RWorksheet_Orada#4b

Jessamine Paula Orada

2025-12-16

```
install.packages("readxl")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)
```

```
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.5'
## (as 'lib' is unspecified)
```

```
 library(readxl)
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
```

##for() loop ##1. Using the for loop, create an R script that will display a 5x5 matrix as shown in ##Figure 1. It must contain vectorA = [1,2,3,4,5] and a 5 x 5 zero matrix.

```
# 1. Initialize vectorA and the zero matrix
vectorA <- c(1, 2, 3, 4, 5)
matrixB <- matrix(0, nrow = 5, ncol = 5)

# 2. Fill the matrix using a nested for loop and the absolute difference
for (i in 1:5) {
  for (j in 1:5) {
    # The value is the absolute difference between the row index (i) and the column value from vectorA
    # However, observing the pattern (0, 1, 2, 3, 4 in the 4th row; 1, 0, 1, 2, 3 in the 5th row; etc.)
    # the value is |row_index - col_index|.
    # For a 1-based index: |(i-1) - (j-1)| = |i - j|
    matrixB[i, j] <- abs(i - j)
  }
}

# The desired pattern appears in the last 5 rows of the combined matrix in Figure 1.
# Let's adjust the indices to match the *specific* pattern shown (which seems to start from |row-1 - co

# Re-initialize for the specific pattern in Figure 1 (rows 4 to 8 of the whole figure)
```

```r
# Let's assume the rows 4 and 5 (1-based index) of the desired output correspond to i=1 and i=2 in a 5x
matrixC <- matrix(0, nrow = 5, ncol = 5)

for (i in 1:5) {
  for (j in 1:5) {
    # The pattern is based on the absolute difference between the column index (j) and the row index (i
    # For the 4th row (i=1 in the 5x5 pattern): abs(j-1) -> 0, 1, 2, 3, 4
    # For the 5th row (i=2 in the 5x5 pattern): abs(j-2) -> 1, 0, 1, 2, 3
    # For the 6th row (i=3 in the 5x5 pattern): abs(j-3) -> 2, 1, 0, 1, 2
    # This suggests the value is |j - i| (absolute difference between column index and row index).
    matrixC[i, j] <- abs(j - i)
  }
}

# Display the matrix (this matches the pattern in Figure 1)
print(matrixC)
```

```
##      [,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

```r
# 1. Define the number of rows and the pattern (pyramid-like)
# The pattern is: *, ***, *****, ******* (1, 3, 5, 7 stars)
num_rows <- 4

for (i in 1:num_rows) {
  # Calculate the number of asterisks for the current row: 2*i - 1
  num_stars <- 2 * i - 1

  # Calculate the number of leading spaces to center the pattern
  num_spaces <- num_rows - i

  # Create the string of stars
  star_string <- paste(rep("*", num_stars), collapse = "")

  # Create the string of spaces
  space_string <- paste(rep(" ", num_spaces), collapse = "")

  # Combine and print
  cat(space_string, star_string, "\n")
}
```

```
##     *
##    ***
##   *****
##  *******
```

##3. Get an input from the user to print the Fibonacci sequence starting from the 1st input up to 500. Use repeat and break statements. Write the R Scripts and its output.

```r
# 1. Get the starting number from the user
# Note: For a standard Fibonacci sequence (starting 0, 1 or 1, 1),
# the '1st input' is usually the starting element.
# Since the sequence is defined by the previous two numbers, we'll ask for the first two.
# For simplicity, we'll assume the user's input is the first term, 'a',
# and the second term, 'b', is 1, so the sequence starts a, 1, a+1, ...
# Standard approach: The Fibonacci sequence starts with two known values, usually 0 and 1, or 1 and 1.
# Since the prompt asks for "1st input up to 500," we'll start with the first two standard terms,
# but stop the sequence *if* the *next* term exceeds the user's input (or 500, whichever is smaller).
# The prompt is slightly ambiguous, but standard Fibonacci up to 500 is the most likely intent.

