Lab Record

R V Abhishek

2025-08-13

Lab Program - 1: Program to check what type of Triangle given 3 sides, and calculate its area

This R program validates the sides of the triangle (taken as input from the user) and then if valid, calculates the area of the triangle using Heron's formula and checks what type of triangle it is

```
# Validating the triangle
is_valid_triangle <- function(a, b, c) {
  return ((a + b > c) & (b + c > a) & (a + c > b ))
}
```

```
# Function to check the type of triangle
triangle_type <- function(a , b , c) {
  if (a == b && b == c) {
    return(" Equilateral ")
  } else if ( a == b || b == c || a == c) {
    return(" Isosceles ")
  } else {
    return("Scalene")
  }
}
```

```
# Calculating Area using Heron's Formula
triangle_area <- function(a , b , c) {
  s <- (a + b + c) / 2 # Semi - perimeter
  # Heron 's formula
  area <- sqrt (s * (s - a) * (s - b) * (s - c))
  return (area)
}
```

```
# Validating inputs
validate_input <- function(x) {
  if (!is.numeric(x) || x <= 0) {
    stop("Error : Input must be a positive number.")
  }
  return(TRUE)
}
```

```r
## Main Code Block

# 1. Defining 3 variables representing the 3 sides of the triangle
cat("Enter the lengths of the sides of the triangle :\n")
```

```
Enter the lengths of the sides of the triangle :
```

```r
a <- as.numeric(readline(prompt = "Side A: "))
```

```
Side A:
```

```r
b <- as.numeric(readline(prompt = "Side B: "))
```

```
Side B:
```

```r
c <- as.numeric(readline(prompt = "Side C: "))
```

```
Side C:
```

```r
# 2.  Input Validation and implementation of all the functions.
# Input validation}
tryCatch ({
  validate_input(a)
  validate_input(b)
  validate_input(c)

  # Check if the inputs form a valid triangle
  if (!is_valid_triangle(a , b , c)) {
    stop("Error : The given sides do not form a valid triangle.")
  }

  # Determine the type of triangle
  type_of_triangle <- triangle_type(a , b , c)
  cat("The triangle is:", type_of_triangle, "\n")
  # Calculate the area of the triangle
  area_of_triangle <- triangle_area(a, b, c)
  cat("The area of the triangle is:", area_of_triangle, "\n")

}, error = function(e){
  cat(e$message, "\n")
})
```

```
missing value where TRUE/FALSE needed
```

Sample Output

```
Enter the lengths of the sides of the triangle:
Side a: 5
Side b: 5
```

```
Side c: 8

The triangle is: Isosceles
The area of the triangle is: 12

Enter the lengths of the sides of the triangle:**
Side a: 1
Side b: 2
Side c: 8

Error: The given sides do not form a valid triangle.
```

Lab Program - 2: Creating and Manipulating Data Structures

This program evaluates the student's understanding of different data structures (vectors, matrices, lists, and data frames) in R and how to manipulate them.

```r
# 1. Create a vector of random numbers and apply operations such as sorting and searching

set.seed(42) # For reproducibility
random_vector <- runif(20, min = 1, max = 100) # Vector of 20 random numbers between 1 and 100
cat("Original random vector:\n")
```

```
Original random vector:
```

```r
print(random_vector)
```

```
 [1] 91.56580 93.77047 29.32781 83.21431 64.53281 52.39050 73.92224 14.33199
 [9] 66.04224 70.80141 46.31644 72.19211 93.53255 26.28745 46.76699 94.06144
[17] 97.84442 12.63125 48.02471 56.47294
```

```r
# Sort the vector
sorted_vector <- sort(random_vector)
cat("Sorted vector:\n")
```

```
Sorted vector:
```

```r
print(sorted_vector)
```

```
 [1] 12.63125 14.33199 26.28745 29.32781 46.31644 46.76699 48.02471 52.39050
 [9] 56.47294 64.53281 66.04224 70.80141 72.19211 73.92224 83.21431 91.56580
[17] 93.53255 93.77047 94.06144 97.84442
```

```r
# Search for a specific value (check if a number is present)
search_value <- 50
is_value_present <- any(random_vector == search_value)
cat("Is", search_value, "present in the vector?", is_value_present, "\n")
```

```
Is 50 present in the vector? FALSE
```

```r
# Find values in the vector greater than 60
values_greater_than_60 <- random_vector[random_vector > 60]
cat("Values greater than 60:\n")
```

```
Values greater than 60:
```

```r
