RWORKSHEET_GENTAPAO#4B

BEA JULIETTE L. GENTAPAO

2025-12-22

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.

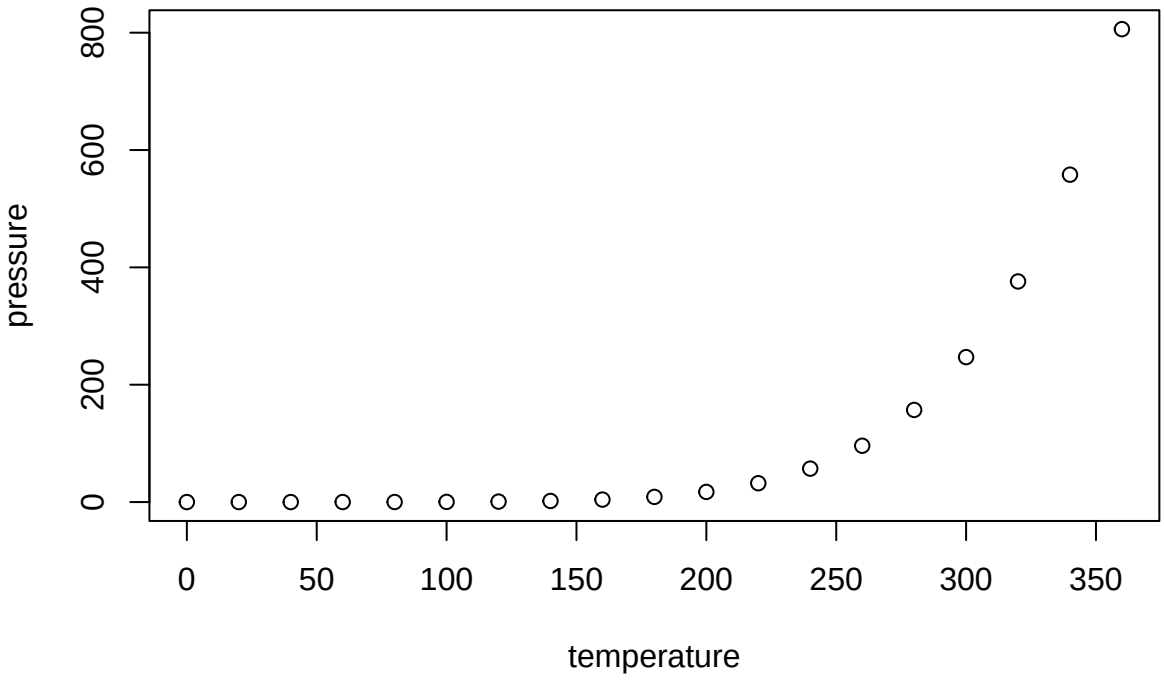
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
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.

1. Create a 5x5 matrix using for loop

```
vectorA <- c(1, 2, 3, 4, 5)
matrix_result <- matrix(0, nrow = 5, ncol = 5)

for (i in 1:5) {
  for (j in 1:5) {
    matrix_result[i, j] <- abs(vectorA[i] - vectorA[j])
  }
}

print(matrix_result)
```

```
##      [,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 string "*" pattern using for loop

```
for (i in 1:5) {
  stars <- ""
  for (j in 1:i) {
    stars <- paste0(stars, "*")
  }
  cat(stars, "\n")
}
```

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

3. Fibonacci sequence up to 500

```
cat("Enter the starting number for Fibonacci sequence: ")
```

```
## Enter the starting number for Fibonacci sequence:
```

```
# For demonstration, we'll use 1 as starting point
start <- 1

fib1 <- 0
fib2 <- 1
fibonacci_seq <- c()

cat("Fibonacci sequence starting from", start, "up to 500:\n")
```

```
## Fibonacci sequence starting from 1 up to 500:
```

```
repeat {
  if (fib2 > 500) {
    break
  }

  if (fib2 >= start) {
    fibonacci_seq <- c(fibonacci_seq, fib2)
    cat(fib2, " ")
  }

  next_fib <- fib1 + fib2
  fib1 <- fib2
  fib2 <- next_fib
}
```

```
## 1  1  2  3  5  8  13  21  34  55  89  144  233  377
```

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

Using Basic Graphics

4. Household Data Analysis

a. Import dataset and display first 6 rows