# Let's ask for the maximum limit (500 is specified, but good practice to ask)
# max_limit <- as.numeric(readline(prompt = "Enter the maximum limit for the Fibonacci sequence (e.g., 
max_limit <- 500

# Start with the first two Fibonacci numbers
a <- 0
b <- 1

cat("Fibonacci Sequence (up to", max_limit, "):\n")
```

```
## Fibonacci Sequence (up to 500 ):
```

```r
repeat {
  # Print the current term 'a'
  cat(a, " ")

  # Calculate the next term
  next_term <- a + b

  # Check the break condition (if the next term exceeds the limit)
  if (next_term > max_limit) {
    break
  }

  # Update 'a' and 'b' for the next iteration
  a <- b
  b <- next_term
}
```

```
## 0  1  1  2  3  5  8  13  21  34  55  89  144  233
```

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

##Using Basic Graphics (plot(),barplot(),pie(),hist()) ##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? Show your codes and its result

```r
# R Script for importing a CSV file
# Use 'read.csv()' for CSV files
# Use 'readxl::read_excel()' for Excel files (.xlsx) - requires installing 'readxl' package.

# Assuming the data is saved as a CSV file named 'shoe_sizes.csv'
# dataset <- read.csv("shoe_sizes.csv", header = TRUE)
# Since the data in Figure 3 is clearly structured, let's recreate it manually for reproducibility.

```

```
shoe_data <- 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,
  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,
  Gender = c("F", "F", "F", "F", "M", "F", "F", "F", "M", "F", "M", "F", "M", "M", "M", "M", "F", "F",
)

# Display the first 6 rows
head(shoe_data)
```

```
##   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
```

##b. Create a subset for gender(female and male). How many observations are there in Male? How about in Female? Write the R scripts and its output.

```
# Create subsets
female_data <- subset(shoe_data, Gender == "F")
male_data <- subset(shoe_data, Gender == "M")

# Count the observations
n_male <- nrow(male_data)
n_female <- nrow(female_data)

cat("Number of observations in Male subset:", n_male, "\n")
```

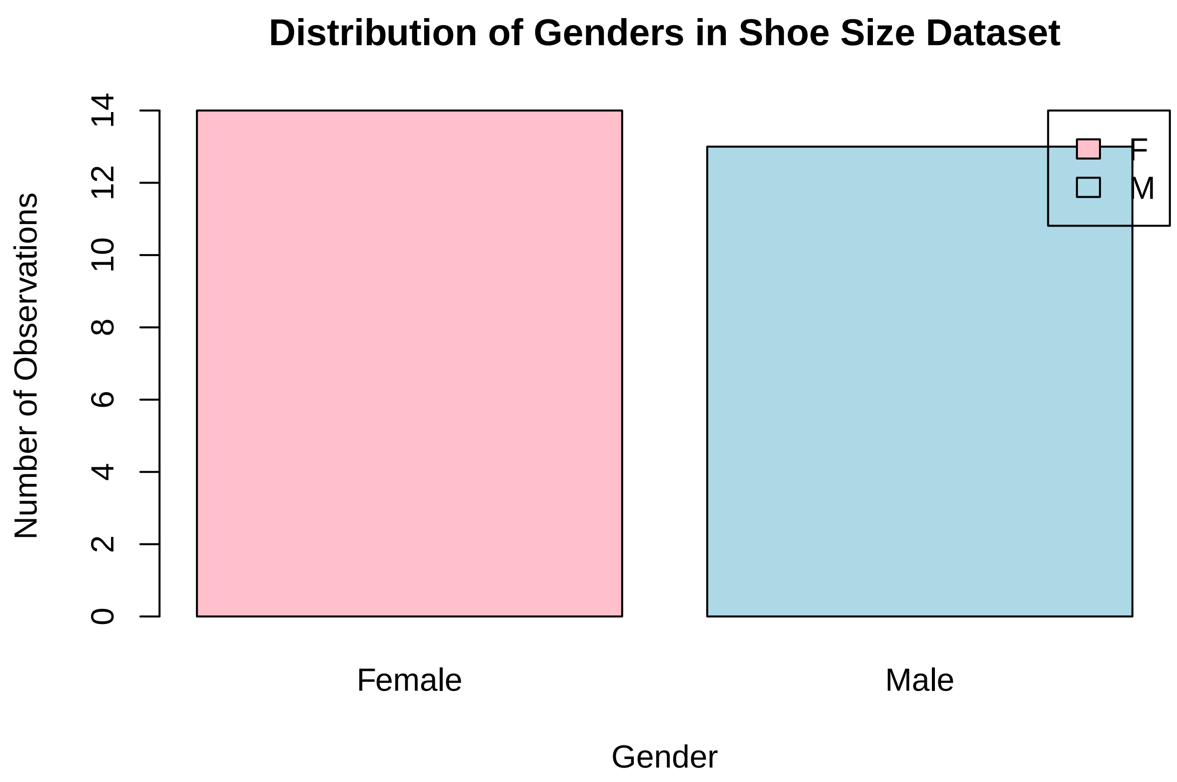
```
## Number of observations in Male subset: 13
```

