Contents

Palmer Penguins: Statistical Analysis and Shiny App Development

Introduction

This project aims to apply advanced statistical analysis techniques to the Palmer Penguins dataset. In this R Markdown document, we will explore the dataset, preprocess the data, handle missing values, and develop a predictive model using a neural network. Additionally, a Shiny app will be used to visualize results interactively.

Step 1: Loading and Preprocessing the Data

Load the Dataset

First, we load the Palmer Penguins dataset using the `palmerpenguins` package.

```r
library(palmerpenguins)
data = penguins
```

Handle Missing Values

There are missing values in the dataset that we need to handle before proceeding with further analysis.

- **Numerical Missing Values**: Impute using median values, grouped by species.
- **Categorical Missing Values**: Use K-means clustering to find patterns or replace using the mode, depending on the distribution.

```r
library(dplyr)
# Numerical values: Impute using median by species
data = data %>%
  group_by(species) %>%
  mutate(
    bill_length_mm = ifelse(is.na(bill_length_mm), median(bill_length_mm, na.rm = TRUE), bill_length_mm
    bill_depth_mm = ifelse(is.na(bill_depth_mm), median(bill_depth_mm, na.rm = TRUE), bill_depth_mm),
    flipper_length_mm = ifelse(is.na(flipper_length_mm), median(flipper_length_mm, na.rm = TRUE), flipp
    body_mass_g = ifelse(is.na(body_mass_g), median(body_mass_g, na.rm = TRUE), body_mass_g)
  ) %>%
  ungroup()

# Categorical values: Replace using mode
mode_function = function(x) {
  ux = unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

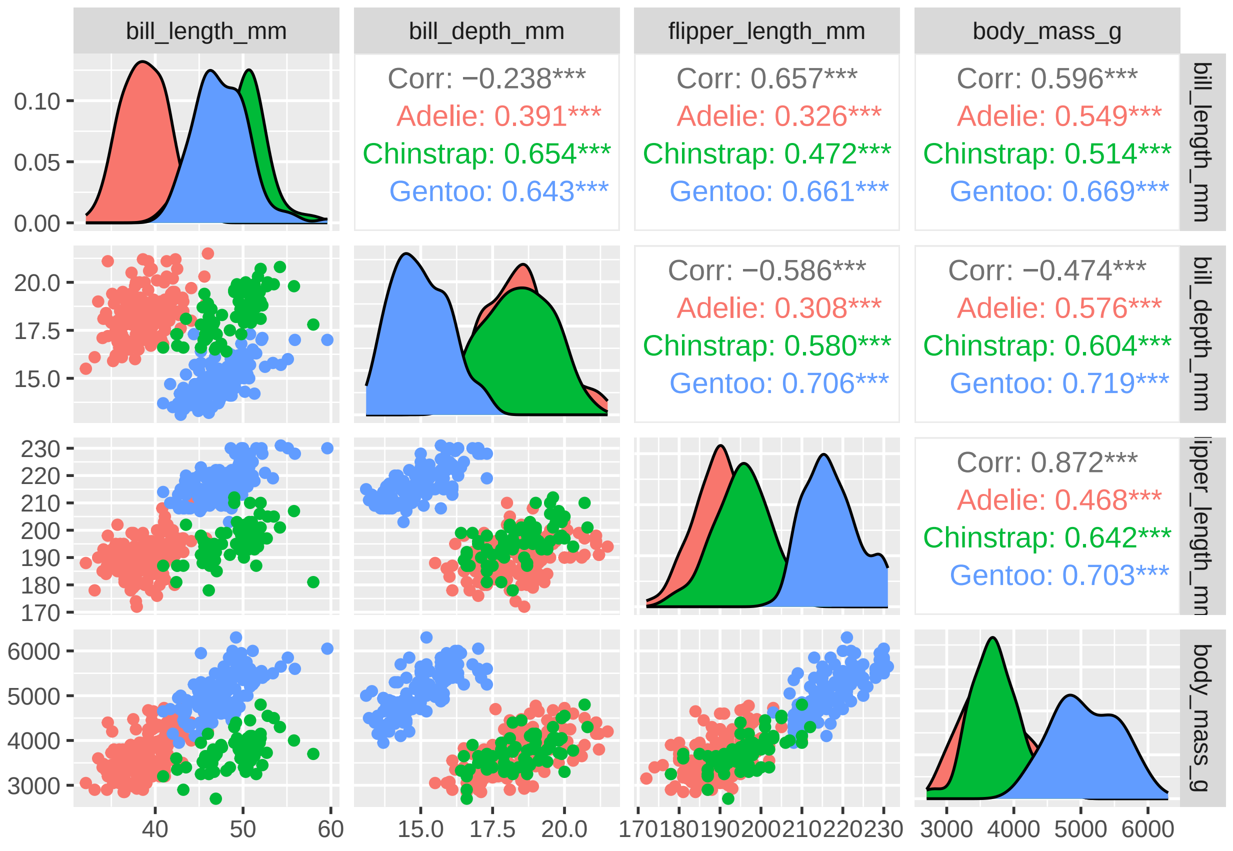
```r
data$island[is.na(data$island)] = mode_function(data$island)
data$sex[is.na(data$sex)] = mode_function(data$sex)
```

Exploratory Data Analysis (EDA)

