# RWorksheet_Infiesto#4b

Infiesto

2024-10-30

```
#1. Using the for loop, create an R script that will display a 5x5 matrix as shown in Figure 1. It must
#Hint Use abs() function to get the absolute value

vectorA <- c(1, 2, 3, 4, 5)
matrixA <- matrix(0,5,5)
for (i in 1:5) {
  for (j in 1:5) {
  matrixA[i, j] <- abs(vectorA[i] - vectorA[j])
  }
}
print(matrixA)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    2    3    4
## [2,]    1    0    1    2    3
## [3,]    2    1    0    1    2
## [4,]    3    2    1    0    1
## [5,]    4    3    2    1    0
```

```
#2. Print the string "*" using for() function. The output should be the same as shown in Figure

for (i in 1:5) {
  line <- ""
  for (j in 1:i) {
    line <- paste(line, "*", sep = " ")
  }
  print(line)
}
```

```
## [1] " *"
## [1] " * *"
## [1] " * * *"
## [1] " * * * *"
## [1] " * * * * *"
```

```
#3. Get an input from the user to print the Fibonacci sequence starting from the 1st input up to 500. U

printFibonacci <- function(start) {
    if (is.na(start) || start < 1) {
        cat("Please enter a valid starting term (a positive integer).\n")
        return()
    }

    first <- 0
```

```
    second <- 1
    next_num <- 0
    current_term <- 1

    while (current_term < start) {
        next_num <- first + second
        first <- second
        second <- next_num
        current_term <- current_term + 1
    }

    repeat {
        if (next_num > 500) break
        cat(next_num, ", ")

        next_num <- first + second
        first <- second
        second <- next_num
    }
}

start <- as.numeric(readline(prompt = "Enter starting term: "))
```

```
## Enter starting term:
```

```
printFibonacci(start)
```

```
## Please enter a valid starting term (a positive integer).
```

```
## NULL
```

```
#4. Import the dataset as shown in Figure 1 you have created previously.

#a. What is the R script for importing an excel or a csv file? Display the first 6 rows of the dataset?
library(readr)
data <- read_csv("/cloud/project/ChristianLee/Worksheet#4/Dataset.csv", col_names = TRUE)
```

```
## New names:
## Rows: 17 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (2): Gender...3, Gender...6 dbl (4): Shoe size...1, Height...2, Shoe size...4,
## Height...5 lgl (2): ...7, ...8
## 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.
## * `Shoe size` -> `Shoe size...1`
## * `Height` -> `Height...2`
## * `Gender` -> `Gender...3`
## * `Shoe size` -> `Shoe size...4`
## * `Height` -> `Height...5`
## * `Gender` -> `Gender...6`
## * `` -> `...7`
## * `` -> `...8`
```

```
head(data)
```

```
## # A tibble: 6 x 8
```

```
##   `Shoe size...1` Height...2 Gender...3 `Shoe size...4` Height...5 Gender...6
##             <dbl>      <dbl> <chr>                <dbl>      <dbl> <chr>
## 1             6.5         66 F                     13           77 M
## 2             9           68 F                     11.5         72 M
## 3             8.5         64.5 F                    8.5         59 F
## 4             8.5         65 F                      5           62 F
## 5            10.5         70 M                     10           72 M
## 6             7           64 F                      6.5         66 F
## # i 2 more variables: ...7 <lgl>, ...8 <lgl>
```

```
#b. Create a subset for gender(female and male). How many observations are there in Male? How about in
```

```
num_females <- sum(data$Gender == "F")
```

```
## Warning: Unknown or uninitialised column: `Gender`.
```

```
num_males <- sum(data$Gender == "M")
```