print(values_greater_than_60)
```

```
 [1] 91.56580 93.77047 83.21431 64.53281 73.92224 66.04224 70.80141 72.19211
 [9] 93.53255 94.06144 97.84442
```

```r
# 2. Convert the vector into a matrix and perform matrix multiplication

# Create a 4x5 matrix from the vector
matrix_from_vector <- matrix(random_vector, nrow = 4, ncol = 5)
cat("Matrix from vector:\n")
```

```
Matrix from vector:
```

```r
print(matrix_from_vector)
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 91.56580 64.53281 66.04224 93.53255 97.84442
[2,] 93.77047 52.39050 70.80141 26.28745 12.63125
[3,] 29.32781 73.92224 46.31644 46.76699 48.02471
[4,] 83.21431 14.33199 72.19211 94.06144 56.47294
```

```r
# Perform matrix multiplication (matrix with its transpose)
matrix_transpose <- t(matrix_from_vector)
matrix_multiplication_result <- matrix_from_vector %*% matrix_transpose
cat("Matrix multiplication result:\n")
```

```
Matrix multiplication result:
```

```r
print(matrix_multiplication_result)
```

```
         [,1]     [,2]     [,3]     [,4]
[1,] 35232.22 20337.59 19587.86 27635.57
[2,] 20337.59 17401.08 11738.17 16851.17
[3,] 19587.86 11738.17 12963.36 13954.70
[4,] 27635.57 16851.17 13954.70 24378.48
```

```r
# Element-wise matrix multiplication (Hadamard product)
elementwise_multiplication_result <- matrix_from_vector * matrix_from_vector
cat("Element-wise matrix multiplication result:\n")
```

```
Element-wise matrix multiplication result:
```

```r
print(elementwise_multiplication_result)
```

```
          [,1]     [,2]     [,3]      [,4]      [,5]
[1,] 8384.2954 4164.483 4361.577 8748.3384 9573.5298
[2,] 8792.9003 2744.764 5012.840  691.0302  159.5484
[3,]  860.1207 5464.498 2145.212 2187.1513 2306.3729
[4,] 6924.6222  205.406 5211.701 8847.5541 3189.1932
```

```r
# 3. Create a list containing different types of elements and perform subsetting

my_list <- list(
  numbers = random_vector,
  characters = c("A", "B", "C", "D"),
  logical_values = c(TRUE, FALSE, TRUE),
  matrix = matrix_from_vector
)
cat("List:\n")
```

```
List:
```

```r
print(my_list)
```

```
$numbers
 [1] 91.56580 93.77047 29.32781 83.21431 64.53281 52.39050 73.92224 14.33199
 [9] 66.04224 70.80141 46.31644 72.19211 93.53255 26.28745 46.76699 94.06144
[17] 97.84442 12.63125 48.02471 56.47294

$characters
[1] "A" "B" "C" "D"

$logical_values
[1]  TRUE FALSE  TRUE

$matrix
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 91.56580 64.53281 66.04224 93.53255 97.84442
[2,] 93.77047 52.39050 70.80141 26.28745 12.63125
[3,] 29.32781 73.92224 46.31644 46.76699 48.02471
[4,] 83.21431 14.33199 72.19211 94.06144 56.47294
```

```r
# Subsetting the list (extracting numeric and logical parts)
subset_numeric <- my_list$numbers
cat("Subset (numeric part of the list):\n", subset_numeric, "\n")
```

```
Subset (numeric part of the list):
 91.5658 93.77047 29.32781 83.21431 64.53281 52.3905 73.92224 14.33199 66.04224 70.80141 46.31644 72.192
```

```r
subset_logical <- my_list$logical_values
cat("Subset (logical part of the list):\n", subset_logical, "\n")
```

```
Subset (logical part of the list):
 TRUE FALSE TRUE
```

```r
# Modify elements in the list (replace the second character with "Z")
my_list$characters[2] <- "Z"
cat("Modified list of characters:\n", my_list$characters, "\n")
```

```
Modified list of characters:
 A Z C D
```

```r
# Apply a function to the numeric part of the list (e.g., calculate the square of the numbers)
squared_numbers <- my_list$numbers ^ 2
cat("Squared numbers:\n", squared_numbers, "\n")
```

```
Squared numbers:
 8384.295 8792.9 860.1207 6924.622 4164.483 2744.764 5464.498 205.406 4361.577 5012.84 2145.212 5211.70
```

```r
# 4. Create a data frame and perform operations such as filtering, summarizing, and handling missing va
# Create a data frame
df <- data.frame(
  ID = 1:20,
  Age = sample(18:65, 20, replace = TRUE),
  Score = runif(20, min = 50, max = 100),
  Passed = sample(c(TRUE, FALSE), 20, replace = TRUE)
)
cat("Data frame:\n")
```

```
Data frame:
```

```r
print(df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