```
cat("Number of observations in Female subset:", n_female, "\n")
```

```
## Number of observations in Female subset: 14
```

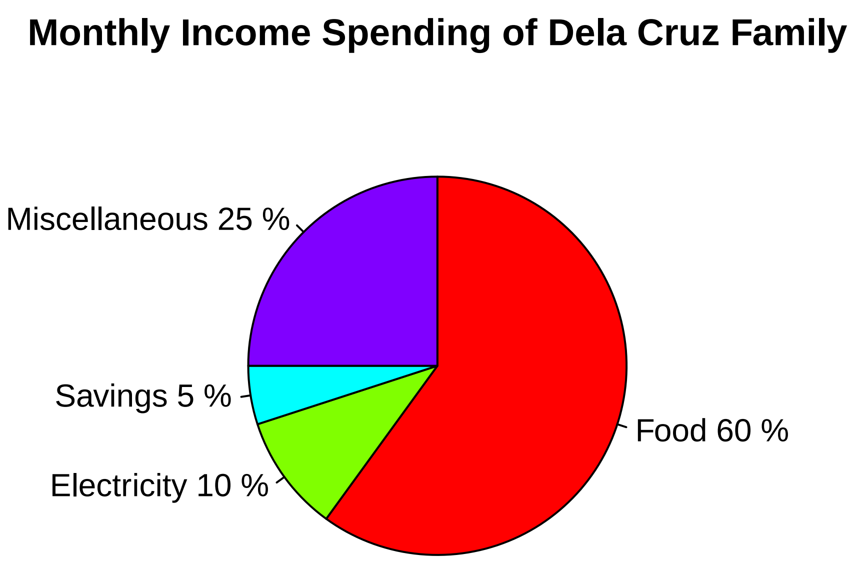
##c. Create a graph for the number of males and females for Household Data. Use plot(), chart type = barplot. Make sure to place title, legends, and colors. Write the R scripts and its result.

```
# Count the occurrences of each gender
gender_counts <- table(shoe_data$Gender)

# Create the barplot
barplot(
  gender_counts,
  main = "Distribution of Genders in Shoe Size Dataset", # Title
  xlab = "Gender",                                        # X-axis label
  ylab = "Number of Observations",                        # Y-axis label
  col = c("pink", "lightblue"),                           # Colors (Female, Male)
  legend.text = names(gender_counts),                     # Legend text
  args.legend = list(x = "topright"),                     # Legend position
  names.arg = c("Female", "Male")                         # Names under the bars
)
```



##5. The monthly income of Dela Cruz family was spent on the following: ##a. Create a piechart that will include labels in percentage.Add some colors and title of the chart. Write the R scripts and show its output.

```r
# Define the expenses and their percentages
expenses <- c(60, 10, 5, 25)
names(expenses) <- c("Food", "Electricity", "Savings", "Miscellaneous")

# Calculate the percentages for labels
percentages <- round(expenses / sum(expenses) * 100)
labels <- paste(names(expenses), percentages, "%", sep = " ")

# Create the pie chart
pie(
  expenses,
  main = "Monthly Income Spending of Dela Cruz Family", # Title
  labels = labels,                                      # Labels with percentages
  col = rainbow(length(expenses)),                      # Colors
  clockwise = TRUE                                      # Plot slices clockwise
)
```



##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 output.