```
## Warning: Unknown or uninitialised column: `Gender`.
```

```
num_females
```

```
## [1] 0
```

```
num_males
```

```
## [1] 0
```

```
#c. Create a graph for the number of males and females for Household Data. Use plot(), chart type = bar
gender_counts <- table(data$Gender)
```

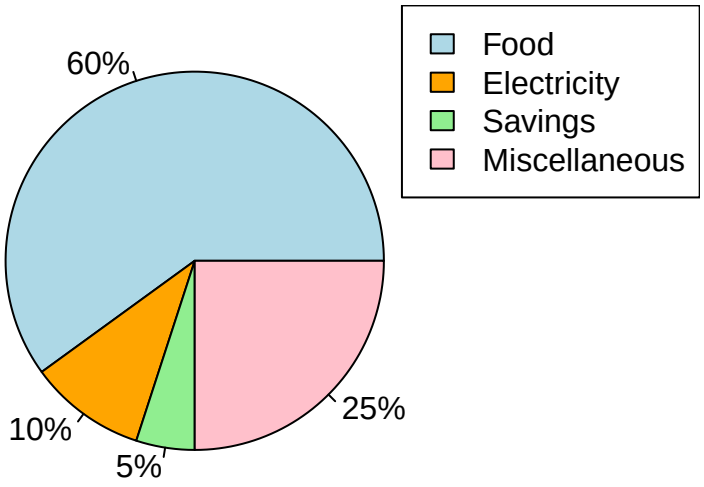
```
## Warning: Unknown or uninitialised column: `Gender`.
```

```
if (length(gender_counts) > 0) {
    barplot(gender_counts,
            main = "Number of Males and Females",
            xlab = "Gender",
            ylab = "Count",
            col = c("blue", "pink"),
            legend = names(gender_counts))
} else {
    print("No valid gender data found for plotting.")
}
```

```
## [1] "No valid gender data found for plotting."
```

```
expenses <- c(60, 10, 5, 25)
labels <- paste0(round(expenses / sum(expenses) * 100), "%")
colors <- c("lightblue", "orange", "lightgreen", "pink")

pie(expenses, labels = labels, col = colors, main = "Dela Cruz Family Monthly Expenses")
legend("topright", legend = c("Food", "Electricity", "Savings", "Miscellaneous"), fill = colors)
```

## Dela Cruz Family Monthly Expenses



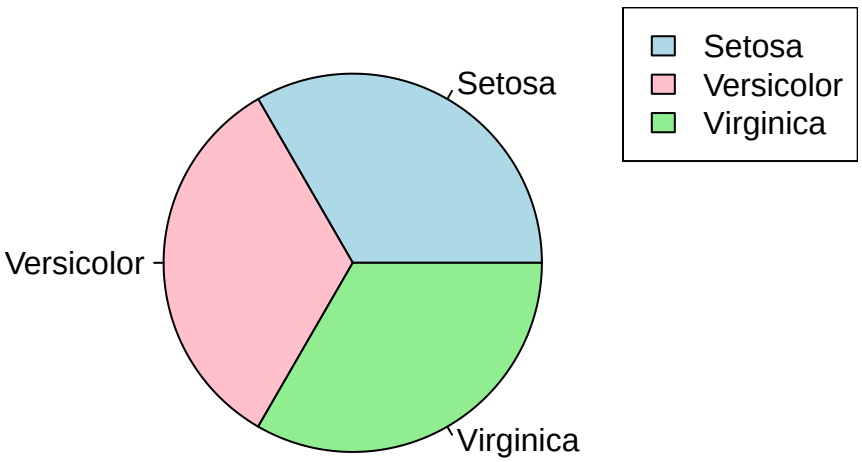

```r
#6. Use the iris dataset.