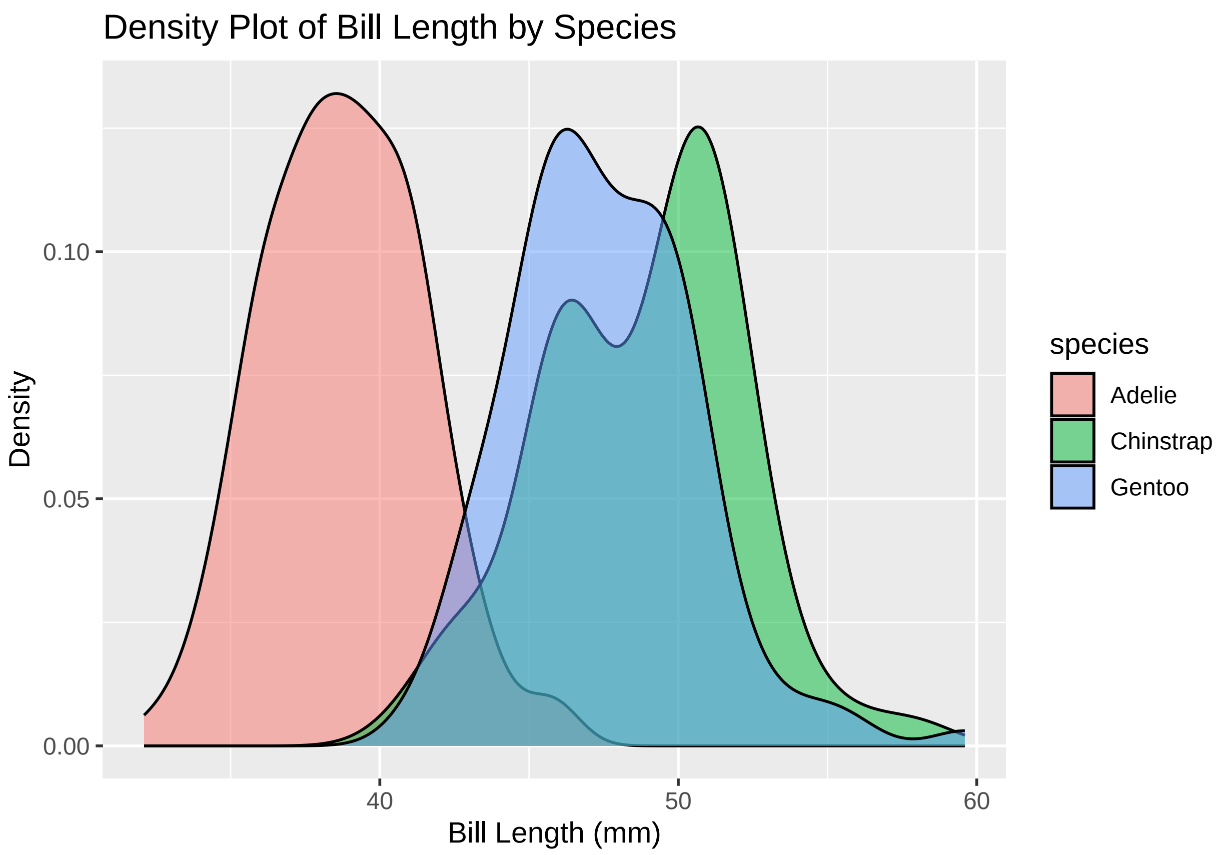
We expand our data exploration to gain a better understanding of relationships between the variables.

- **Pairwise Scatter Plots**: Display pairwise relationships between multiple variables to visualize potential correlations.
- **Density Plots**: Explore the distribution of numerical variables for each species.
- **Correlation Heatmap**: Visualize the correlation between numerical variables.

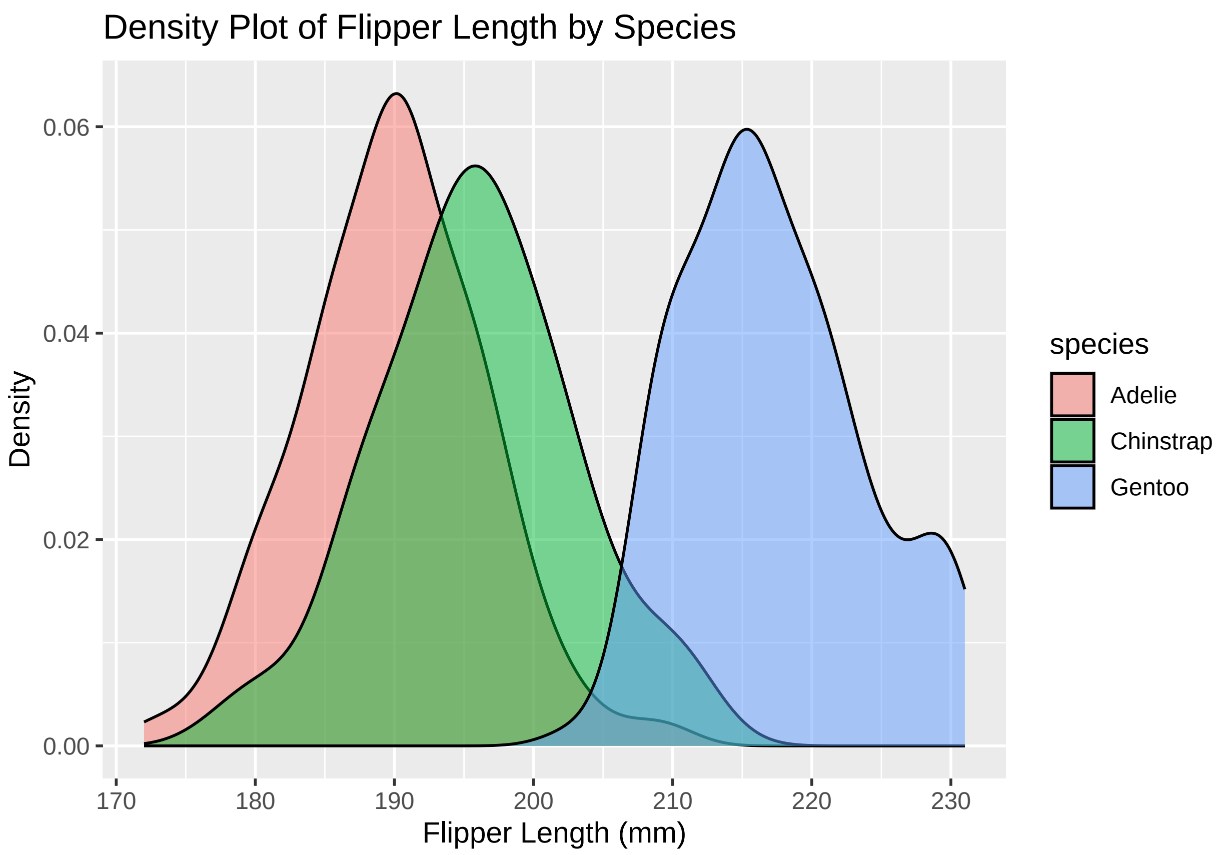
```r
# Pairwise scatter plot matrix for numeric variables
ggpairs(data, columns = c("bill_length_mm", "bill_depth_mm", "flipper_length_mm", "body_mass_g"), aes(co
```



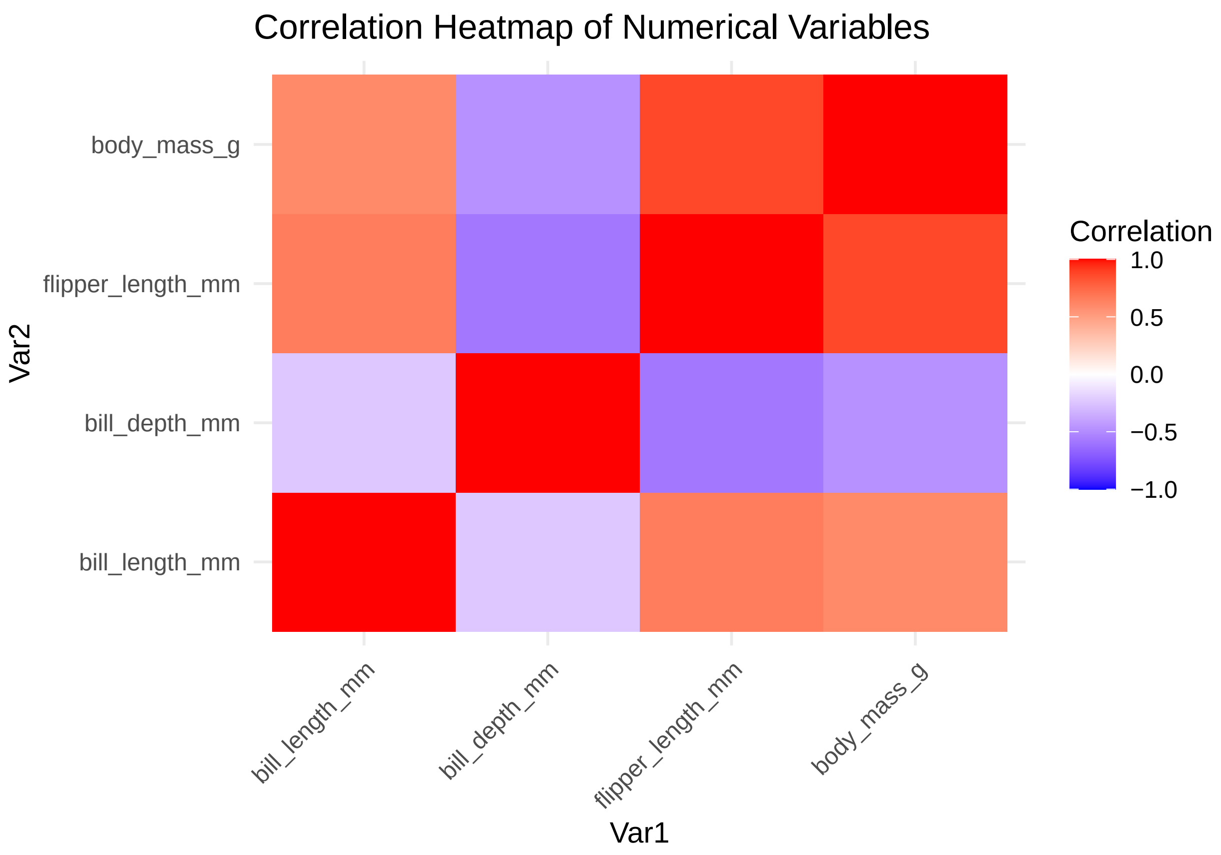
```r
# Density plot: Bill Length by Species
ggplot(data, aes(x = bill_length_mm, fill = species)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Bill Length by Species", x = "Bill Length (mm)", y = "Density")
```