```
# Create the shoe size dataset first
household_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, 8.5, 5.0,
                10.0, 6.5, 7.5, 8.5, 10.5, 8.5, 10.5, 11.0, 9.0, 13.0),
  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, 72.0, 59.0, 62.0,
             72.0, 66.0, 64.0, 67.0, 73.0, 69.0, 72.0, 70.0, 69.0, 70.0),
  Gender = c("F", "F", "F", "F", "M", "F", "F", "F", "M", "F", "M",
             "F", "M", "M", "M", "M", "F", "F", "M", "F", "F", "M",
             "M", "F", "M", "M", "M", "M")
)

# Save as CSV
write.csv(household_data, "household_data.csv", row.names = FALSE)

# Import the CSV file
imported_data <- read.csv("household_data.csv")

# Display first 6 rows
cat("First 6 rows of the dataset:\n")
```

```
## First 6 rows of the dataset:
```

```
head(imported_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 subset for gender and count observations

```
# Subset for males
males_subset <- imported_data[imported_data$Gender == "M", ]
cat("Number of Male observations:", nrow(males_subset), "\n\n")
```

```
## Number of Male observations: 14
```

```
print(males_subset)
```

```
##    Shoe_Size Height Gender
## 5       10.5   70.0      M
## 9       13.0   72.0      M
## 11      10.5   74.5      M
## 13      12.0   71.0      M
## 14      10.5   71.0      M
```

```
## 15      13.0   77.0      M
## 16      11.5   72.0      M
## 19      10.0   72.0      M
## 22       8.5   67.0      M
## 23      10.5   73.0      M
## 25      10.5   72.0      M
## 26      11.0   70.0      M
## 27       9.0   69.0      M
## 28      13.0   70.0      M
```

```r
# Subset for females
females_subset <- imported_data[imported_data$Gender == "F", ]
cat("\nNumber of Female observations:", nrow(females_subset), "\n\n")
```

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

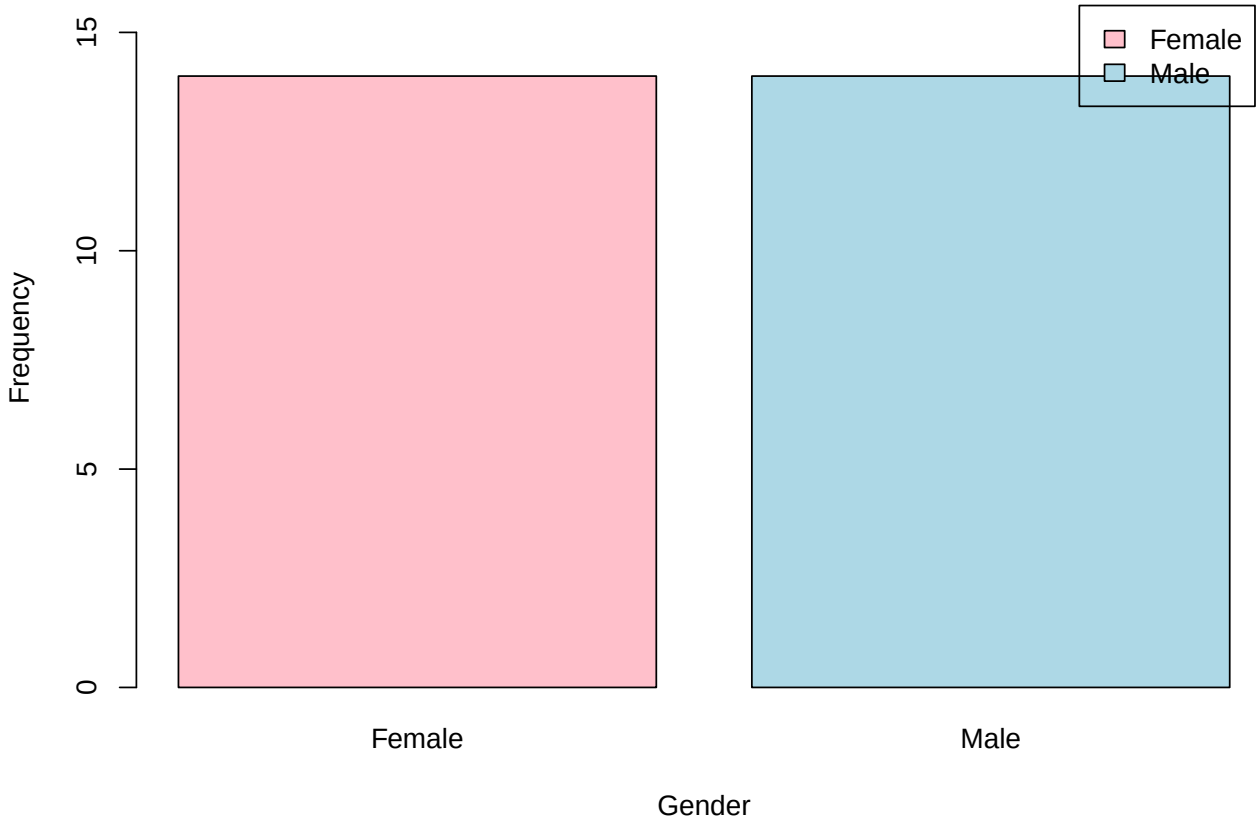
```r
print(females_subset)
```

```
##    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
## 6        7.0   64.0      F
## 7        9.5   70.0      F
## 8        9.0   71.0      F
## 10       7.5   64.0      F
## 12       8.5   67.0      F
## 17       8.5   59.0      F
## 18       5.0   62.0      F
## 20       6.5   66.0      F
## 21       7.5   64.0      F
## 24       8.5   69.0      F
```