#data(iris)

#a. Check for the structure of the dataset using the str() function. Describe what you have seen in the

data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
#b. Create an R object that will contain the mean of the sepal.length, sepal.width,petal.length,and pet

mean_values <- colMeans(iris[, 1:4])
mean_values
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```r
#c. Create a pie chart for the Species distribution. Add title, legends, and colors. Write the R script

species_counts <- table(iris$Species)
pie(species_counts, main = "Species Distribution in Iris Dataset",
    col = c("lightblue", "pink", "lightgreen"),
    labels = c("Setosa", "Versicolor", "Virginica"))
legend("topright", legend = c("Setosa", "Versicolor", "Virginica"),
       fill = c("lightblue", "pink", "lightgreen"))
```




```
#d. Subset the species into setosa, versicolor, and virginica. Write the R scripts and show the last si

setosa <- subset(iris, Species == "setosa")
versicolor <- subset(iris, Species == "versicolor")
virginica <- subset(iris, Species == "virginica")

tail(setosa, 6)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 45          5.1         3.8          1.9         0.4  setosa
## 46          4.8         3.0          1.4         0.3  setosa
## 47          5.1         3.8          1.6         0.2  setosa
## 48          4.6         3.2          1.4         0.2  setosa
## 49          5.3         3.7          1.5         0.2  setosa
## 50          5.0         3.3          1.4         0.2  setosa
```

```
tail(versicolor, 6)
```

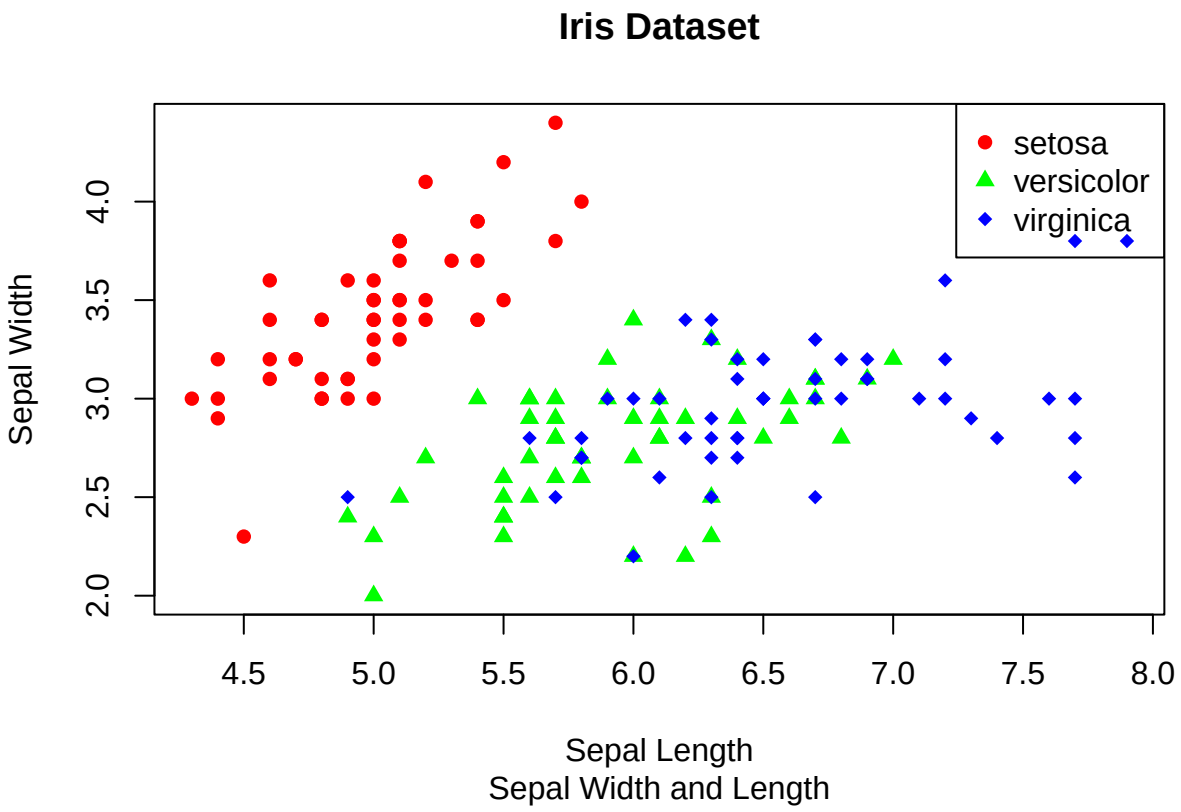
```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 95           5.6         2.7          4.2         1.3 versicolor