```r
# Density plot: Flipper Length by Species
ggplot(data, aes(x = flipper_length_mm, fill = species)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Flipper Length by Species", x = "Flipper Length (mm)", y = "Density")
```



```
# Correlation Heatmap
numeric_data = data %>% select(bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g) %>% na.om
corr_matrix = cor(numeric_data)

corr_melted = melt(corr_matrix)

ggplot(data = corr_melted, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1, 1), space
                       name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Correlation Heatmap of Numerical Variables")
```



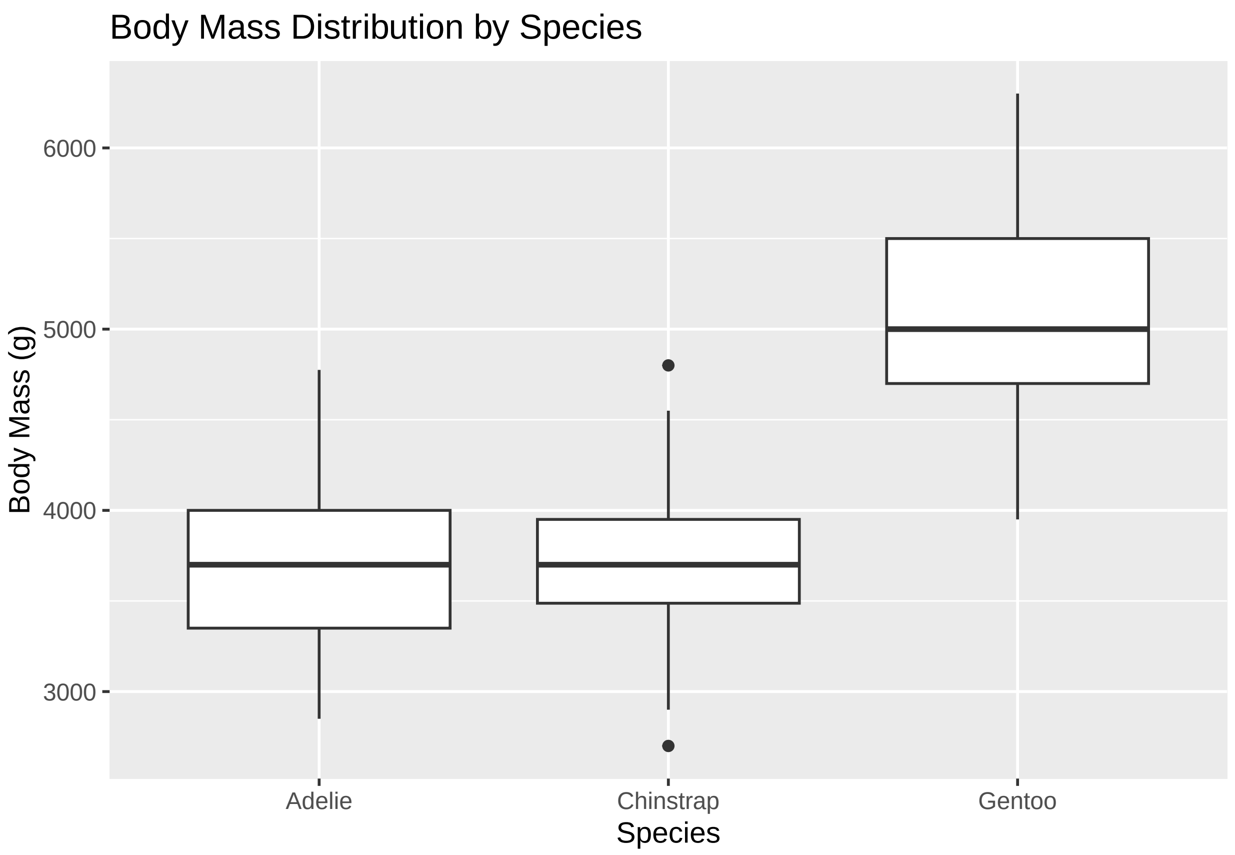
We generate summary statistics and initial visualizations to explore relationships in the data.

- **Box Plots and Scatter Plots**: Used to visually identify patterns and relationships between the variables.

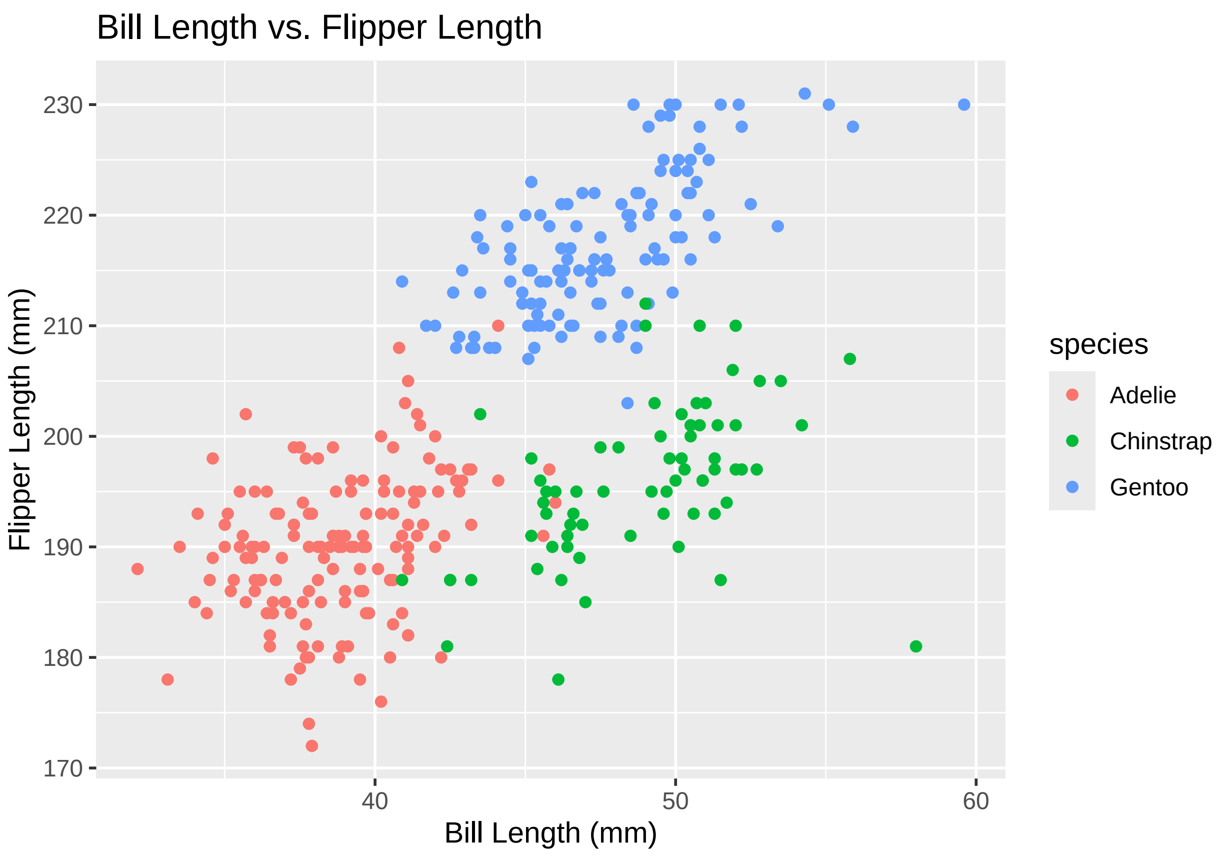
```
library(ggplot2)
# Summary statistics
summary(data)
```