c. Create barplot for number of males and females

```r
# Count males and females
gender_counts <- table(imported_data$Gender)

# Create barplot
barplot(gender_counts,
        main = "Number of Males and Females in Household Data",
        xlab = "Gender",
        ylab = "Frequency",
        col = c("pink", "lightblue"),
        legend.text = c("Female", "Male"),
        names.arg = c("Female", "Male"),
        ylim = c(0, max(gender_counts) + 2))
```

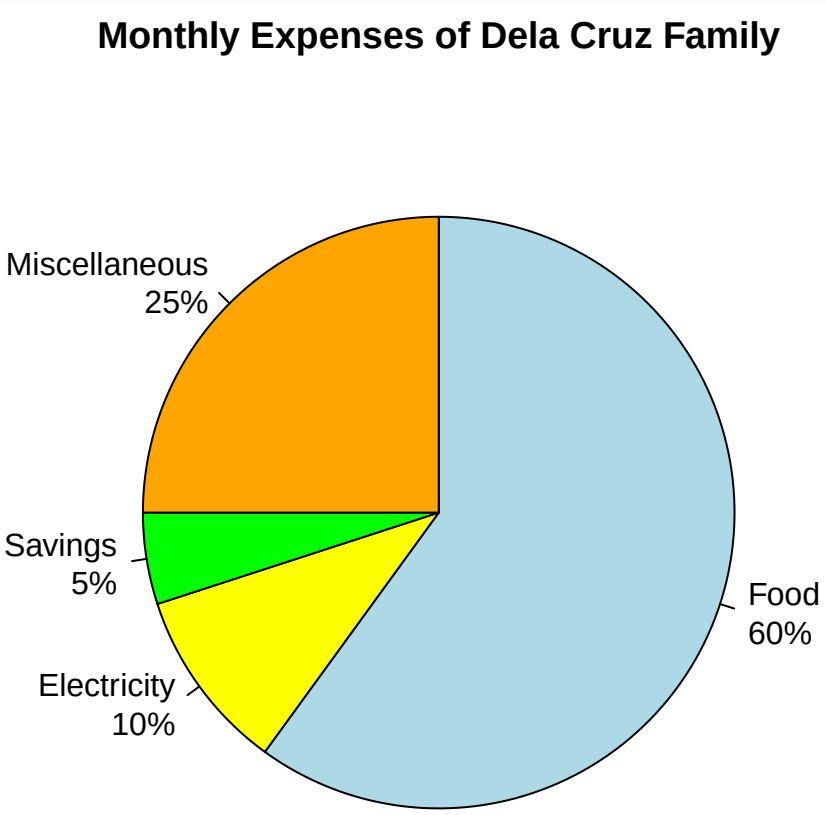

Number of Males and Females in Household Data
Female
Male
Frequency
0
5
10
15
Female
Male
Gender