## 96           5.7         3.0          4.2         1.2 versicolor
## 97           5.7         2.9          4.2         1.3 versicolor
## 98           6.2         2.9          4.3         1.3 versicolor
## 99           5.1         2.5          3.0         1.1 versicolor
## 100          5.7         2.8          4.1         1.3 versicolor
```

```
tail(virginica, 6)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```
#e. Create a scatterplot of the sepal.length and sepal.width using the different species(setosa,versico
```

```r
iris$Species <- as.factor(iris$Species)

plot(iris$Sepal.Length, iris$Sepal.Width,
     col = c("red", "green", "blue")[as.numeric(iris$Species)],
     pch = c(16, 17, 18)[as.numeric(iris$Species)],
     xlab = "Sepal Length", ylab = "Sepal Width",
     main = "Iris Dataset", sub = "Sepal Width and Length")

legend("topright", legend = levels(iris$Species),
       col = c("red", "green", "blue"), pch = c(16, 17, 18))
```




```r
#f. Interpret the result.

# The scatterplot shows that Setosa is easy to identify because it has smaller sepals and forms a separ
```

```r
#7. Import the alexa-file.xlsx. Check on the variations. Notice that there are extra whitespaces among
```

```r
library(readxl)
library(knitr)

RObject <- read_excel("alexa-file.xlsx")

head(RObject)
```

```
## # A tibble: 6 x 5
##   rating date       variation  verified_reviews                    feedback
##    <dbl> <chr>      <chr>      <chr>                                  <dbl>
## 1      5 2018-07-30 Black Dot  It works great!!                           1
```

```
## 2      5 2018-07-30 Black Plus PHENOMENAL                                    1
## 3      5 2018-07-30 Black Show I used it to control my smart home devi~      1
## 4      4 2018-07-29 Black Spot Very convenient                               1
## 5      4 2018-07-29 White Dot  A decent buy                                  0
## 6      3 2018-07-28 White Plus Good value for money                          1
```

```
