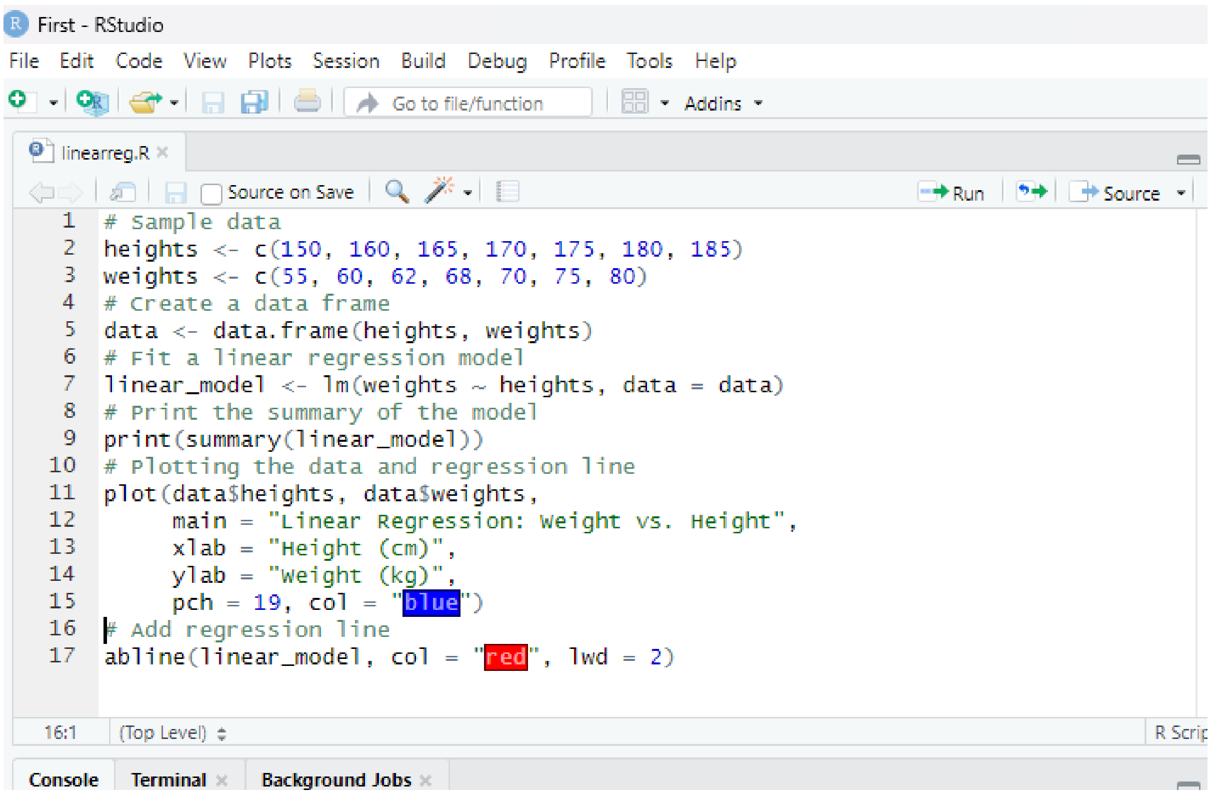
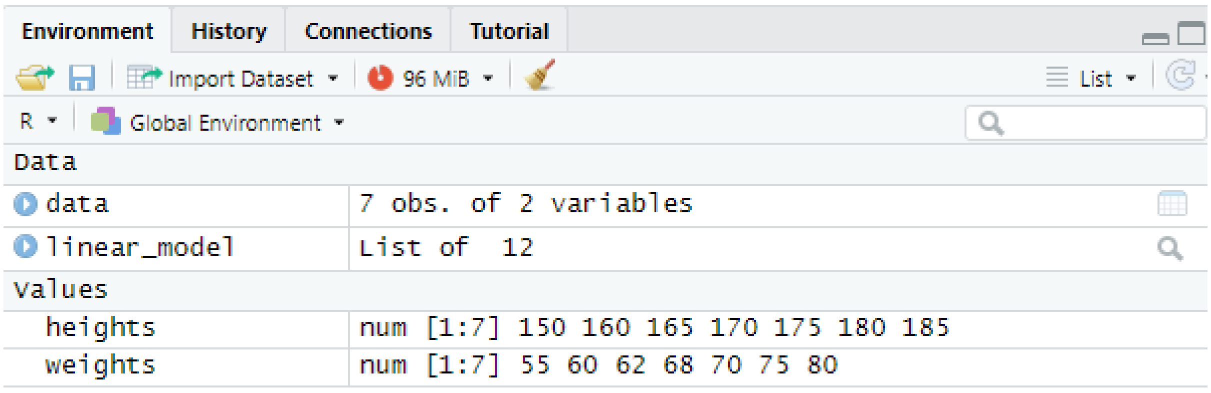
Exp:7 Implement Linear and Logistic Regression