```r
# Load the iris dataset (already built-in)
data(iris)

# Check the structure
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 ...
```

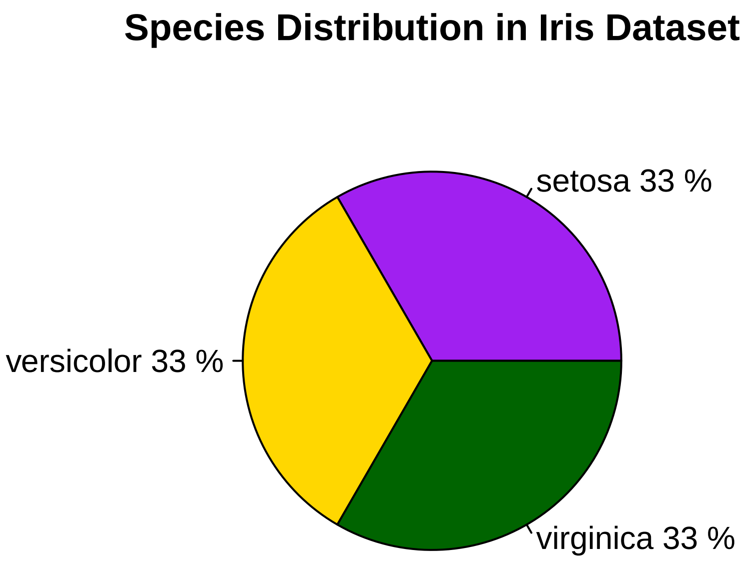
##b. Create an R object that will contain the mean of the sepal.length, sepal.width,petal.length,and petal.width. What is the R script and its result?

```r
# Calculate the mean of each column
mean_sepal_length <- mean(iris$Sepal.Length)
mean_sepal_width <- mean(iris$Sepal.Width)
mean_petal_length <- mean(iris$Petal.Length)
mean_petal_width <- mean(iris$Petal.Width)

# Store the means in an R object (e.g., a named vector)
iris_means <- c(
  Sepal.Length = mean_sepal_length,
  Sepal.Width = mean_sepal_width,
  Petal.Length = mean_petal_length,
  Petal.Width = mean_petal_width
)

# Display the result
print(iris_means)
```

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

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

```r
# Get the counts for each species
species_counts <- table(iris$Species)

# Calculate percentages for labels
percentages <- round(species_counts / sum(species_counts) * 100)
labels <- paste(names(species_counts), percentages, "%", sep = " ")

# Create the pie chart
pie(
  species_counts,
  main = "Species Distribution in Iris Dataset",    # Title
  labels = labels,                                   # Labels with percentages
  col = c("purple", "gold", "darkgreen")             # Colors for the 3 species
)
```



##d. Subset the species into setosa, versicolor, and virginica. Write the R scripts and show the last six (6) rows of each species.

```r
# 1. Subset the species
setosa <- subset(iris, Species == "setosa")
versicolor <- subset(iris, Species == "versicolor")
virginica <- subset(iris, Species == "virginica")