#a. Rename the white and black variants by using gsub() function.

RObject$variation <- gsub("Black Dot", "Black_Dot", RObject$variation)
RObject$variation <- gsub("Black Plus", "Black_Plus", RObject$variation)
RObject$variation <- gsub("Black Show", "Black_Show", RObject$variation)
RObject$variation <- gsub("Black Spot", "Black_Spot", RObject$variation)

RObject$variation <- gsub("White Dot", "White_Dot", RObject$variation)
RObject$variation <- gsub("White Plus", "White_Plus", RObject$variation)
RObject$variation <- gsub("White Show", "White_Show", RObject$variation)
RObject$variation <- gsub("White Spot", "White_Spot", RObject$variation)

head(RObject)
```

```
## # A tibble: 6 x 5
##   rating date       variation  verified_reviews                          feedback
##    <dbl> <chr>      <chr>      <chr>                                        <dbl>
## 1      5 2018-07-30 Black_Dot  It works great!!                                 1
## 2      5 2018-07-30 Black_Plus PHENOMENAL                                       1
## 3      5 2018-07-30 Black_Show I used it to control my smart home devi~         1
## 4      4 2018-07-29 Black_Spot Very convenient                                  1
## 5      4 2018-07-29 White_Dot  A decent buy                                     0
## 6      3 2018-07-28 White_Plus Good value for money                             1
```

```
#b. Get the total number of each variations and save it into another object. Save the object as variati

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

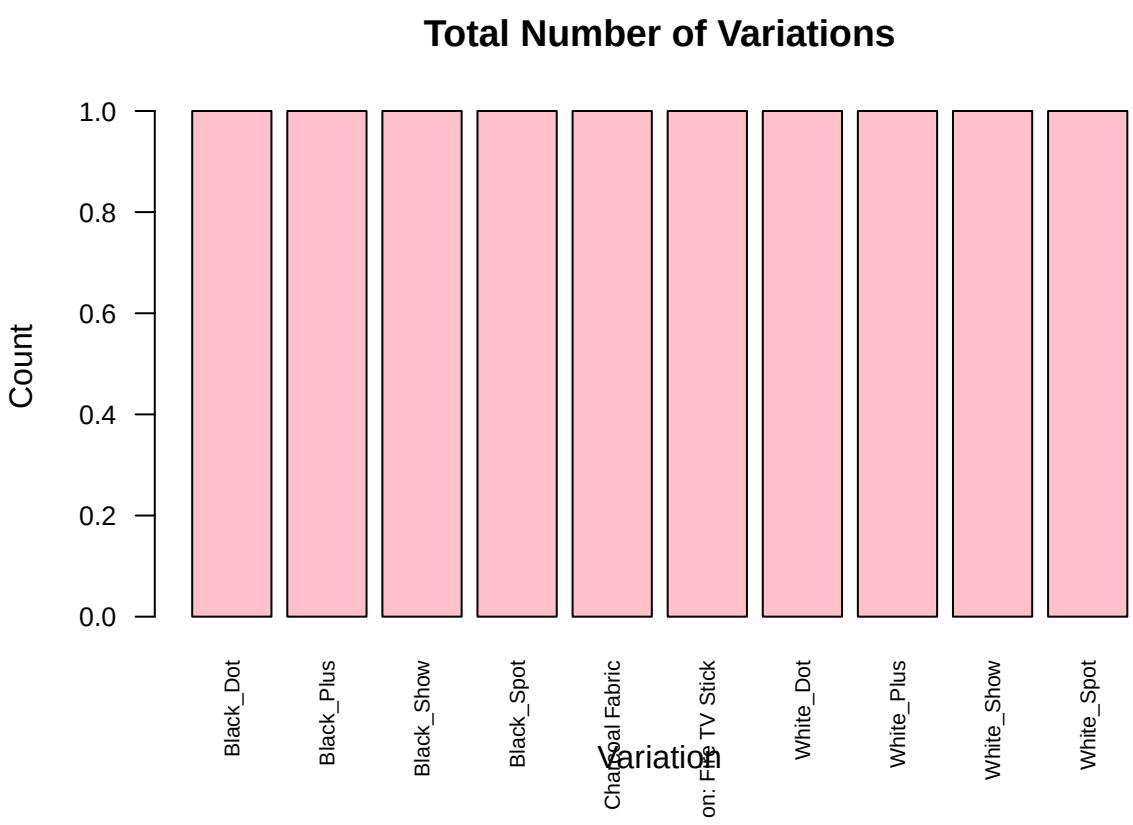
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
variations_summary <- RObject %>%
  count(variation)

save(variations_summary, file = "variations.RData")
print(variations_summary)
```

```