5. Dela Cruz Family Monthly Expenses

```r
# Create expense data
expenses <- c(60, 10, 5, 25)
expense_labels <- c("Food", "Electricity", "Savings", "Miscellaneous")

# Calculate percentages
percentages <- round(expenses / sum(expenses) * 100, 1)
pie_labels <- paste(expense_labels, "\n", percentages, "%", sep = "")

# Create pie chart
pie(expenses,
    labels = pie_labels,
    main = "Monthly Expenses of Dela Cruz Family",
    col = c("lightblue", "yellow", "green", "orange"),
    clockwise = TRUE)
```



6. Iris Dataset Analysis

a. Check structure of iris dataset

```
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 ...
```

Description: The iris dataset contains 150 observations of 5 variables. There are 4 numeric variables (Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) measured in centimeters, and 1 factor variable (Species) with 3 levels: setosa, versicolor, and virginica. Each species has 50 observations.

b. Create object containing means

```
sepal_length_mean <- mean(iris$Sepal.Length)
sepal_width_mean <- mean(iris$Sepal.Width)
petal_length_mean <- mean(iris$Petal.Length)
petal_width_mean <- mean(iris$Petal.Width)

iris_means <- c(
  Sepal.Length = sepal_length_mean,
  Sepal.Width = sepal_width_mean,
  Petal.Length = petal_length_mean,
  Petal.Width = petal_width_mean
)

cat("Mean values of iris measurements:\n")
```

```
## Mean values of iris measurements:
```

```
print(iris_means)
```

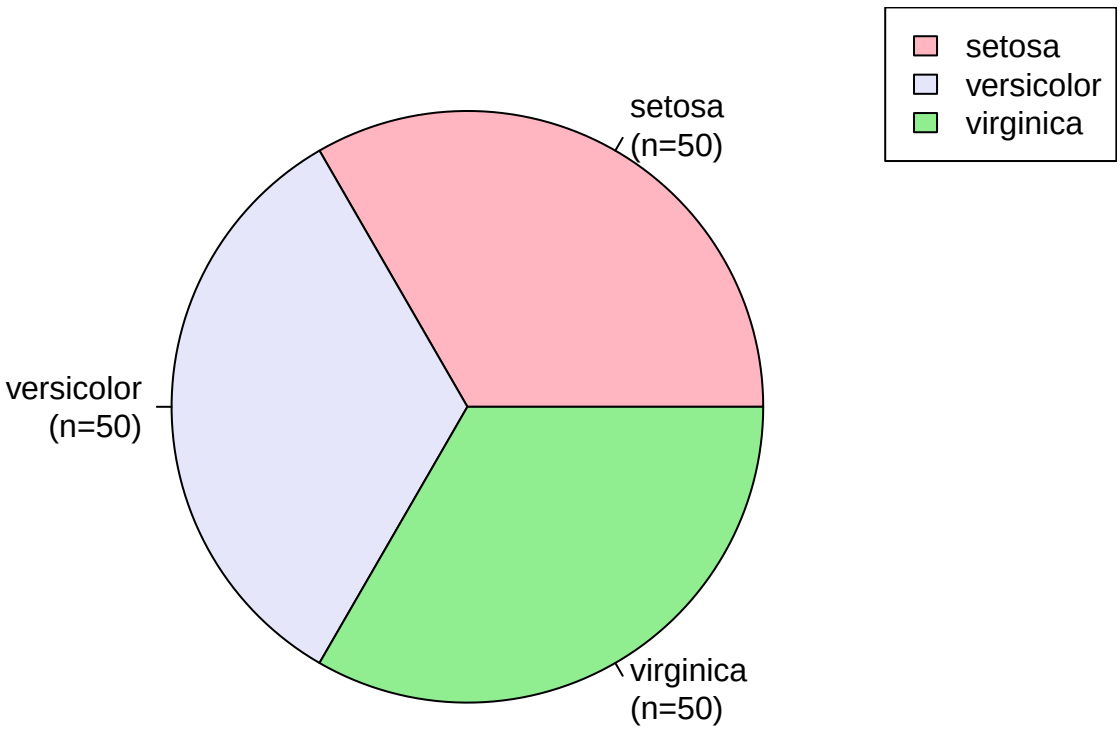
```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

c. Create pie chart for Species distribution