2   2  20 51.87155  FALSE
3   3  58 98.67700  FALSE
4   4  42 71.58756  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62 80.94191   TRUE
10 10  22 66.67136  FALSE
11 11  37 67.33741   TRUE
12 12  51 69.92427  FALSE
13 13  45 89.23464  FALSE
14 14  57 51.94682  FALSE
15 15  20 87.43977  FALSE
16 16  50 83.86384   TRUE
17 17  59 58.56322   TRUE
18 18  41 63.05440   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Filter the data frame (rows where Age > 30 and Score > 70)
filtered_df <- subset(df, Age > 30 & Score > 70)
cat("Filtered data frame (Age > 30 and Score > 70):\n")
```

```
Filtered data frame (Age > 30 and Score > 70):
```

```r
print(filtered_df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
3   3  58 98.67700  FALSE
4   4  42 71.58756  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62 80.94191   TRUE
13 13  45 89.23464  FALSE
16 16  50 83.86384   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Calculate mean, sum, and variance of numerical columns (Age and Score)
mean_age <- mean(df$Age)
sum_age <- sum(df$Age)
var_age <- var(df$Age)

mean_score <- mean(df$Score)
sum_score <- sum(df$Score)
var_score <- var(df$Score)

cat("Summary statistics for Age column:\n")
```

```
Summary statistics for Age column:
```

```r
cat("Mean Age:", mean_age, "\n")
```

```
Mean Age: 46.7
```

```r
cat("Sum of Age:", sum_age, "\n")
```

```
Sum of Age: 934
```

```r
cat("Variance of Age:", var_age, "\n")
```

```
Variance of Age: 179.6947
```

```r
cat("Summary statistics for Score column:\n")
```

```
Summary statistics for Score column:
```

```r
cat("Mean Score:", mean_score, "\n")
```

```
Mean Score: 77.26086
```

```r
cat("Sum of Score:", sum_score, "\n")
```

```
Sum of Score: 1545.217
```

```r
cat("Variance of Score:", var_score, "\n")
```

```
Variance of Score: 219.2162
```

```r
# 5. Handling missing values in the data frame

# Introduce some NA values in the Score column
df$Score[sample(1:20, 5)] <- NA
cat("Data frame with missing values:\n")
```

```
Data frame with missing values:
```

```r
print(df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
2   2  20 51.87155  FALSE
3   3  58 98.67700  FALSE
4   4  42       NA  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62       NA   TRUE
10 10  22       NA  FALSE
11 11  37 67.33741   TRUE
12 12  51       NA  FALSE
13 13  45       NA  FALSE
14 14  57 51.94682  FALSE
15 15  20 87.43977  FALSE
16 16  50 83.86384   TRUE
17 17  59 58.56322   TRUE
18 18  41 63.05440   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Replace NA values with the mean of the Score column
df$Score[is.na(df$Score)] <- mean(df$Score, na.rm = TRUE)
cat("Data frame after imputation of missing values:\n")
```

```
Data frame after imputation of missing values:
```

```r
print(df)
```

```
   ID Age    Score Passed
1   1  64 71.78858  FALSE
2   2  20 51.87155  FALSE
3   3  58 98.67700  FALSE
4   4  42 77.79050  FALSE
5   5  44 97.87883  FALSE
6   6  53 94.38775  FALSE
7   7  54 81.99894   TRUE
8   8  48 98.54833  FALSE
9   9  62 77.79050   TRUE
10 10  22 77.79050  FALSE
11 11  37 67.33741   TRUE
12 12  51 77.79050  FALSE
13 13  45 77.79050  FALSE
14 14  57 51.94682  FALSE
15 15  20 87.43977  FALSE
16 16  50 83.86384   TRUE
17 17  59 58.56322   TRUE
18 18  41 63.05440   TRUE
19 19  47 75.72065   TRUE
20 20  60 83.78036  FALSE
```

```r
# Grouping the data by Passed status and calculating group-wise statistics
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
```

```r
grouped_stats <- df %>%
  group_by(Passed) %>%
  summarise(
    mean_score = mean(Score, na.rm = TRUE),
    mean_age = mean(Age)
  )
cat("Grouped statistics by Passed status:\n")
```

```
Grouped statistics by Passed status:
```

```r
print(grouped_stats)
```

```
# A tibble: 2 x 3
  Passed mean_score mean_age
  <lgl>       <dbl>    <dbl>
1 FALSE        80.6     44.9
2 TRUE         72.6     50
```

Lab Program - 3: Basic Statistical Operations on Open-Source Datasets

Objective: This program emphasizes the application of statistical concepts on real-world datasets and visualization of the data.