## # A tibble: 10 x 2
##    variation                      n
##    <chr>                      <int>
##  1 Black_Dot                      1
##  2 Black_Plus                     1
##  3 Black_Show                     1
```

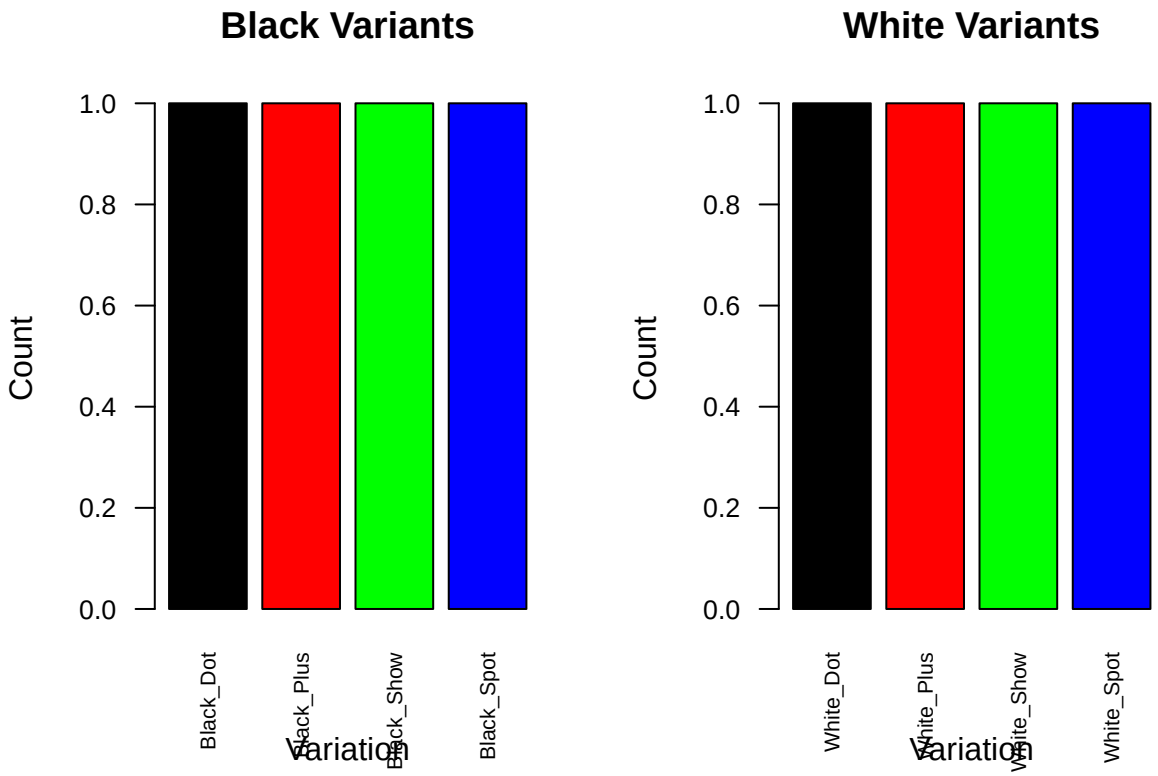
```
##  4 Black_Spot                       1
##  5 Charcoal Fabric                  1
##  6 Configuration: Fire TV Stick     1
##  7 White_Dot                        1
##  8 White_Plus                       1
##  9 White_Show                       1
## 10 White_Spot                       1
```

```r
#c. From the variations.RData, create a barplot(). Complete the details of the chart which include the

load("variations.RData")

barplot(variations_summary$n,
        names.arg = variations_summary$variation,
        col = "pink",
        main = "Total Number of Variations",
        xlab = "Variation",
        ylab = "Count",
        las = 2,
        cex.names = 0.7,
        cex.axis = 0.8)
```




```r
#d. Create a barplot() for the black and white variations. Plot it in 1 frame, side by side. Complete t

black_white_variations <- variations_summary %>%
  filter(grepl("Black|White", variation))

black_variations <- black_white_variations %>% filter(grepl("Black", variation))
white_variations <- black_white_variations %>% filter(grepl("White", variation))
```

```r
par(mfrow = c(1, 2))

barplot(black_variations$n,
        names.arg = black_variations$variation,
        col = c("black", "red", "green", "blue", "cyan"),
        main = "Black Variants",
        xlab = "Variation",
        ylab = "Count",
        las = 2,
        cex.names = 0.7,
        cex.axis = 0.8)

barplot(white_variations$n,
        names.arg = white_variations$variation,
        col = c("black", "red", "green", "blue", "cyan"),
        main = "White Variants",
        xlab = "Variation",
        ylab = "Count",
        las = 2,
        cex.names = 0.7,
        cex.axis = 0.8)
```




```r
par(mfrow = c(1, 1))
```