a)Linear regression

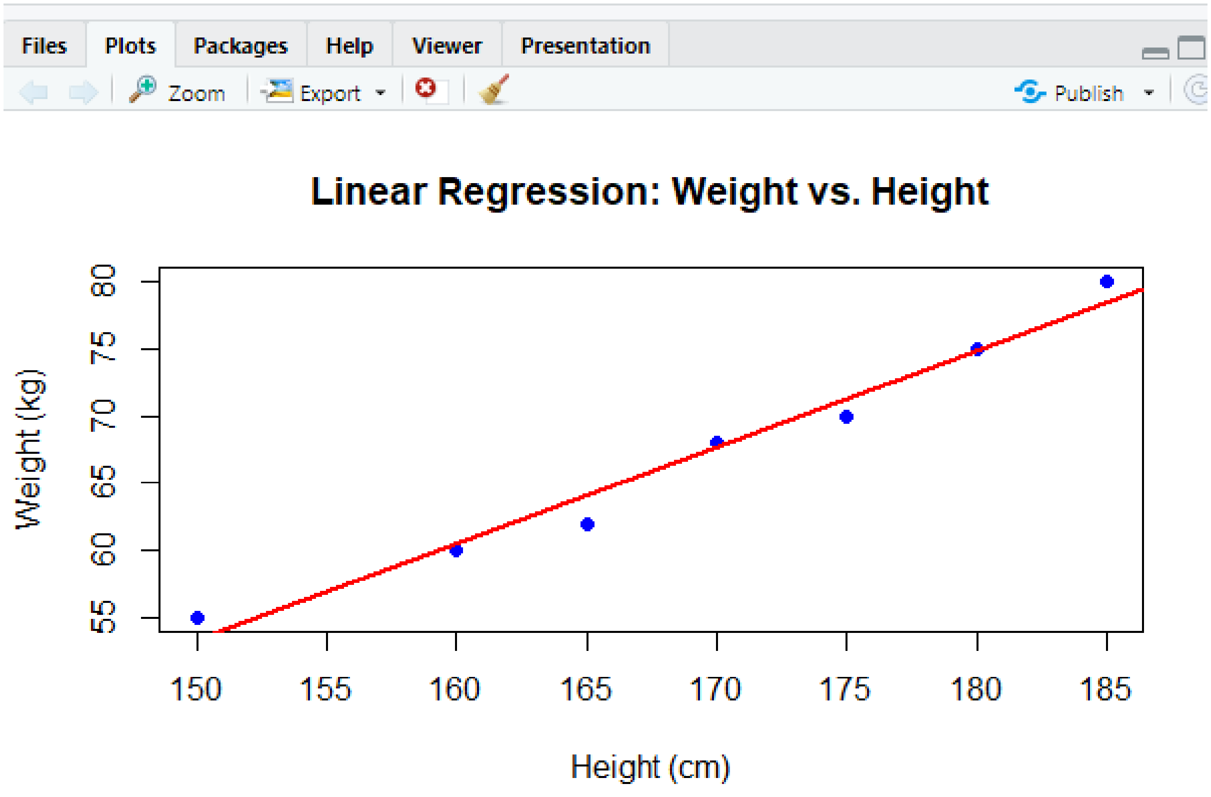
```
# Sample data
heights <- c(150, 160, 165, 170, 175, 180, 185)
weights <- c(55, 60, 62, 68, 70, 75, 80)
# Create a data frame
data <- data.frame(heights, weights)
# Fit a linear regression model
linear_model <- lm(weights ~ heights, data = data)
# Print the summary of the model
print(summary(linear_model))
# Plotting the data and regression line
plot(data$heights, data$weights,
main = "Linear Regression: Weight vs. Height",
xlab = "Height (cm)",
ylab = "Weight (kg)",
pch = 19, col = "blue")
# Add regression line
abline(linear_model, col = "red", lwd = 2)
```



```r
# Sample data
heights <- c(150, 160, 165, 170, 175, 180, 185)
weights <- c(55, 60, 62, 68, 70, 75, 80)
# Create a data frame
data <- data.frame(heights, weights)
# Fit a linear regression model
linear_model <- lm(weights ~ heights, data = data)
# Print the summary of the model
print(summary(linear_model))
# Plotting the data and regression line
plot(data$heights, data$weights,
     main = "Linear Regression: Weight vs. Height",
     xlab = "Height (cm)",
     ylab = "Weight (kg)",
     pch = 19, col = "blue")
# Add regression line
abline(linear_model, col = "red", lwd = 2)
```



```
Data
data            7 obs. of 2 variables
linear_model    List of  12
Values
  heights       num [1:7] 150 160 165 170 175 180 185
  weights       num [1:7] 55 60 62 68 70 75 80
```