```r
# Load necessary
library(dplyr)  # For data manipulation
library(ggplot2)  # For visualization
library(moments)  # For skewness and kurtosis
library(palmerpenguins) # For Palmer Penguins dataset
```

```
Attaching package: 'palmerpenguins'

The following objects are masked from 'package:datasets':

    penguins, penguins_raw
```

```r
data(iris)  # Load Iris dataset

data(penguins)  # Load Palmer Penguins
```

```r
# Function to calculate mode
calc_mode <- function(x) {
  return (as.numeric (names (sort (table (x), decreasing = TRUE)) [1] ))
}
```

```r
# Perform Statistical Analysis on Iris Dataset
print("----- Iris Dataset Analysis -----")
```

```
[1] "----- Iris Dataset Analysis -----"
```

```r
# Mean
iris_mean <- sapply (iris[, 1:4], mean, na.rm = TRUE )
print(paste("Mean of Iris dataset : ", iris_mean))
```

```
[1] "Mean of Iris dataset :  5.84333333333333"
[2] "Mean of Iris dataset :  3.05733333333333"
[3] "Mean of Iris dataset :  3.758"
[4] "Mean of Iris dataset :  1.19933333333333"
```

```r
#Median
iris_median <- sapply(iris[, 1:4], median, na.rm = TRUE )
print(paste("Median of Iris dataset : ", iris_median))
```

```
[1] "Median of Iris dataset :  5.8"  "Median of Iris dataset :  3"
[3] "Median of Iris dataset :  4.35" "Median of Iris dataset :  1.3"
```

```r
#Mode
iris_mode <- sapply(iris[, 1:4], calc_mode )
print(paste("Mode of Iris dataset : ", iris_median))
```

```
[1] "Mode of Iris dataset :  5.8"  "Mode of Iris dataset :  3"
[3] "Mode of Iris dataset :  4.35" "Mode of Iris dataset :  1.3"
```

```r
#Variance
