RWorksheet_GENER#4a.Rmd

Kent Gener

2023-10-25

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

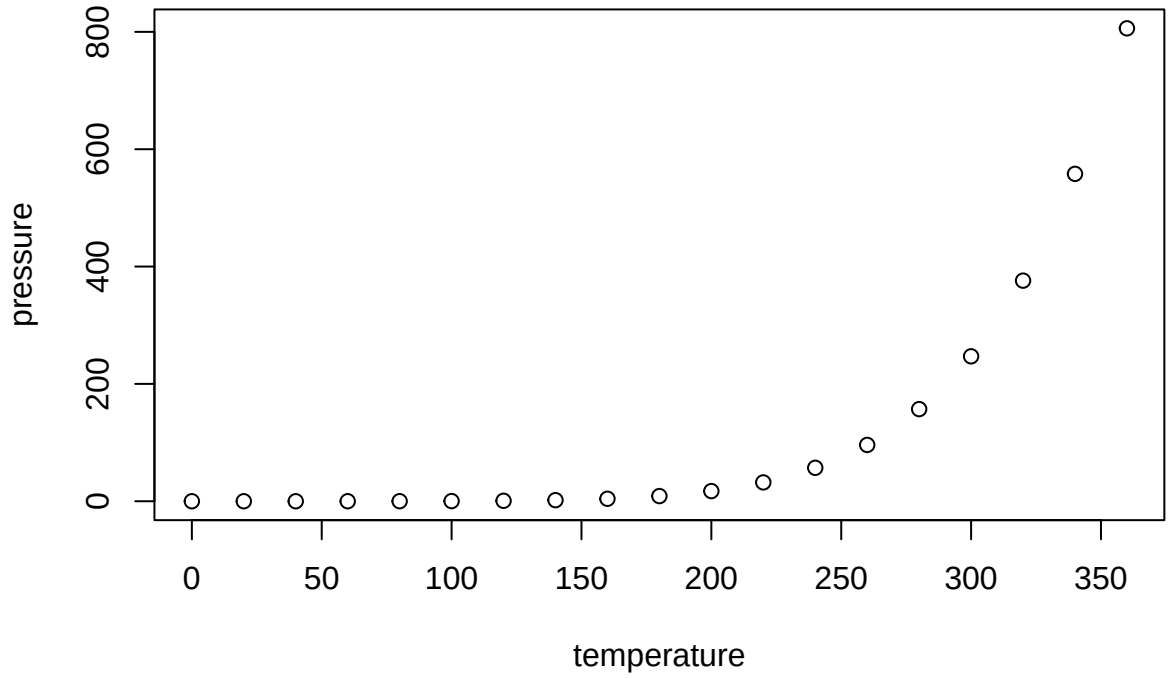
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Worksheet-4a in R

Worksheet for R Programming Instructions: • Use RStudio or the RStudio Cloud accomplish this worksheet. • Create folder for this worksheet#4. Inside the folder, create an .Rmd (R Markdown) for this worksheet and saved it as RWorksheet_lastname#4a.Rmd • Knit to pdf to render a pdf file. • On your own GitHub repository, push the .Rmd file, as well as the pdf worksheet knitted to the repo you have created before. • Do not forget to comment your Git repo on our VLE • Accomplish this worksheet by answering the questions being asked and writing the code manually.

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

a. Describe the data.

```
ShoeSize1 <- c(6.5, 90, 85, 85, 10.5, 70, 95, 90, 13.0, 7.5, 10.5, 8.5, 12.0, 10.5)
Height1 <- 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)
Gender1 <- c("F", "F", "F", "F", "M", "F", "F", "F", "M", "F", "M", "F", "M", "M")

ShoeSize2 <- c(13.0, 11.5, 8.5, 5.0, 10.0, 6.5, 7.5, 8.5, 10.5, 8.5, 10.5, 11.0, 9.0, 13.0)
Height2 <- c(77.0, 72.0, 59.0, 62.0, 72.0, 66.0, 60.0, 67.6, 73.0, 69.0, 72.0, 70.0, 69.0, 70.0)
Gender2 <- c("M", "M", "F", "F", "M", "F", "F", "M", "M", "F", "M", "M", "M", "M")

ShoeSize <- c(ShoeSize1, ShoeSize2)
Height <- c(Height1, Height2)
```

```r
Gender <- c(Gender1, Gender2)

data <- data.frame(ShoeSize, Height, Gender)

summary(data)
```

```
##     ShoeSize         Height         Gender
##  Min.   : 5.00   Min.   :59.00   Length:28
##  1st Qu.: 8.50   1st Qu.:65.75   Class :character
##  Median :10.50   Median :69.50   Mode  :character
##  Mean   :25.96   Mean   :68.45
##  3rd Qu.:13.00   3rd Qu.:71.25
##  Max.   :95.00   Max.   :77.00
```