```
##       species          island    bill_length_mm  bill_depth_mm
##  Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
##  Chinstrap: 68   Dream    :124   1st Qu.:39.20   1st Qu.:15.57
##  Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
##                                  Mean   :43.92   Mean   :17.15
##                                  3rd Qu.:48.50   3rd Qu.:18.70
##                                  Max.   :59.60   Max.   :21.50
##  flipper_length_mm  body_mass_g       sex           year
##  Min.   :172.0     Min.   :2700   female:165   Min.   :2007
##  1st Qu.:190.0     1st Qu.:3550   male  :179   1st Qu.:2007
##  Median :197.0     Median :4050                Median :2008
##  Mean   :200.9     Mean   :4203                Mean   :2008
##  3rd Qu.:213.2     3rd Qu.:4756                3rd Qu.:2009
##  Max.   :231.0     Max.   :6300                Max.   :2009
```

```
# Box plot: Body Mass by Species
ggplot(data, aes(x = species, y = body_mass_g)) +
  geom_boxplot() +
  labs(title = "Body Mass Distribution by Species", y = "Body Mass (g)", x = "Species")
```



```r
# Scatter plot: Bill Length vs. Flipper Length
ggplot(data, aes(x = bill_length_mm, y = flipper_length_mm, color = species)) +
  geom_point() +
  labs(title = "Bill Length vs. Flipper Length", x = "Bill Length (mm)", y = "Flipper Length (mm)")
```



Step 2: Shiny App Development

Adding New Columns in Shiny

We create a Shiny app that allows the user to create a new column based on existing data, such as calculating the BMI.

```r
library(shiny)
ui = fluidPage(
  titlePanel("Palmer Penguins Shiny App"),
  sidebarLayout(
    sidebarPanel(
      textInput("new_column", "New Column Name", "BMI"),
      actionButton("add_column", "Add Column")
    ),
    mainPanel(
      tableOutput("data_table")
    )
  )
)

server = function(input, output) {
  penguin_data = reactiveVal(data)

  observeEvent(input$add_column, {
    if (input$new_column == "BMI") {
      updated_data = penguin_data() %>%
        mutate(BMI = body_mass_g / (flipper_length_mm * 0.01)^2)
      penguin_data(updated_data)
    }
  })

  output$data_table = renderTable({
    head(penguin_data())
  })
}

shinyApp(ui = ui, server = server)
```

Step 3: Model Building - Neural Network

We now move on to building a predictive model. For this project, we use a neural network to predict the species of penguins based on various features.

Splitting Data into Train and Test Sets

We first split the dataset into training and testing sets to validate our model.

```
set.seed(123)
trainIndex = createDataPartition(data$species, p = .7, list = FALSE)
train_data = data[trainIndex, ]
test_data = data[-trainIndex, ]
```

Building a Neural Network Model

We use the `nnet` package to create a neural network model that predicts the species of penguins.

- **Input Variables**: Bill length, bill depth, flipper length, and body mass.
- **Output Variable**: Species.

```
library(nnet)
model = nnet(species ~ bill_length_mm + bill_depth_mm + flipper_length_mm + body_mass_g,
              data = train_data, size = 5, linout = FALSE, maxit = 200)
```

```
## # weights:  43
## initial  value 267.007994
## final  value 253.978590
## converged
```

Evaluating Model Performance

We evaluate the model using accuracy metrics on the test dataset. Ensure that both `test_predictions` and `test_data$species` are factors with the same levels to avoid errors.