b) Logistic regression

# Load the dataset

data(mtcars)

# Convert 'am' to a factor (categorical variable)

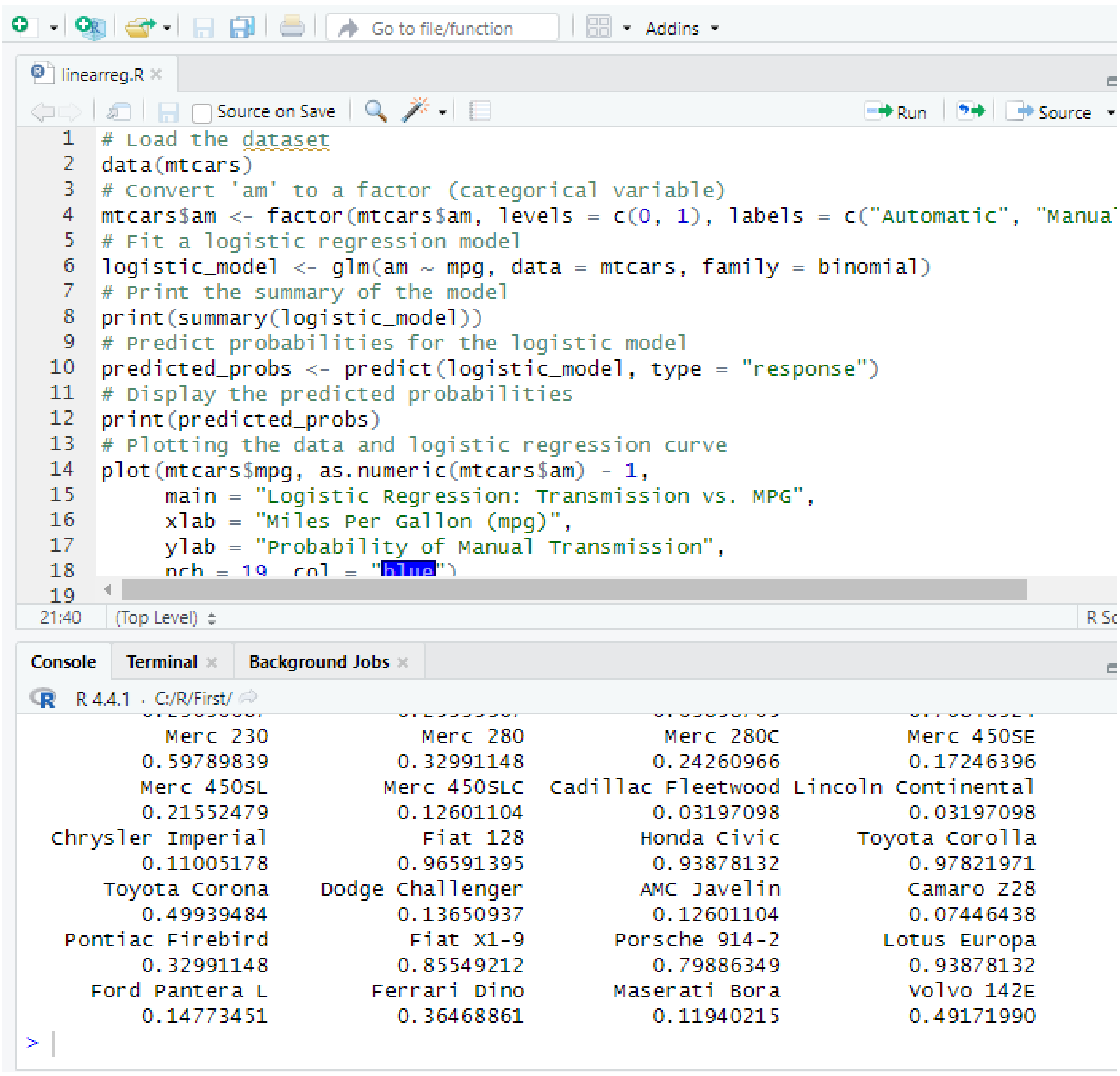
mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("Automatic", "Manual"))