iris_variance <- sapply(iris[, 1:4], var, na.rm = TRUE )
print(paste("Variance of Iris dataset : ", iris_variance))
```

```
[1] "Variance of Iris dataset :  0.685693512304251"
[2] "Variance of Iris dataset :  0.189979418344519"
[3] "Variance of Iris dataset :  3.11627785234899"
[4] "Variance of Iris dataset :  0.581006263982103"
```

```r
#Standard Deviation
iris_sd <- sapply(iris[, 1:4], sd, na.rm = TRUE )
print(paste("Standard Deviation of Iris dataset : ", iris_sd))
```

```
[1] "Standard Deviation of Iris dataset :  0.828066127977863"
[2] "Standard Deviation of Iris dataset :  0.435866284936698"
[3] "Standard Deviation of Iris dataset :  1.76529823325947"
[4] "Standard Deviation of Iris dataset :  0.762237668960347"
```

```r
#Skewness
iris_skewness <- sapply(iris[, 1:4], skewness, na.rm = TRUE )
print(paste("Skewness of Iris dataset : ", iris_skewness))
```

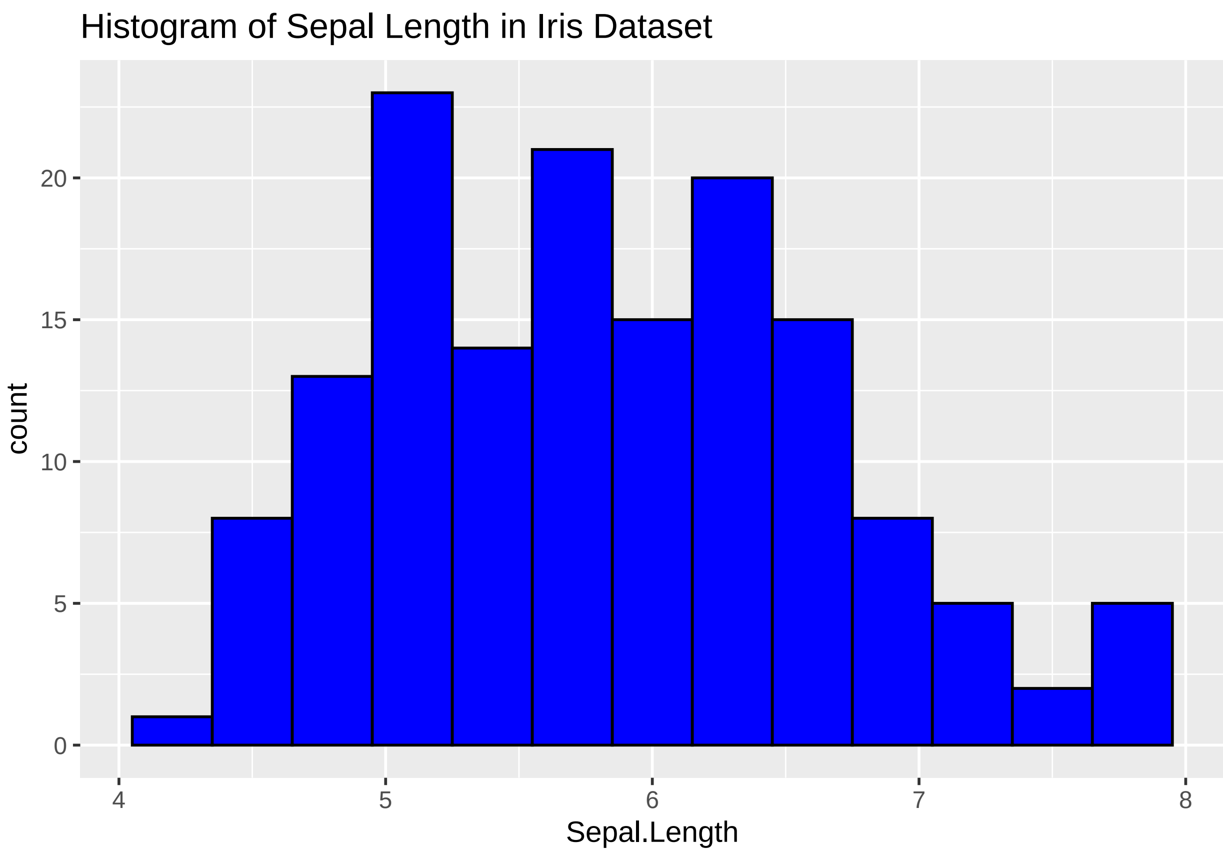
```
[1] "Skewness of Iris dataset :  0.311753058502296"
[2] "Skewness of Iris dataset :  0.315767106338938"
[3] "Skewness of Iris dataset :  -0.272127666456721"
[4] "Skewness of Iris dataset :  -0.101934206565599"
```

```r
# Hypothesis Testing (t-test) between Sepal.Length of Setosa and Versicolor
setosa <- subset(iris, Species == "setosa")$Sepal.Length
versicolor <- subset(iris, Species == "versicolor")$Sepal.Length
t_test <- t.test(setosa, versicolor)
print(t_test)
```

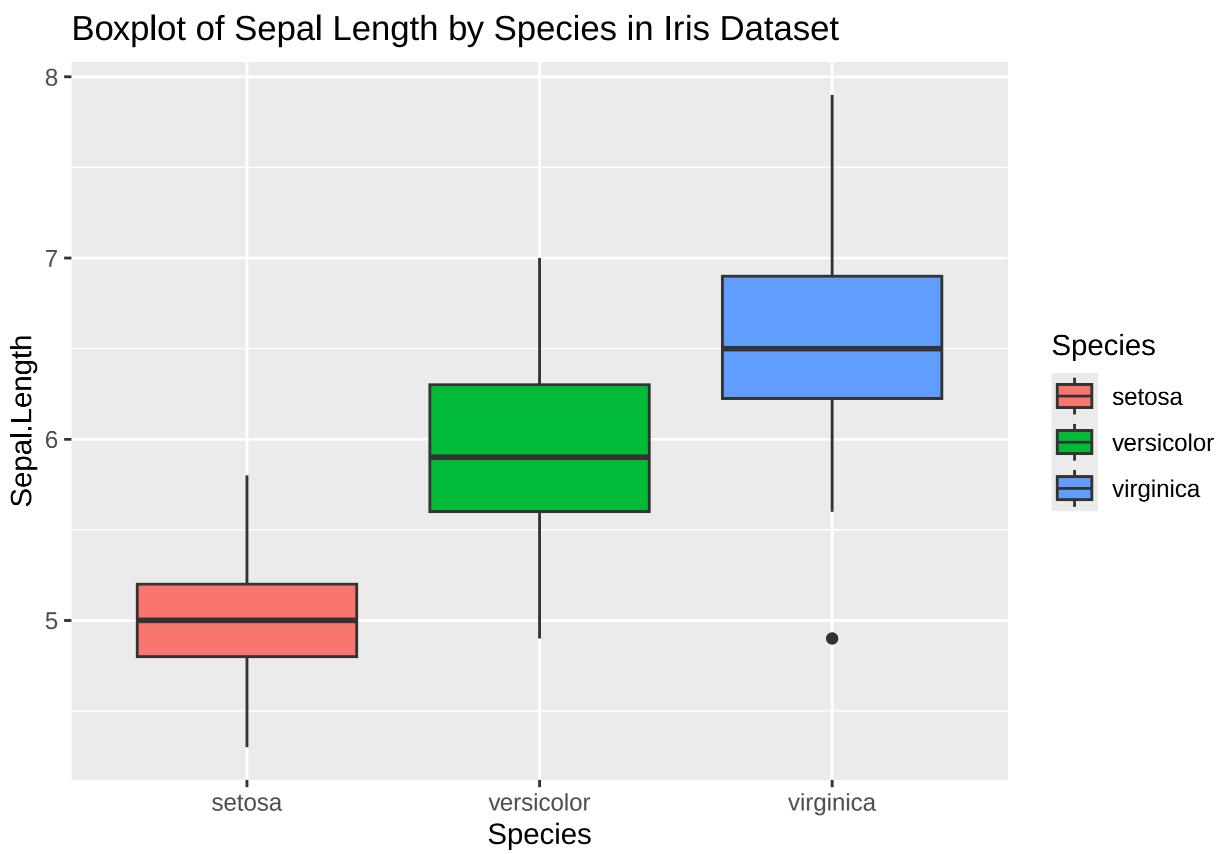
```
    Welch Two Sample t-test
```

```
data:  setosa and versicolor
t = -10.521, df = 86.538, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.1057074 -0.7542926