```
# Ensure species columns are factors with the same levels
levels = levels(train_data$species)
test_data$species = factor(test_data$species, levels = levels)

# Make predictions on test data
test_predictions = predict(model, test_data, type = "class")

test_predictions = factor(test_predictions, levels = levels)
```

```r
# Confusion matrix
confusionMatrix(test_predictions, test_data$species)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Adelie Chinstrap Gentoo
##   Adelie        45        20     37
##   Chinstrap      0         0      0
##   Gentoo         0         0      0
##
## Overall Statistics
##
##                Accuracy : 0.4412
##                  95% CI : (0.3429, 0.5429)
##     No Information Rate : 0.4412
##     P-Value [Acc > NIR] : 0.5381
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Adelie Class: Chinstrap Class: Gentoo
## Sensitivity                 1.0000           0.0000        0.0000
## Specificity                 0.0000           1.0000        1.0000
## Pos Pred Value              0.4412              NaN           NaN
## Neg Pred Value                 NaN           0.8039        0.6373
## Prevalence                  0.4412           0.1961        0.3627
## Detection Rate              0.4412           0.0000        0.0000
## Detection Prevalence        1.0000           0.0000        0.0000
## Balanced Accuracy           0.5000           0.5000        0.5000
```

Step 4: Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that allows us to reduce the number of features while retaining as much variability in the data as possible. We perform PCA to explore the main components that explain the variability in the dataset.

Performing PCA

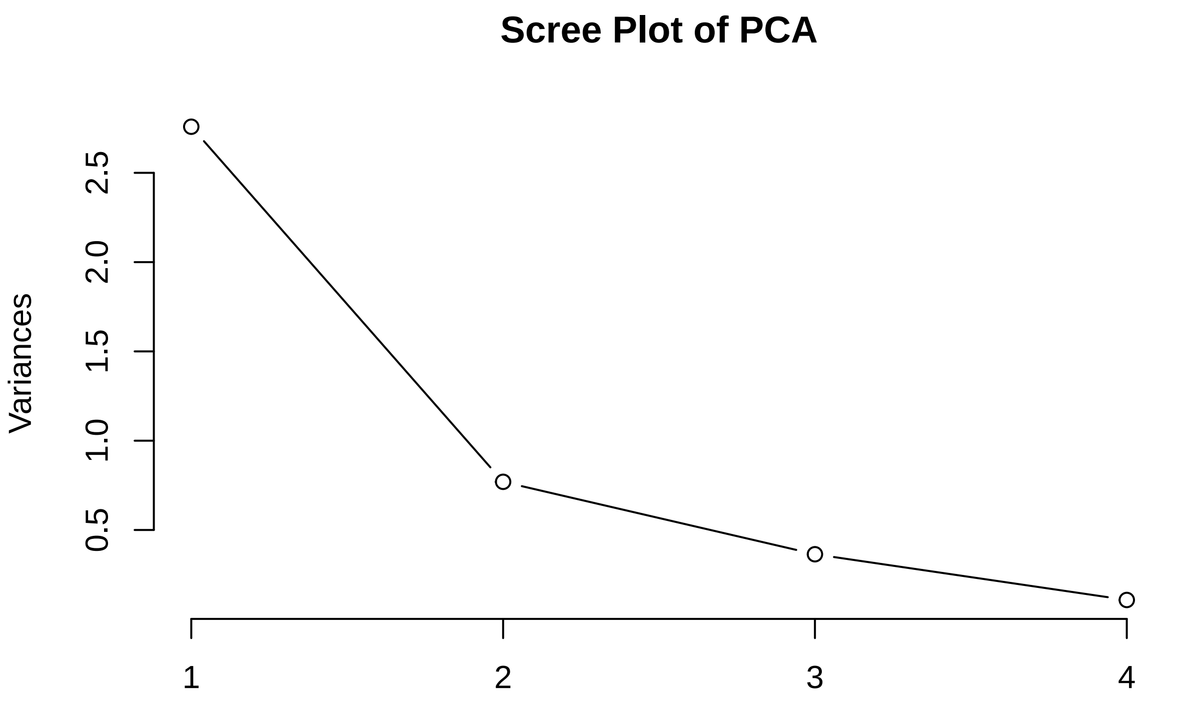
We first scale the numeric features and then perform PCA to determine which components account for the most variance.

```r
# Select numeric features and remove species column
numeric_data = data %>% select(bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g) %>% na.om