# Fit a logistic regression model

logistic_model <- glm(am ~ mpg, data = mtcars, family = binomial)

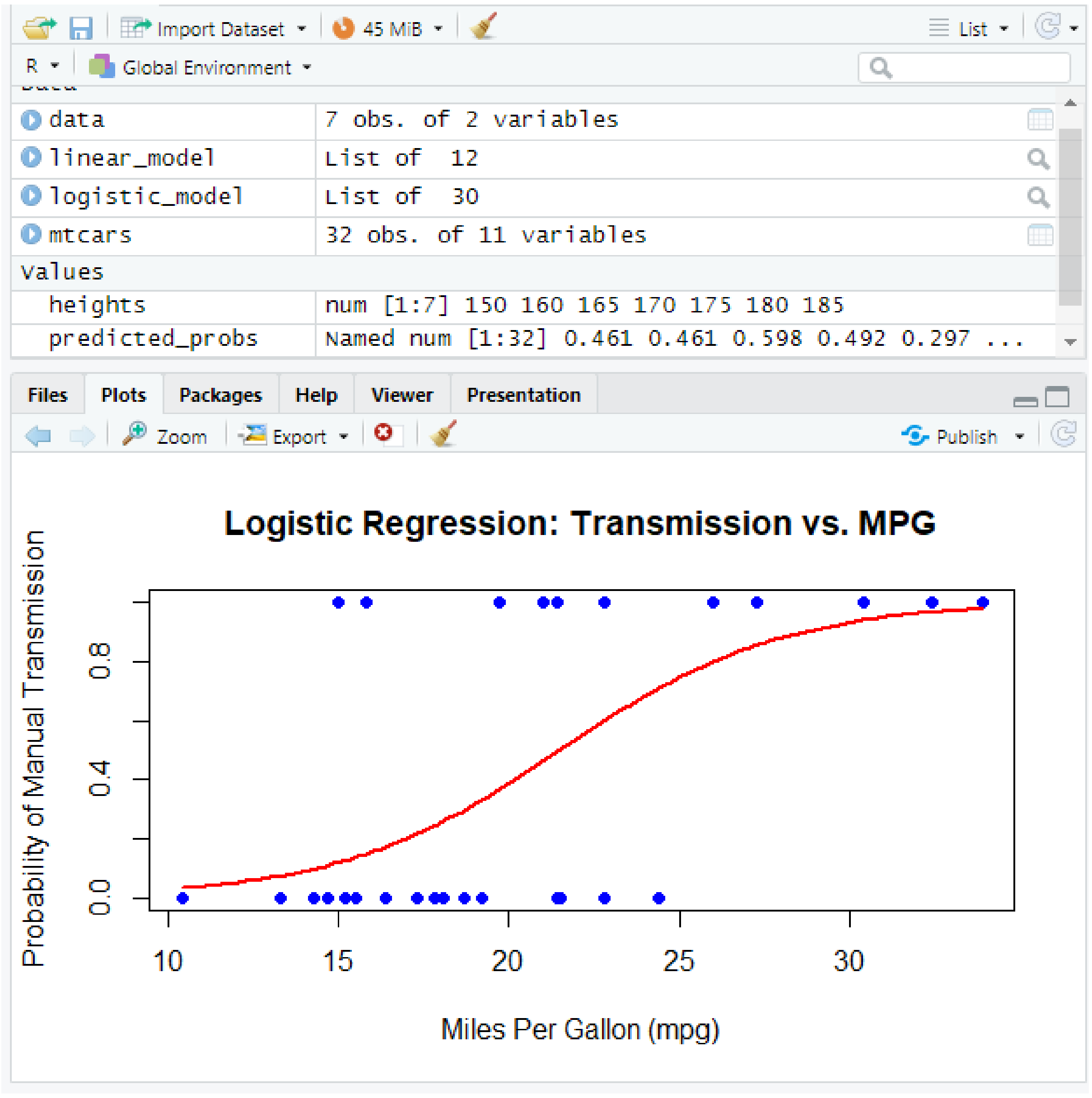
# Print the summary of the model

```
print(summary(logistic_model))

# Predict probabilities for the logistic model

predicted_probs <- predict(logistic_model, type = "response")

# Display the predicted probabilities

print(predicted_probs)

# Plotting the data and logistic regression curve

plot(mtcars$mpg, as.numeric(mtcars$am) - 1,

main = "Logistic Regression: Transmission vs. MPG",

xlab = "Miles Per Gallon (mpg)",

ylab = "Probability of Manual Transmission",

pch = 19, col = "blue")

# Add the logistic regression curve

curve(predict(logistic_model, data.frame(mpg = x), type = "response"),

add = TRUE, col = "red", lwd = 2)
```



```r
# Load the dataset
data(mtcars)
# Convert 'am' to a factor (categorical variable)
mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("Automatic", "Manua
# Fit a logistic regression model
logistic_model <- glm(am ~ mpg, data = mtcars, family = binomial)
# Print the summary of the model
print(summary(logistic_model))
# Predict probabilities for the logistic model
predicted_probs <- predict(logistic_model, type = "response")
# Display the predicted probabilities
print(predicted_probs)
# Plotting the data and logistic regression curve
plot(mtcars$mpg, as.numeric(mtcars$am) - 1,
     main = "Logistic Regression: Transmission vs. MPG",
     xlab = "Miles Per Gallon (mpg)",
     ylab = "Probability of Manual Transmission",
     pch = 19, col = "blue")
```

```
           Merc 230            Merc 280           Merc 280C          Merc 450SE 
         0.59789839          0.32991148          0.24260966          0.17246396 
         Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental 
         0.21552479          0.12601104          0.03197098          0.03197098 
  Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla 
         0.11005178          0.96591395          0.93878132          0.97821971 
      Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28 
         0.49939484          0.13650937          0.12601104          0.07446438 
   Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa 
         0.32991148          0.85549212          0.79886349          0.93878132 
     Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E 
         0.14773451          0.36468861          0.11940215          0.49171990 
> 
```