```
species_counts <- table(iris$Species)

pie(species_counts,
    main = "Distribution of Iris Species",
    col = c("lightpink", "lavender", "lightgreen"),
    labels = paste(names(species_counts), "\n(n=", species_counts, ")", sep = ""))

legend("topright",
       legend = names(species_counts),
       fill = c("lightpink", "lavender", "lightgreen"))
```



d. Subset species and show last 6 rows

```
# Subset setosa
setosa <- iris[iris$Species == "setosa", ]
cat("Last 6 rows of Setosa:\n")
```

```
## Last 6 rows of Setosa:
```

```
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
```

```
# Subset versicolor
versicolor <- iris[iris$Species == "versicolor", ]
cat("\nLast 6 rows of Versicolor:\n")
```

```
##
## Last 6 rows of Versicolor:
```

```
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
```

```
# Subset virginica
virginica <- iris[iris$Species == "virginica", ]
cat("\nLast 6 rows of Virginica:\n")
```

```
##
## Last 6 rows of Virginica:
```

```
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
```

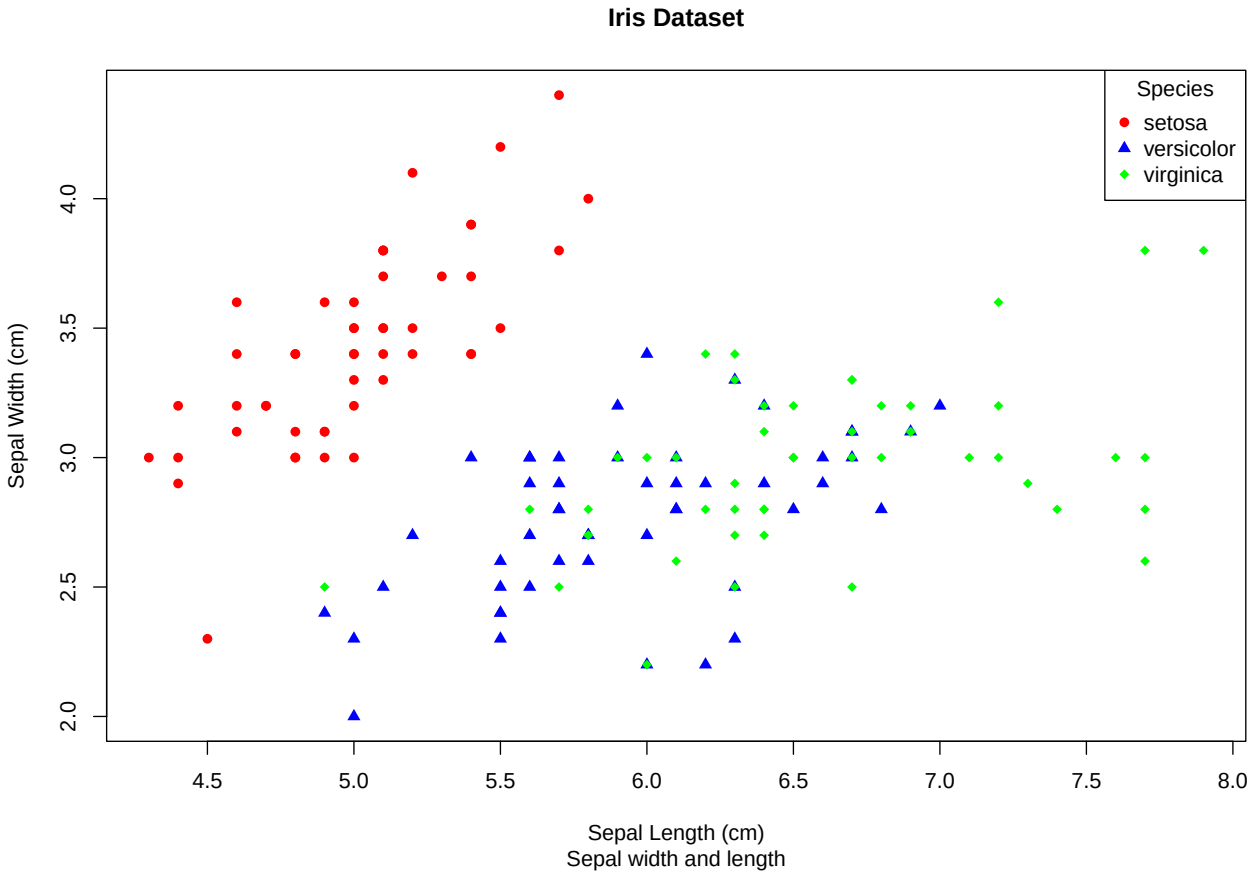
e. Create scatterplot of sepal measurements