# 2. Show the last six rows of each species using tail()
cat("Last six rows of Setosa:\n")
```

```
## Last six rows of Setosa:
```

```r
print(tail(setosa))
```

```
##    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
```

```r
cat("\nLast six rows of Versicolor:\n")
```

```
##
## Last six rows of Versicolor:
```

```r
print(tail(versicolor))
```

```
##     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
```

```r
cat("\nLast six rows of Virginica:\n")
```

```
##
## Last six rows of Virginica:
```

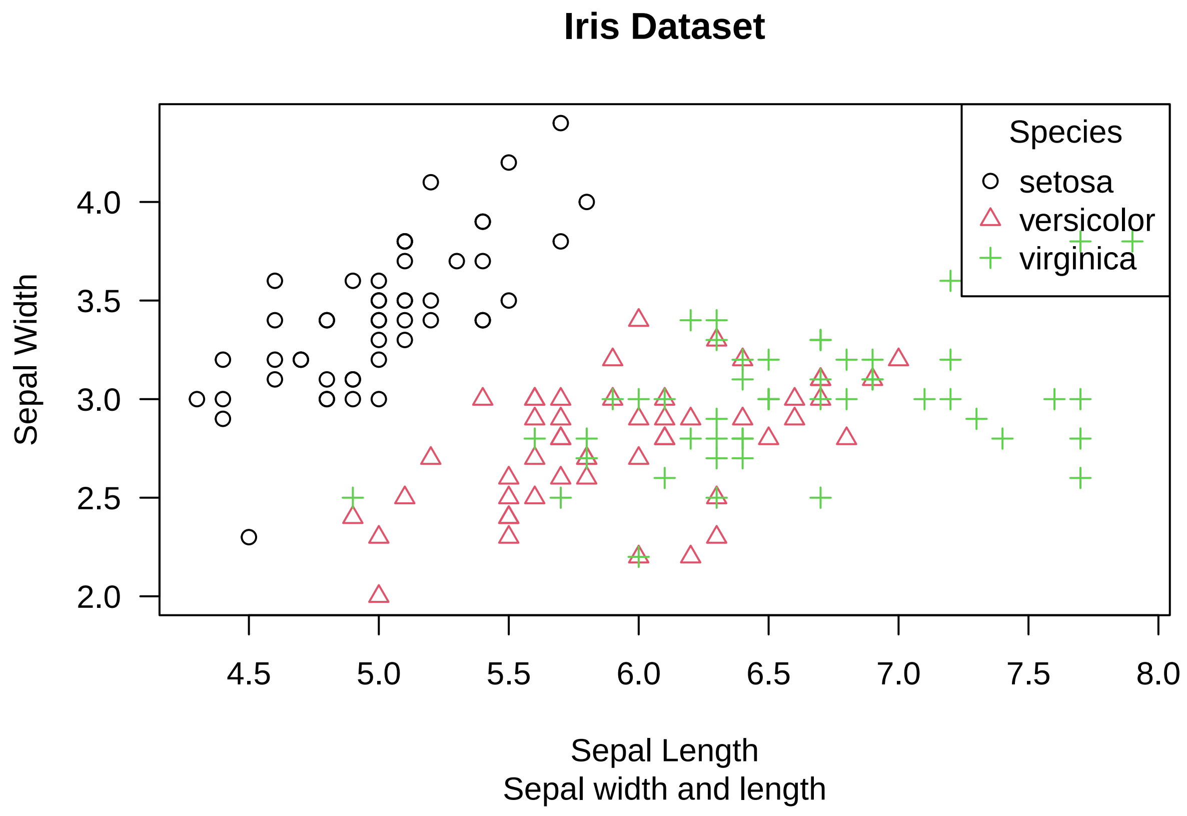
```r
print(tail(virginica))
```

```
##     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,versicolor,virginica). Add a title = "Iris Dataset", subtitle = "Sepal width and length, labels for the x and y axis, the pch symbol and colors should be based on the species.

##Hint: Need to convert to factors the species to store categorical variables.

```r
# Create the scatterplot
plot(
  iris$Sepal.Length,
  iris$Sepal.Width,
  main = "Iris Dataset",                    # Main Title
  sub = "Sepal width and length",           # Subtitle
  xlab = "Sepal Length",                    # X-axis label
  ylab = "Sepal Width",                     # Y-axis label
  # Use the Species factor to determine color (col) and plotting character (pch)
  col = as.numeric(iris$Species),           # Colors: 1=setosa, 2=versicolor, 3=virginica
  pch = as.numeric(iris$Species),           # Symbols: 1=circle, 2=triangle, 3=plus
  las = 1                                   # Label orientation (horizontal)
)

# Add a legend to distinguish the species
legend(
  "topright",
  legend = levels(iris$Species),
  col = 1:3,
  pch = 1:3,
  title = "Species"
)
```



##f. Interpret the result.

```
##Interpretation: The scatterplot clearly shows that the three Iris species can be distinctly separated

##Iris setosa (e.g., blue circles) typically has the shortest sepal lengths and the widest sepal widths

##Iris versicolor (e.g., red triangles) and Iris virginica (e.g., green pluses) have longer sepal lengt
```

##a.

```r
variation_vec <- c("Black Dot ", "Black Dot ", "Black Dot ", "Black Dot ", "Black Dot ",
                   "White Plus ", "White Spot ", "Black Show ", "White Dot ", "Black Spot ")
alexa_data <- data.frame(variation = variation_vec)


alexa_data$variation <- gsub ("\\s+$", "", alexa_data$variation)

# Show a snippet of the output
print(head(alexa_data))
```

```
##    variation
## 1  Black Dot
## 2  Black Dot
## 3  Black Dot
## 4  Black Dot
## 5  Black Dot
## 6 White Plus
```