# Scale the data
scaled_data = scale(numeric_data)

# Perform PCA
pca_result = prcomp(scaled_data, center = TRUE, scale. = TRUE)
```

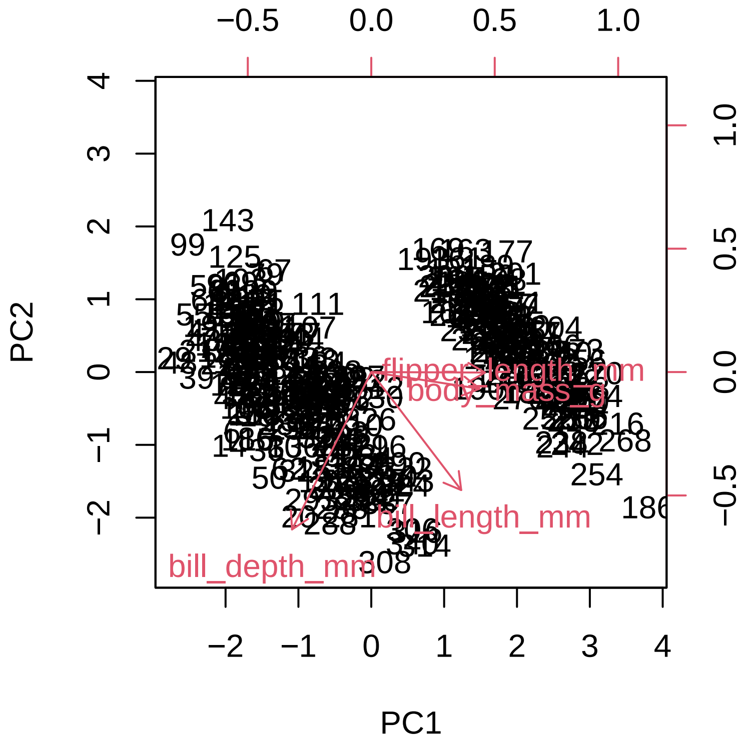
```r
# Summary of PCA
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4
## Standard deviation     1.6608 0.8773 0.60336 0.3286
## Proportion of Variance 0.6896 0.1924 0.09101 0.0270
## Cumulative Proportion  0.6896 0.8820 0.97300 1.0000
```

```r
# Plot the proportion of variance explained
plot(pca_result, type = "l", main = "Scree Plot of PCA")
```



```r
# Biplot of the first two principal components
biplot(pca_result, scale = 0)
```



Interpretation of PCA Results

- **Principal Components**: The first two principal components explain a significant portion of the variance in the data, with PC1 capturing the most variance, followed by PC2. This suggests that most of the information in the dataset can be effectively represented in two dimensions.
- **Scree Plot**: The scree plot indicates that the first two components explain most of the variability in the dataset, suggesting they are sufficient for further analysis.
- **Biplot**: The biplot helps visualize the relationship between variables and observations in the context of the principal components. It shows how different variables contribute to the principal components and how the species are distributed.

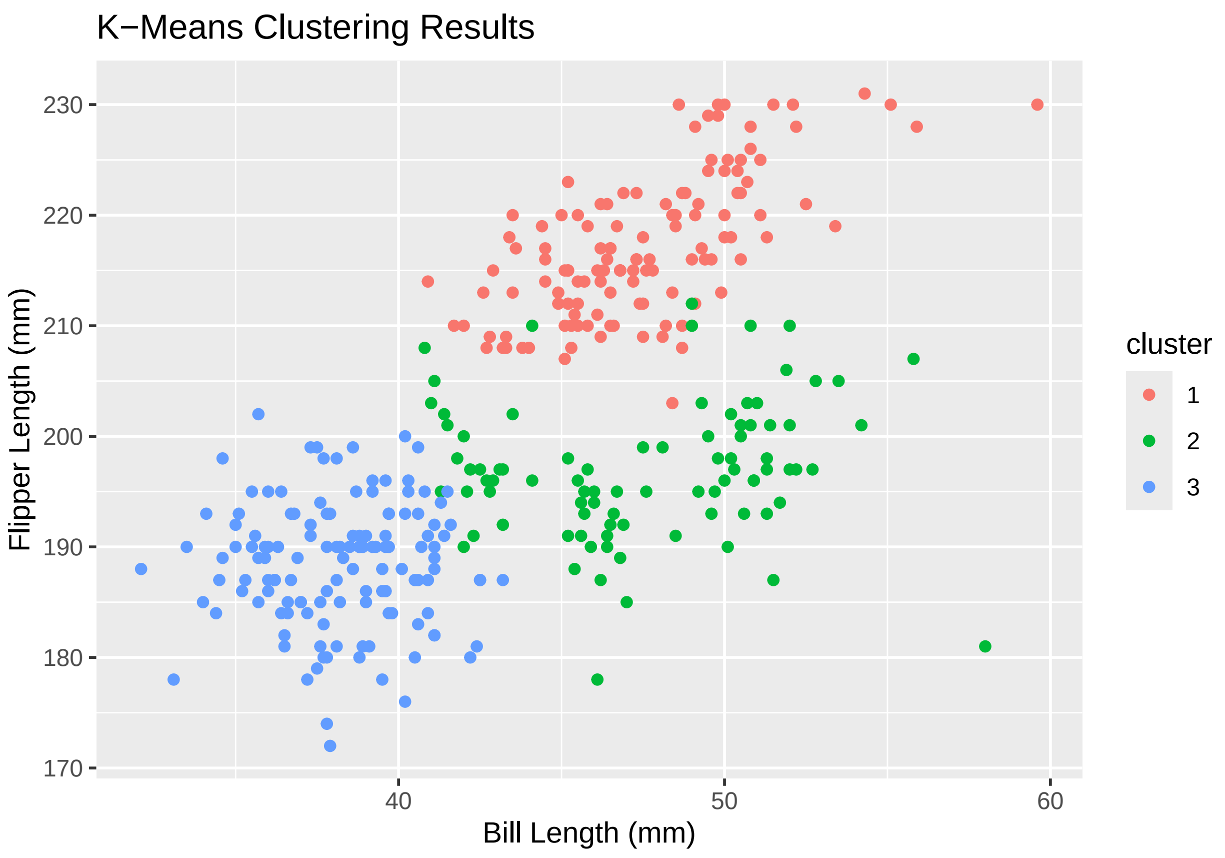
Step 5: K-Means Clustering

We apply K-means clustering to the numerical variables to identify patterns and group similar observations.

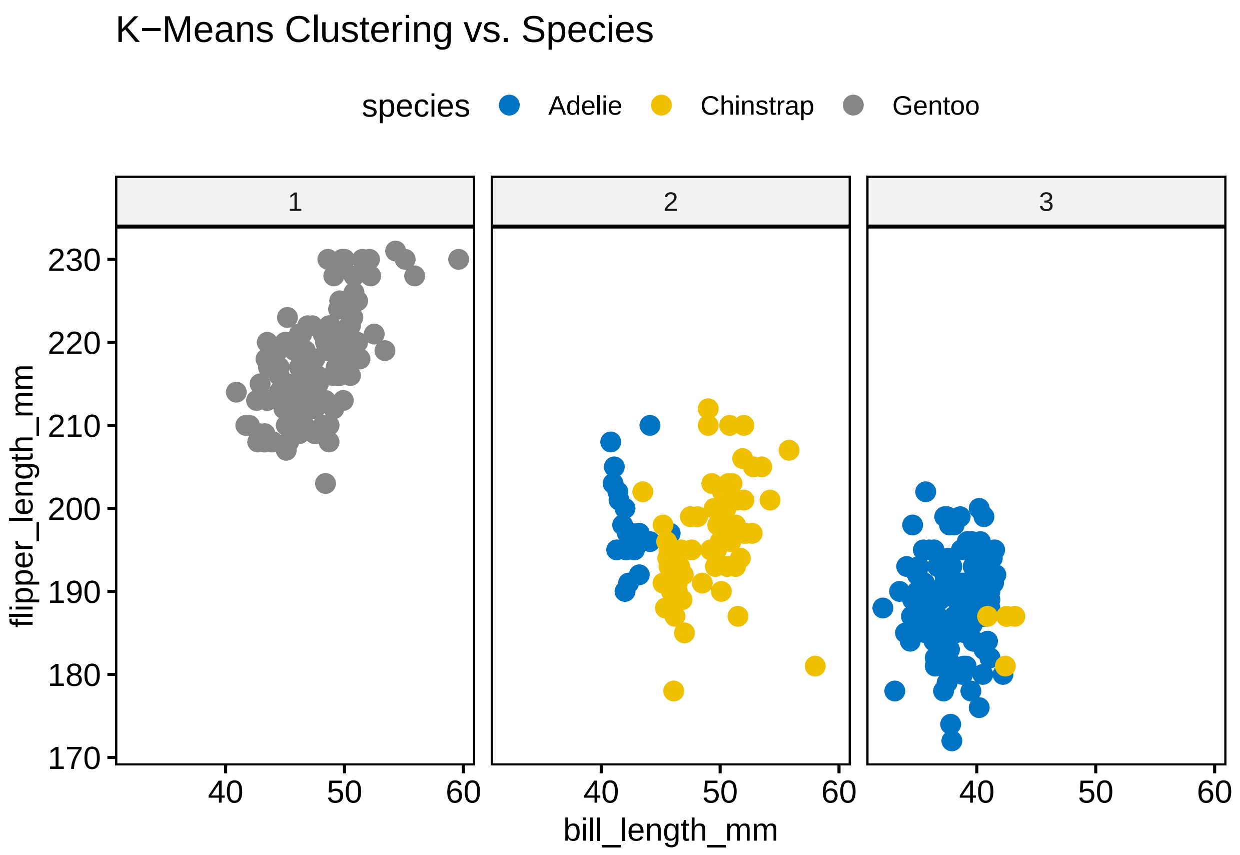
Performing K-Means Clustering

We use the scaled numerical data to perform K-means clustering with three clusters, as we know there are three species of penguins in the dataset.

