8.5   64.5
## 4        8.5   65.0
## 5       10.5   70.0
## 6        7.0   64.0
## 7        9.5   70.0
## 8        9.0   71.0
## 9       13.0   72.0
## 10       7.5   64.0
## 11      10.5   74.5
## 12       8.5   67.0
## 13      12.0   71.0
## 14      10.5   71.0
## 15      13.0   77.0
## 16      11.5   72.0
## 17       8.5   59.0
## 18       5.0   62.0
## 19      10.0   72.0
## 20       6.5   66.0
## 21       7.5   64.0
## 22       8.5   67.0
## 23      10.5   73.0
## 24       8.5   69.0
## 25      10.5   72.0
## 26      11.0   70.0
## 27       9.0   69.0
## 28      13.0   70.0
```

```
femaledata <- subset(Table, Gender = "F", select = c(Shoe_size, Height))
```

```
## Warning: In subset.data.frame(Table, Gender = "F", select = c(Shoe_size,
##     Height)) :
##  extra argument 'Gender' will be disregarded
```

```
femaledata
```

```
##    Shoe_size Height
## 1        6.5   66.0
## 2        9.0   68.0
## 3        8.5   64.5
## 4        8.5   65.0
## 5       10.5   70.0
```

```
## 6        7.0   64.0
## 7        9.5   70.0
## 8        9.0   71.0
## 9       13.0   72.0
## 10       7.5   64.0
## 11      10.5   74.5
## 12       8.5   67.0
## 13      12.0   71.0
## 14      10.5   71.0
## 15      13.0   77.0
## 16      11.5   72.0
## 17       8.5   59.0
## 18       5.0   62.0
## 19      10.0   72.0
## 20       6.5   66.0
## 21       7.5   64.0
## 22       8.5   67.0
## 23      10.5   73.0
## 24       8.5   69.0
## 25      10.5   72.0
## 26      11.0   70.0
## 27       9.0   69.0
## 28      13.0   70.0
```

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

```
meanmaleshoesize <- mean(maledata$Shoe_size)
meanfemaleshoesize <- mean(femaledata$Shoe_size)
meanmaleheight <- mean(maledata$Height)
meanfemaleheight <- mean(femaledata$Height)
print("Mean female's shoe size")
```

```
## [1] "Mean female's shoe size"
```

```
meanfemaleshoesize
```

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

```
print("Mean male's shoe size")
```

```
## [1] "Mean male's shoe size"
```

```
meanmaleshoesize
```

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

```
print("Mean female's heights")
```

```
## [1] "Mean female's heights"
```

```
meanfemaleheight
```

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

```
print("Mean male's heights")
```

```
## [1] "Mean male's heights"
```

```
meanmaleheight
```

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

d. Is there a relationship between shoe size and height? Why? There appears to be a positive relationship between shoe size and height, as individuals with larger shoe sizes generally have greater heights. This is because larger shoe sizes typically accommodate taller individuals who tend to have longer feet.

Figure 1: Household Data

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

```
months_vector <- c("March", "April", "January", "November", "January", "September", "October", "Septembe
cat("Vector:")
```

```
## Vector:
```

```
months_vector
```

```
##  [1] "March"     "April"     "January"   "November"  "January"   "September"
##  [7] "October"   "September" "November"  "August"    "January"   "November"
## [13] "November"  "February"  "May"       "August"    "July"      "December"
## [19] "August"    "August"    "September" "November"  "February"  "April"
```

```
cat("\n")
```

```
print("Factor:")
```

```
## [1] "Factor:"
```

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

```
cat("\n")
```

```
print("Level:")
```

```
## [1] "Level:"
```

```
levels(factor_months_vector)
```

```
##  [1] "April"     "August"    "December"  "February"  "January"   "July"
##  [7] "March"     "May"       "November"  "October"   "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?

```
summary(months_vector)
```

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

```
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
```

4. Create a vector and factor for the table below. Direction Frequency East 1 West 4 North 3 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)

```
Direction <- c("East", "West", "North")
Frequency <- c( 1, 4, 3)
print("Direction: ")
```

```
## [1] "Direction: "
```

```
Direction
```

```
## [1] "East"  "West"  "North"
```

```
print("Frequency: ")
```

```
## [1] "Frequency: "
```

```
Frequency
```

```
## [1] 1 4 3
```