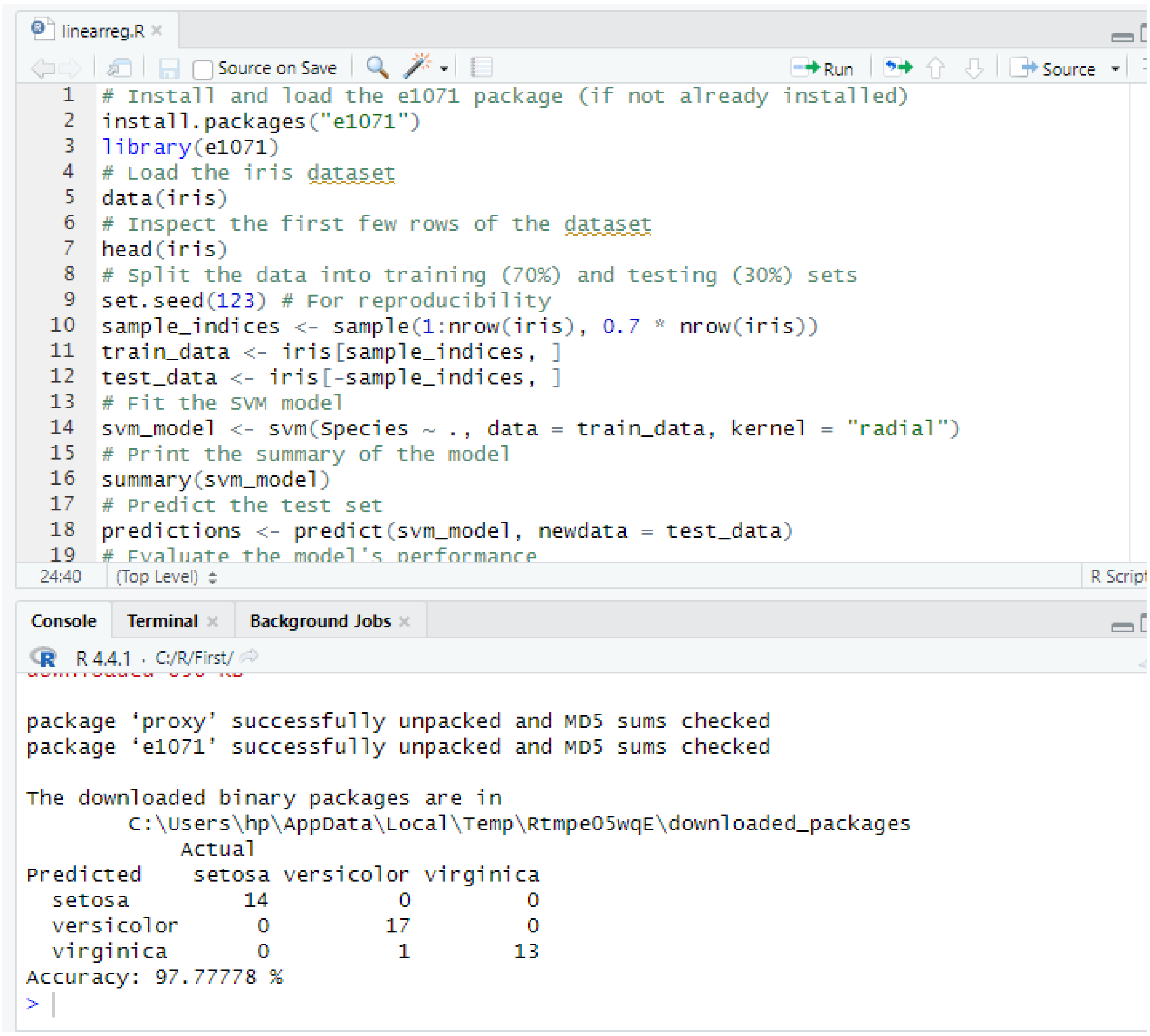


```
data              7 obs. of 2 variables
linear_model      List of  12
logistic_model    List of  30
mtcars            32 obs. of 11 variables
Values
  heights         num [1:7] 150 160 165 170 175 180 185
  predicted_probs Named num [1:32] 0.461 0.461 0.598 0.492 0.297 ...
```

Exp8: Implement SVM/Decision tree classification techniques

a) SVM IN R

```
# Install and load the e1071 package (if not already installed)

install.packages("e1071")

library(e1071)

# Load the iris dataset

data(iris)

# Inspect the first few rows of the dataset

head(iris)

# Split the data into training (70%) and testing (30%) sets

set.seed(123) # For reproducibility

sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))

train_data <- iris[sample_indices, ]

test_data <- iris[-sample_indices, ]

# Fit the SVM model

svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")

# Print the summary of the model

summary(svm_model)

# Predict the test set

predictions <- predict(svm_model, newdata = test_data)

# Evaluate the model's performance

confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)

print(confusion_matrix)

# Calculate accuracy

accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

cat("Accuracy:", accuracy * 100, "%\n")
```



```
# Install and load the e1071 package (if not already installed)
install.packages("e1071")
library(e1071)
# Load the iris dataset
data(iris)
# Inspect the first few rows of the dataset
head(iris)
# Split the data into training (70%) and testing (30%) sets
set.seed(123) # For reproducibility
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]
# Fit the SVM model
svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")
# Print the summary of the model
summary(svm_model)
# Predict the test set
predictions <- predict(svm_model, newdata = test_data)
# Evaluate the model's performance
```

```
package ‘proxy’ successfully unpacked and MD5 sums checked
package ‘e1071’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\hp\AppData\Local\Temp\Rtmpe05wqE\downloaded_packages
            Actual
Predicted    setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13
Accuracy: 97.77778 %
>
```