```
# Convert Species to factor (already a factor, but being explicit)
iris$Species <- as.factor(iris$Species)

# Create color vector
colors <- c("red", "blue", "green")
pch_symbols <- c(16, 17, 18)

# Create scatterplot
plot(iris$Sepal.Length, iris$Sepal.Width,
     main = "Iris Dataset",
     sub = "Sepal width and length",
     xlab = "Sepal Length (cm)",
     ylab = "Sepal Width (cm)",
     pch = pch_symbols[as.numeric(iris$Species)],
     col = colors[as.numeric(iris$Species)])

# Add legend
legend("topright",
       legend = levels(iris$Species),
       col = colors,
       pch = pch_symbols,
       title = "Species")
```



f. Interpretation of the result

Interpretation:

The scatterplot reveals distinct patterns among the three iris species:

1. **Setosa (red circles):** Forms a clearly separated cluster with wider sepals (3.0-4.4 cm) and shorter sepal lengths (4.3-5.8 cm). This species is easily distinguishable from the others.
2. **Versicolor (blue triangles):** Shows intermediate characteristics with moderate sepal widths (2.0-3.4 cm) and lengths (4.9-7.0 cm). There is some overlap with virginica species.
3. **Virginica (green diamonds):** Generally has narrower sepals (2.2-3.8 cm) but longer sepal lengths (4.9-7.9 cm). Some overlap exists with versicolor in the middle range.

The plot demonstrates that sepal measurements alone can effectively distinguish setosa from the other two species, but versicolor and virginica show some overlap, suggesting these measurements may not be sufficient alone to perfectly classify all iris specimens.

Basic Cleaning and Transformation of Objects

7. Alexa File Analysis

Note: Since the actual alexa-file.xlsx is not provided, I'll create a sample dataset that mimics the structure described in the problem.

```
# Create sample Alexa data
alexa_data <- data.frame(
  variation = c("Black  Dot", "Black  Dot", "Black Dot",
                "Black  Plus", "Black Plus",
                "Black  Show", "Black Show",
                "Black  Spot", "Black Spot",
                "White  Dot", "White Dot",
                "White  Plus", "White Plus",
                "White  Show", "White Show",
                "White  Spot", "White Spot"),
  rating = c(5, 4, 5, 4, 5, 3, 4, 5, 4, 5, 4, 3, 5, 4, 5, 3, 4),
  feedback = sample(c("Love it", "Great", "Good", "Excellent"), 17, replace = TRUE)
)

# Save as RData for demonstration
save(alexa_data, file = "alexa_file.RData")
```

a. Rename variants using gsub() function

```
# Load the data
load("alexa_file.RData")

# Show original data snippet
cat("Original data (first 10 rows):\n")
```

```
## Original data (first 10 rows):
```

```
print(head(alexa_data, 10))
```

```
##      variation rating  feedback
## 1   Black  Dot      5      Good
## 2   Black  Dot      4   Love it
## 3    Black Dot      5      Good
## 4  Black  Plus      4      Good
## 5   Black Plus      5     Great
## 6  Black  Show      3     Great
## 7   Black Show      4   Love it
## 8  Black  Spot      5   Love it
## 9   Black Spot      4 Excellent
## 10  White  Dot      5   Love it
```

```
# Clean up extra whitespaces using gsub
alexa_data$variation <- gsub("Black  Dot", "Black Dot", alexa_data$variation)
alexa_data$variation <- gsub("Black  Plus", "Black Plus", alexa_data$variation)
alexa_data$variation <- gsub("Black  Show", "Black Show", alexa_data$variation)
alexa_data$variation <- gsub("Black  Spot", "Black Spot", alexa_data$variation)

alexa_data$variation <- gsub("White  Dot", "White Dot", alexa_data$variation)
alexa_data$variation <- gsub("White  Plus", "White Plus", alexa_data$variation)
```