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

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

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

```
library(readr)
import_march <- read_csv("/cloud/project/import_march.csv")
```

```
## Rows: 6 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Students
## dbl (3): Strategy 1, Strategy 2, Strategy 3
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
import_march
```

```
## # A tibble: 6 x 4
##   Students `Strategy 1` `Strategy 2` `Strategy 3`
##   <chr>           <dbl>        <dbl>        <dbl>
## 1 Male                8           10            8
```

```
## 2 <NA>                4            8            6
## 3 <NA>                0            6            4
## 4 Female             14            4           15
## 5 <NA>               10            2           12
## 6 <NA>                6            0            9
```

Figure 2: Excel Data

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

```
import_march <- read.table("/cloud/project/import_march.csv", header = TRUE, sep = ",")
```

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

```
print(import_march)
```

```
##   Students Strategy.1 Strategy.2 Strategy.3
## 1     Male          8         10          8
## 2                   4          8          6
## 3                   0          6          4
## 4   Female         14          4         15
## 5                  10          2         12
## 6                   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.

```
exhaustivesearch <- function() {
  chosen_number <- as.integer(readline(prompt = "Enter a number between 1 and 50: "))

  # Check if the input is NA
  if (is.na(chosen_number)) {
    print("Invalid input. Please enter a valid number.")
  } else if (chosen_number < 1 || chosen_number > 50) {
    print("The number selected is beyond the range of 1 to 50")
  } else if (chosen_number == 20) {
    print("TRUE")
  } else {
    print(paste("The chosen number is:", chosen_number))
  }
}

exhaustivesearch()
```

```
## Enter a number between 1 and 50:
## [1] "Invalid input. Please enter a valid number."
```

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.

```
min_bills <- function(price) {
  bills <- c(1000, 500, 200, 100, 50)
  count <- 0

  for (bill in bills) {
    while (price >= bill) {
      price <- price - bill
      count <- count + 1
    }
  }

  return(count)
}

snack_price <- as.integer(readline(prompt = "Enter the price of the snack (multiple of 50): "))
```

```
## Enter the price of the snack (multiple of 50):
```

```
if (is.na(snack_price)) {
  cat("Invalid input. Please enter a valid number.\n")
} else if (snack_price %% 50 == 0) {
  min_count <- min_bills(snack_price)
  cat("Minimum number of bills needed:", min_count, "\n")
} else {
  cat("The price must be a multiple of 50.\n")
}
```

```
## Invalid input. Please enter a valid number.
```

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.

```
students <- data.frame(
  Name = c("Annie", "John", "Lisa", "Tom"),
  Grade1 = c(95, 85, 92, 75),
  Grade2 = c(90, 88, 94, 80),
  Grade3 = c(88, 82, 90, 78),
  Grade4 = c(92, 85, 85, 70)
)
students
```

```
##    Name Grade1 Grade2 Grade3 Grade4
## 1 Annie     95     90     88     92
## 2  John     85     88     82     85
## 3  Lisa     92     94     90     85
## 4   Tom     75     80     78     70
```

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.

```
students$Average <- (students$Grade1 + students$Grade2 + students$Grade3 + students$Grade4) / 4
```

```r
for (i in 1:nrow(students)) {
  if (students$Average[i] > 90) {
    cat(students$Name[i], "'s average grade this semester is", round(students$Average[i], 2), "\n")
  }
}
```

```
## Annie 's average grade this semester is 91.25
## Lisa 's average grade this semester is 90.25
```

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.

```r
test_averages <- c()
for (j in 2:5) {
  test_averages[j - 1] <- sum(students[, j]) / nrow(students)
}
for (j in 1:length(test_averages)) {
  if (test_averages[j] < 80) {
    suffix <- switch(as.character(j),
                     "1" = "st",
                     "2" = "nd",
                     "3" = "rd",
                     "4" = "th")
    cat("The", j, suffix, "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.

```r
for (i in 1:nrow(students)) {
  highest_score <- students[i, 2]
  for (j in 3:5) {
    if (students[i, j] > highest_score) {
      highest_score <- students[i, j]
    }
  }
  if (highest_score > 90) {
    cat(students$Name[i], "'s highest grade this semester is", highest_score, "\n")
  }
}
```

```
## Annie 's highest grade this semester is 95
## Lisa 's highest grade this semester is 94
```