sample estimates:
mean of x mean of y
    5.006     5.936
```

```
# Visualization of Iris Dataset
# Histogram for Sepal.Length
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram(binwidth = 0.3, fill = "blue", color = "black") +
  ggtitle("Histogram of Sepal Length in Iris Dataset")
```




```
# Boxplot for Sepal.Length across Species
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  ggtitle("Boxplot of Sepal Length by Species in Iris Dataset")
```




```
print("----- Palmer Penguins Dataset Analysis -----")
```

```
[1] "----- Palmer Penguins Dataset Analysis -----"
```

```
# Remove rows with missing values
penguins_clean <- na.omit(penguins)

# Mean
penguins_mean <- sapply(penguins_clean[, 3:6], mean, na.rm = TRUE)
print(paste("Mean of Palmer Penguins dataset:", penguins_mean))
```

```
[1] "Mean of Palmer Penguins dataset: 43.9927927927928"
[2] "Mean of Palmer Penguins dataset: 17.1648648648649"
[3] "Mean of Palmer Penguins dataset: 200.966966966967"
[4] "Mean of Palmer Penguins dataset: 4207.05705705706"
```

```
# Median
penguins_median <- sapply(penguins_clean[, 3:6], median, na.rm = TRUE)
print(paste("Median of Palmer Penguins dataset:", penguins_median))
```

```
[1] "Median of Palmer Penguins dataset: 44.5"
[2] "Median of Palmer Penguins dataset: 17.3"
[3] "Median of Palmer Penguins dataset: 197"
[4] "Median of Palmer Penguins dataset: 4050"
```

```r
# Mode
penguins_mode <- sapply(penguins_clean[, 3:6], calc_mode)
print(paste("Mode of Palmer Penguins dataset:", penguins_mode))
```

```
[1] "Mode of Palmer Penguins dataset: 41.1"