```r
alexa_data$variation <- gsub("White  Show", "White Show", alexa_data$variation)
alexa_data$variation <- gsub("White  Spot", "White Spot", alexa_data$variation)

# Show cleaned data snippet
cat("\n\nCleaned data (first 10 rows):\n")
```

```
##
##
## Cleaned data (first 10 rows):
```

```r
print(head(alexa_data, 10))
```

```
##     variation rating  feedback
## 1   Black Dot      5      Good
## 2   Black Dot      4   Love it
## 3   Black Dot      5      Good
## 4  Black Plus      4      Good
## 5  Black Plus      5     Great
## 6  Black Show      3     Great
## 7  Black Show      4   Love it
## 8  Black Spot      5   Love it
## 9  Black Spot      4 Excellent
## 10  White Dot      5   Love it
```

b. Get total number of each variation

```r
# Install and load dplyr if not already installed
if (!require(dplyr)) {
  install.packages("dplyr")
  library(dplyr)
}
```

```
## Loading required package: 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
```

```r
# Count variations
variations <- alexa_data %>%
  count(variation)

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

cat("Total number of each variation:\n")
```

```
## Total number of each variation:
```

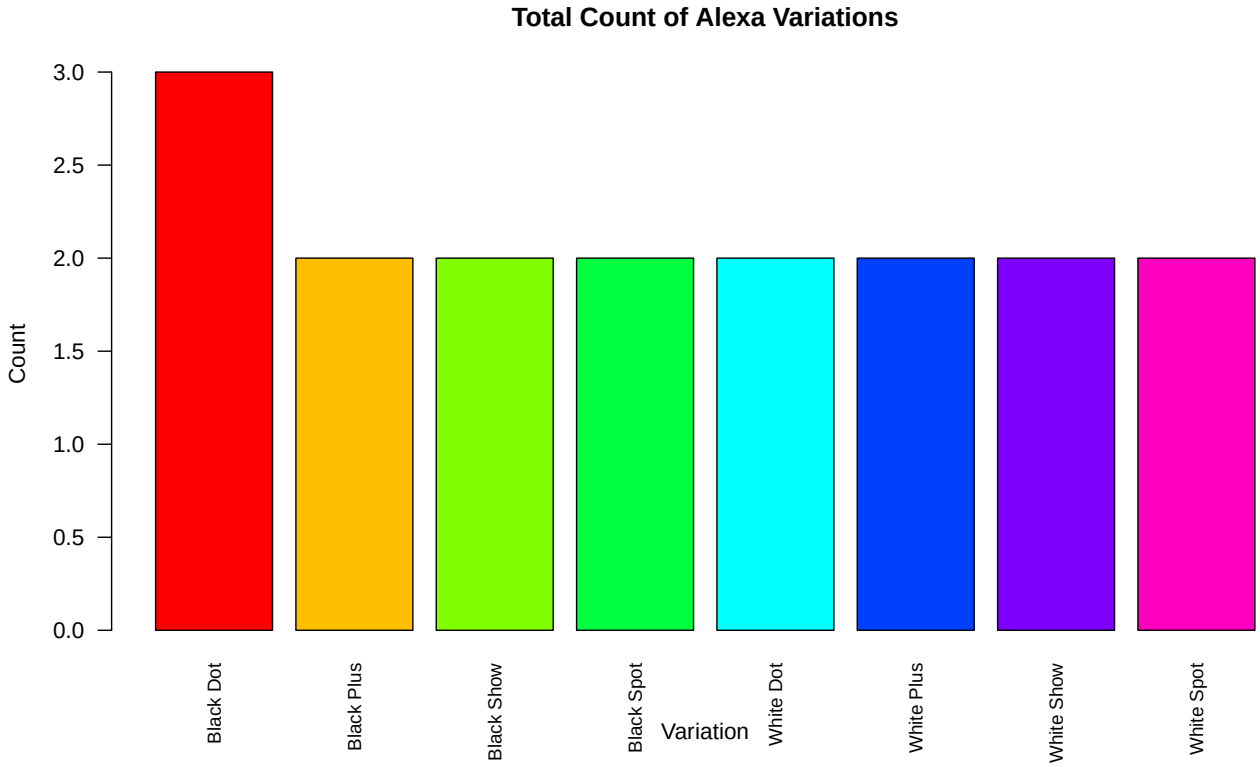
```r
print(variations)
```