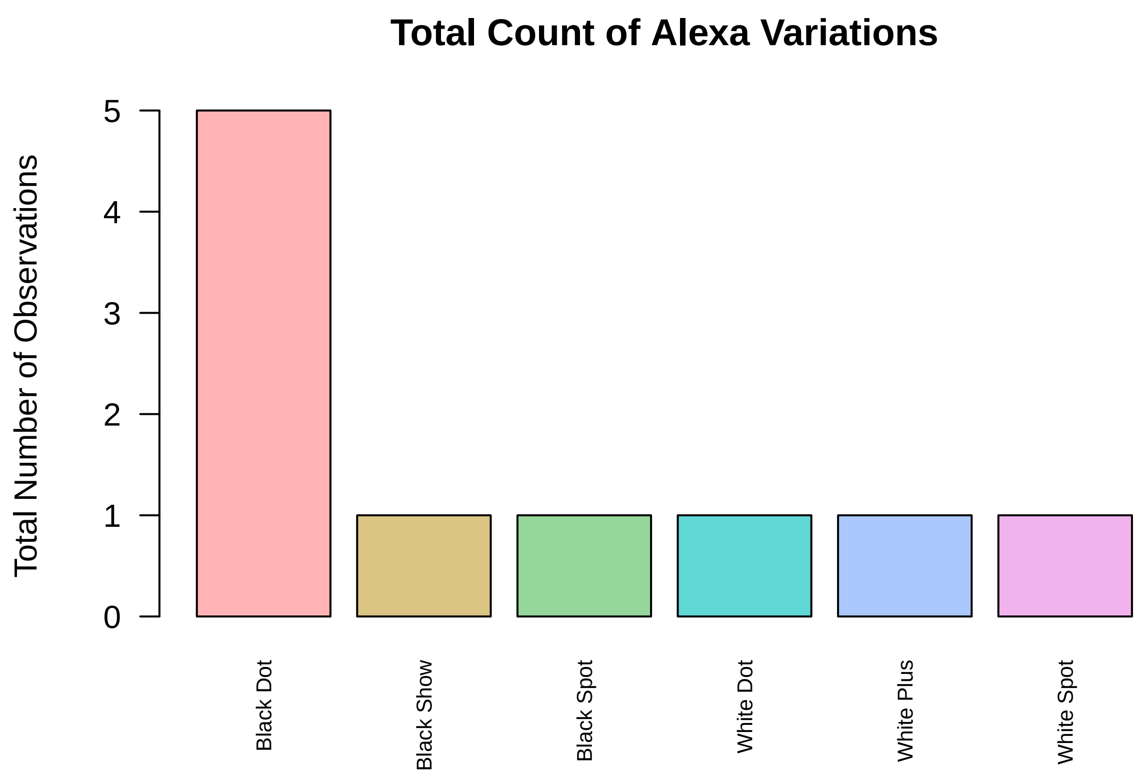
##b.

```
# Count the total number of each variation
variation_counts <- alexa_data %>%
  count(variation)

# Rename the count column to 'Total' (as suggested by Figure 5)
variation_counts <- rename(variation_counts, Total = n)

# Save the object as variations.RData
save(variation_counts, file = "variations.RData")

# Print the result
print(variation_counts)
```

```
##    variation Total
## 1  Black Dot     5
## 2 Black Show     1
## 3 Black Spot     1
## 4  White Dot     1
## 5 White Plus     1
## 6 White Spot     1
```