[2] "Mode of Palmer Penguins dataset: 17"
[3] "Mode of Palmer Penguins dataset: 190"
[4] "Mode of Palmer Penguins dataset: 3800"
```

```r
# Variance
penguins_variance <- sapply(penguins_clean[, 3:6], var, na.rm = TRUE)
print(paste("Variance of Palmer Penguins dataset:", penguins_variance))
```

```
[1] "Variance of Palmer Penguins dataset: 29.9063334418756"
[2] "Variance of Palmer Penguins dataset: 3.87788830999674"
[3] "Variance of Palmer Penguins dataset: 196.441676616375"
[4] "Variance of Palmer Penguins dataset: 648372.487698542"
```

```r
# Standard Deviation
penguins_sd <- sapply(penguins_clean[, 3:6], sd, na.rm = TRUE)
print(paste("Standard Deviation of Palmer Penguins dataset:", penguins_sd))
```

```
[1] "Standard Deviation of Palmer Penguins dataset: 5.46866834264756"
[2] "Standard Deviation of Palmer Penguins dataset: 1.9692354633199"
[3] "Standard Deviation of Palmer Penguins dataset: 14.0157652882879"
[4] "Standard Deviation of Palmer Penguins dataset: 805.215801942897"
```

```r
# Skewness
penguins_skewness <- sapply(penguins_clean[, 3:6], skewness, na.rm = TRUE)
print(paste("Skewness of Palmer Penguins dataset:", penguins_skewness))
```

```
[1] "Skewness of Palmer Penguins dataset: 0.0451359779776739"
[2] "Skewness of Palmer Penguins dataset: -0.149044996398334"
[3] "Skewness of Palmer Penguins dataset: 0.358523654622741"
[4] "Skewness of Palmer Penguins dataset: 0.470116171418382"
```

```r
# Kurtosis
penguins_kurtosis <- sapply(penguins_clean[, 3:6], kurtosis, na.rm = TRUE)
print(paste("Kurtosis of Palmer Penguins dataset:", penguins_kurtosis))