```
##    variation n
## 1  Black Dot 3
## 2 Black Plus 2
## 3 Black Show 2
## 4 Black Spot 2
## 5  White Dot 2
## 6 White Plus 2
## 7 White Show 2
## 8 White Spot 2
```

c. Create barplot for variations

```r
# Load variations data
load("variations.RData")

# Create barplot
barplot(variations$n,
        names.arg = variations$variation,
        main = "Total Count of Alexa Variations",
        xlab = "Variation",
        ylab = "Count",
        col = rainbow(nrow(variations)),
        las = 2,
        cex.names = 0.8)
```



d. Create side-by-side barplot for black and white variations

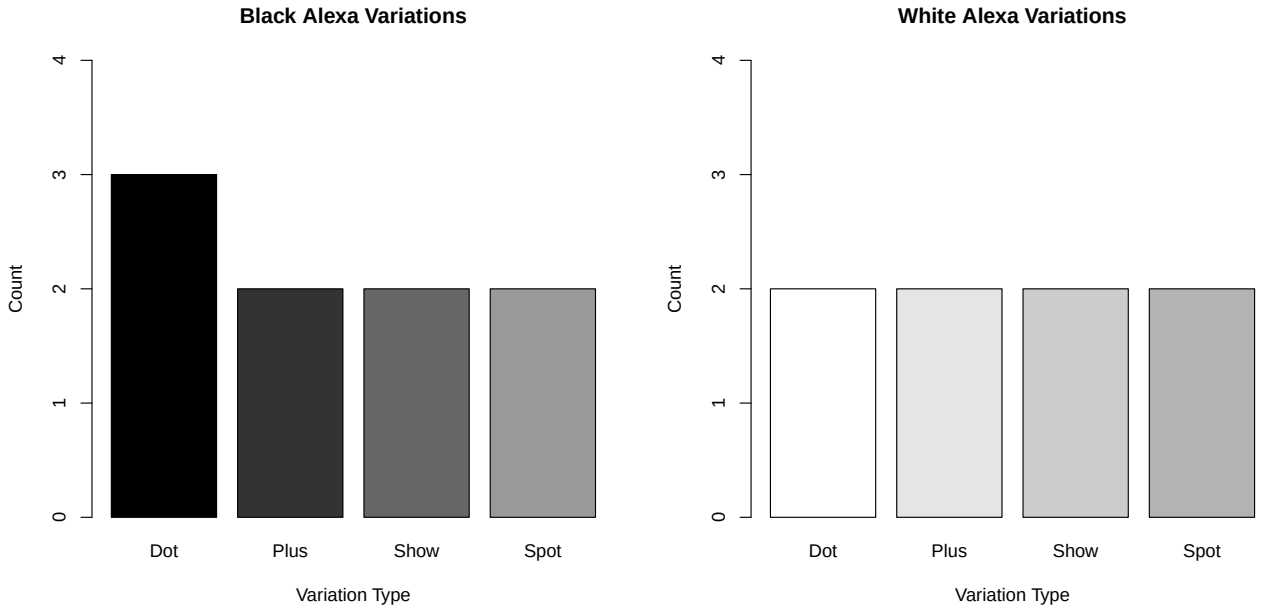
```r
# Separate black and white variations
black_variations <- variations[grepl("Black", variations$variation), ]
```

```r
white_variations <- variations[grepl("White", variations$variation), ]

# Create side-by-side plots
par(mfrow = c(1, 2))

# Black variations barplot
barplot(black_variations$n,
        names.arg = gsub("Black ", "", black_variations$variation),
        main = "Black Alexa Variations",
        xlab = "Variation Type",
        ylab = "Count",
        col = c("black", "gray20", "gray40", "gray60"),
        ylim = c(0, max(variations$n) + 1))

# White variations barplot
barplot(white_variations$n,
        names.arg = gsub("White ", "", white_variations$variation),
        main = "White Alexa Variations",
        xlab = "Variation Type",
        ylab = "Count",
        col = c("white", "gray90", "gray80", "gray70"),
        border = "black",
        ylim = c(0, max(variations$n) + 1))
```




```r
# Reset plotting parameters
par(mfrow = c(1, 1))
```