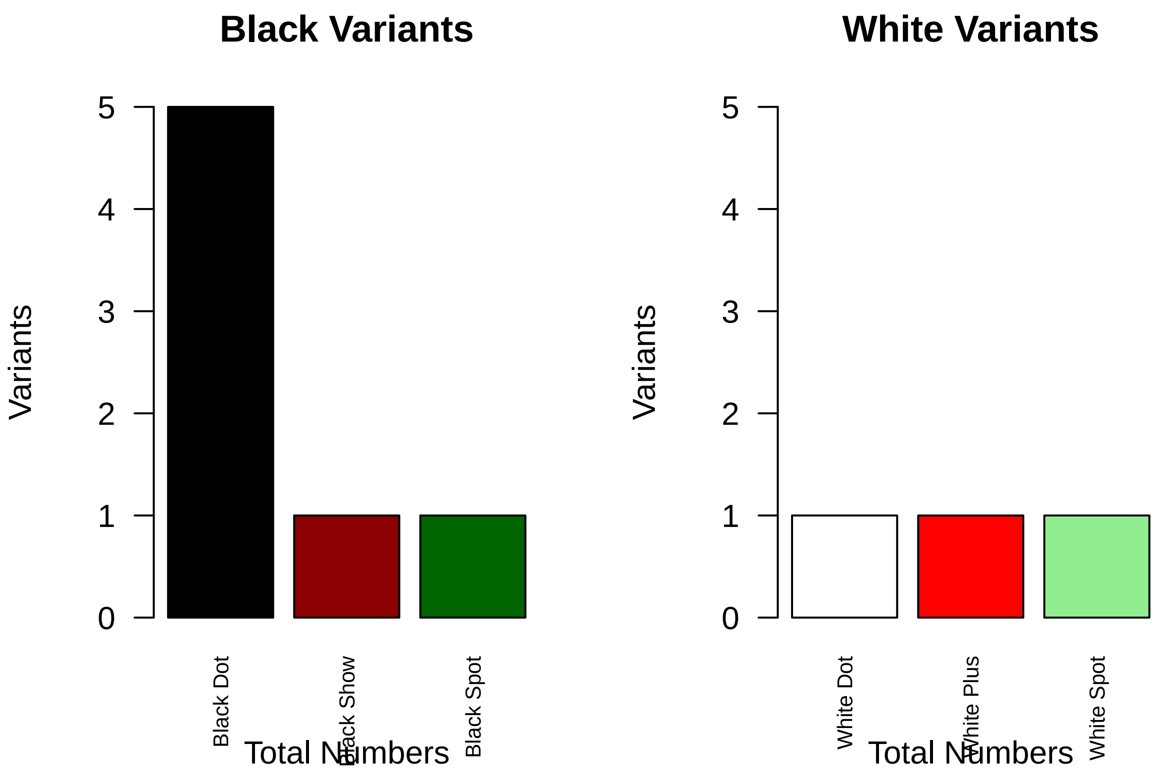
##c.

```
load("variations.RData")

# Create the barplot
barplot(
  variation_counts$Total,
  names.arg = variation_counts$variation, # Labels for each bar
  main = "Total Count of Alexa Variations",
  ylab = "Total Number of Observations",
  col = hcl.colors(nrow(variation_counts), "Set 3"), # A set of distinct colors
  las = 2, # Rotate x-axis labels vertically for readability
  cex.names = 0.7 # Reduce label size
)
```



##d

```r
# 1. Filter the black and white variants from the counts
black_variants <- variation_counts %>%
  filter(grepl("Black", variation))

white_variants <- variation_counts %>%
  filter(grepl("White", variation))

# 2. Set up the plotting area for 1 row and 2 columns
par(mfrow = c(1, 2))

# 3. Barplot for Black Variants
barplot(
  black_variants$Total,
  names.arg = black_variants$variation,
  main = "Black Variants",
  ylab = "Variants",
  xlab = "Total Numbers",
  col = c("black", "darkred", "darkgreen", "darkblue", "cyan"),
  ylim = c(0, max(variation_counts$Total)), # Use a consistent y-axis limit
  las = 2,
  cex.names = 0.7
)

# 4. Barplot for White Variants
barplot(
  white_variants$Total,
  names.arg = white_variants$variation,
  main = "White Variants",
  ylab = "Variants",
  xlab = "Total Numbers",
  col = c("white", "red", "lightgreen", "lightblue", "cyan"),
  border = "black", # Add border to distinguish white bars
  ylim = c(0, max(variation_counts$Total)), # Use a consistent y-axis limit
  las = 2,
  cex.names = 0.7
)
```



```r
# Reset the plotting area layout
par(mfrow = c(1, 1))
```