```

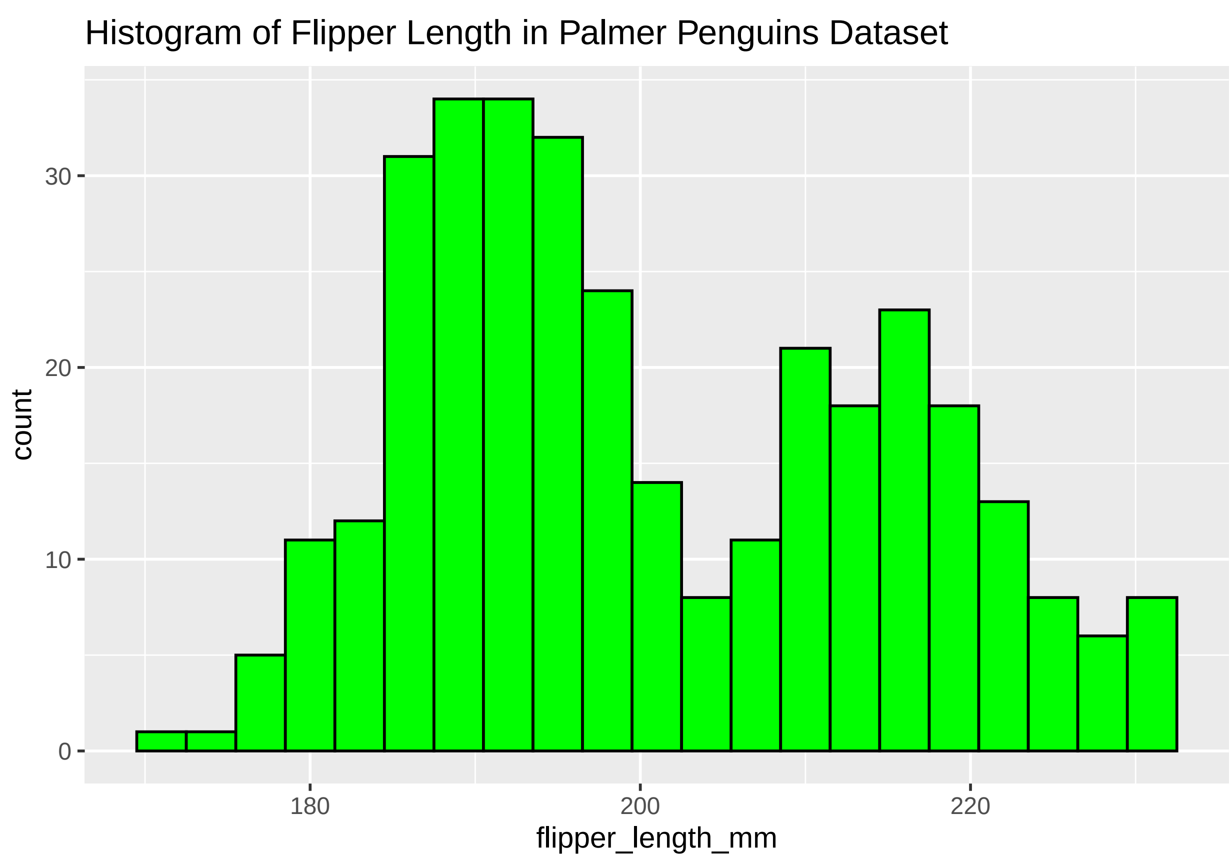
```
[1] "Kurtosis of Palmer Penguins dataset: 2.11182658541194"
[2] "Kurtosis of Palmer Penguins dataset: 2.10341274887238"
[3] "Kurtosis of Palmer Penguins dataset: 2.03516741259049"
[4] "Kurtosis of Palmer Penguins dataset: 2.25951411974012"
```

```r
# Hypothesis Testing (t-test) between flipper_length_mm of Adelie and Gentoo species
adelie <- subset(penguins_clean, species == "Adelie")$flipper_length_mm
gentoo <- subset(penguins_clean, species == "Gentoo")$flipper_length_mm
t_test_penguins <- t.test(adelie, gentoo)
print(t_test_penguins)
```

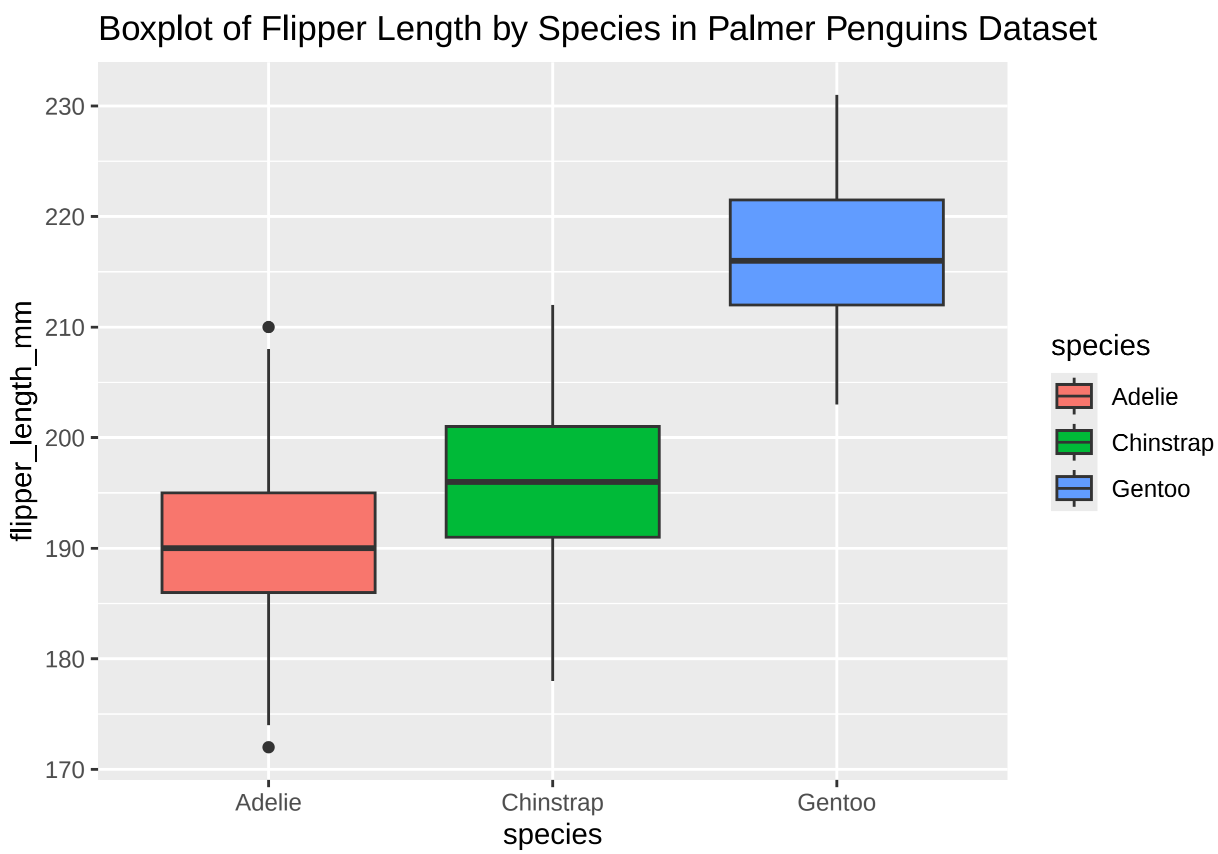
```
	Welch Two Sample t-test

data:  adelie and gentoo
t = -33.506, df = 251.35, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -28.72740 -25.53771
sample estimates:
mean of x mean of y 
 190.1027  217.2353 
```

```r
# Visualization of Palmer Penguins Dataset
# Histogram for flipper_length_mm
ggplot(penguins_clean, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 3, fill = "green", color = "black") +
  ggtitle("Histogram of Flipper Length in Palmer Penguins Dataset")
```




```r
# Boxplot for flipper_length_mm across Species
ggplot(penguins_clean, aes(x = species, y = flipper_length_mm, fill = species)) +
  geom_boxplot() +
  ggtitle("Boxplot of Flipper Length by Species in Palmer Penguins Dataset")
```


Boxplot of Flipper Length by Species in Palmer Penguins Dataset
230
220
210
200
190
180
170
flipper_length_mm
Adelie
Chinstrap
Gentoo
species
species
Adelie
Chinstrap
Gentoo