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

```r
males <- data[data$Gender == "M", c("ShoeSize", "Height")]
females <- data[data$Gender == "F", c("ShoeSize", "Height")]

males
```

```
##    ShoeSize Height
## 5      10.5   70.0
## 9      13.0   72.0
## 11     10.5   74.5
## 13     12.0   71.0
## 14     10.5   71.0
## 15     13.0   77.0
## 16     11.5   72.0
## 19     10.0   72.0
## 22      8.5   67.6
## 23     10.5   73.0
## 25     10.5   72.0
## 26     11.0   70.0
## 27      9.0   69.0
## 28     13.0   70.0
```

```r
females
```

```
##    ShoeSize Height
## 1       6.5   66.0
## 2      90.0   68.0
## 3      85.0   64.5
## 4      85.0   65.0
## 6      70.0   64.0
## 7      95.0   70.0
## 8      90.0   71.0
## 10      7.5   64.0
## 12      8.5   67.0
## 17      8.5   59.0
## 18      5.0   62.0
```

```
## 20      6.5   66.0
## 21      7.5   60.0
## 24      8.5   69.0
```

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

```
mean_shoe_size <- mean(data$ShoeSize)
mean_height <- mean(data$Height)

mean_shoe_size
```

```
## [1] 25.96429
```

```
mean_height
```

```
## [1] 68.45
```

d. Is there a relationship between shoe size and height? Why?

```
correlation <- cor(data$ShoeSize, data$Height)

correlation
```

```
## [1] -0.09758546
```

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","Januar "September","October","September","November","August", "January","November","November","February","May","A

```
months_vector <- c("March", "April", "January", "November", "January", "September", "October", "Septembe

factor_months_vector <- factor(months_vector)

levels(factor_months_vector)
```

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

```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
```

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
```

4. Create a vector and factor for the table below.

```r
direction <- c("East", "West", "North")
frequency <- c(1, 4, 3)

data <- data.frame(Direction = direction, Frequency = frequency)

custom_order <- c("East", "West", "North")

factor_direction <- factor(direction, levels = custom_order)

direction
```

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

```r
factor_direction
```

```
## [1] East  West  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.F

```r
std_data <- read.table("import_march.csv", header = TRUE, sep = ",", stringsAsFactors = FALSE)
```

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

```r
head(std_data)
```

```
##   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.

```r
exhaustive_search <- function(number) {
  if (number < 1 || number > 50) {
    return("The number selected is beyond the range of 1 to 50")
  } else if (number == 20) {
    return(TRUE)
  } else {r
    return(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.

```r
get_min_bills <- function(price) {
  bills <- c(1000, 500, 200, 100, 50)
  bill_counts <- c(0, 0, 0, 0, 0)
  names(bill_counts) <- bills

  for (bill in bills) {
    bill_counts[bill] <- price %/% bill
    price <- price %% bill
  }

  total_bills <- sum(bill_counts)
  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.

```
students_scores <- 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)
)

print(students_scores)
```

```
##    Name Grade1 Grade2 Grade3 Grade4
## 1 Annie     85     65     85    100
## 2  Thea     65     75     90     90
## 3 Steve     75     55     80     85
## 4 Hanna     95     75    100     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.

```
students_scores$Average <- rowSums(students_scores[,2:5]) / 4

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

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.

```
test_averages <- colSums(students_scores[,2:5]) / nrow(students_scores)

for (i in 1:length(test_averages)) {
  if (test_averages[i] < 80) {
    cat("The", names(test_averages)[i], "test was difficult.\n")
  }
}
```

```
## The Grade2 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
students_scores$Highest <- apply(students_scores[,2:5], 1, function(x) sort(x, decreasing = TRUE)[1])

for (i in 1:nrow(students_scores)) {
  if (students_scores$Highest[i] > 90) {
    cat(students_scores$Name[i], "highest grade this semester is", students_scores$Highest[i], "\n")
  }
}
```

```
## Annie highest grade this semester is 100
## Hanna highest grade this semester is 100
```