```r
set.seed(123)

# Perform K-means clustering with 3 clusters
kmeans_result = kmeans(scaled_data, centers = 3, nstart = 25)

data$cluster = factor(kmeans_result$cluster)

# Plot the clustering result
ggplot(data, aes(x = bill_length_mm, y = flipper_length_mm, color = cluster)) +
  geom_point() +
  labs(title = "K-Means Clustering Results", x = "Bill Length (mm)", y = "Flipper Length (mm)")
```



```r
# Compare clusters with species
library(ggpubr)
ggscatter(data, x = "bill_length_mm", y = "flipper_length_mm", color = "species",
          palette = "jco", shape = 19, size = 3,
          facet.by = "cluster", title = "K-Means Clustering vs. Species")
```



Interpretation of K-Means Clustering Results

- **Cluster Analysis**: The K-means clustering resulted in three clusters, which roughly correspond to the three species of penguins. However, there may be some overlap or misclassification, which can be analyzed by comparing the clusters to the actual species labels.
- **Visualization**: The clustering plot and comparison with species indicate how well the K-means algorithm was able to separate the different species based on the numerical features. Some clusters might contain a mix of species, indicating that additional features or more sophisticated clustering methods may be needed to improve accuracy.

Conclusion

In this project, we covered advanced statistical analysis, data preprocessing, visualization, predictive modeling using a neural network, dimensionality reduction with PCA, and clustering using K-means. We implemented an interactive Shiny app to provide visual insights and allowed users to manipulate the dataset. This hands-on approach allowed us to deeply explore the dataset and apply machine learning techniques to predict penguin species.

Next Steps

- Improve the neural network model by tuning hyperparameters.
- Add additional features to the Shiny app for deeper analysis, such as model comparison and different model types.
- Experiment with other machine learning models, such as random forests or SVMs, for better accuracy.
- Further analyze clustering results using hierarchical clustering or Gaussian Mixture Models to improve cluster separation.