b) Decision tree in R

Install and load the rpart package (if not already installed)

install.packages("rpart")

library(rpart)

Load the iris dataset

data(iris)

Split the data into training (70%) and testing (30%) sets

set.seed(123) # For reproducibility

sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))

train_data <- iris[sample_indices,]

test_data <- iris[-sample_indices,]

Fit the Decision Tree model

tree_model <- rpart(Species ~ ., data = train_data, method = "class")

Print the summary of the model

summary(tree_model)

Plot the Decision Tree

plot(tree_model)

text(tree_model, pretty = 0)

Predict the test set

predictions <- predict(tree_model, newdata = test_data, type = "class")

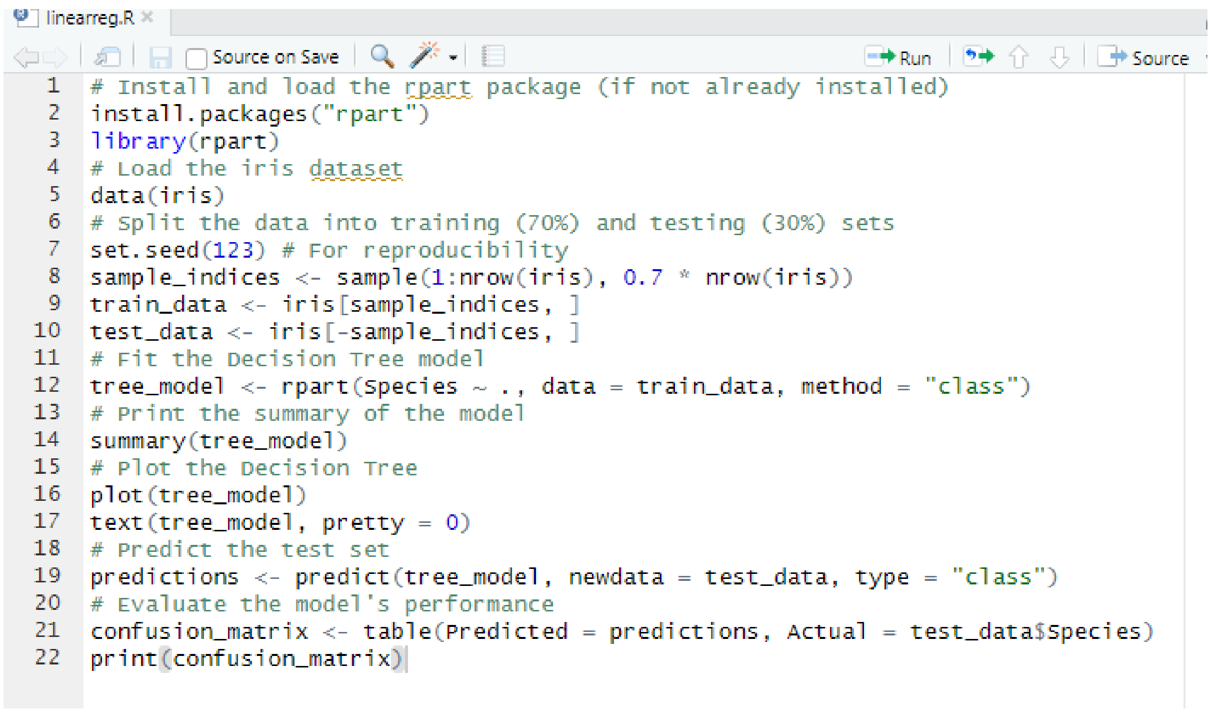
Evaluate the model's performance

confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)

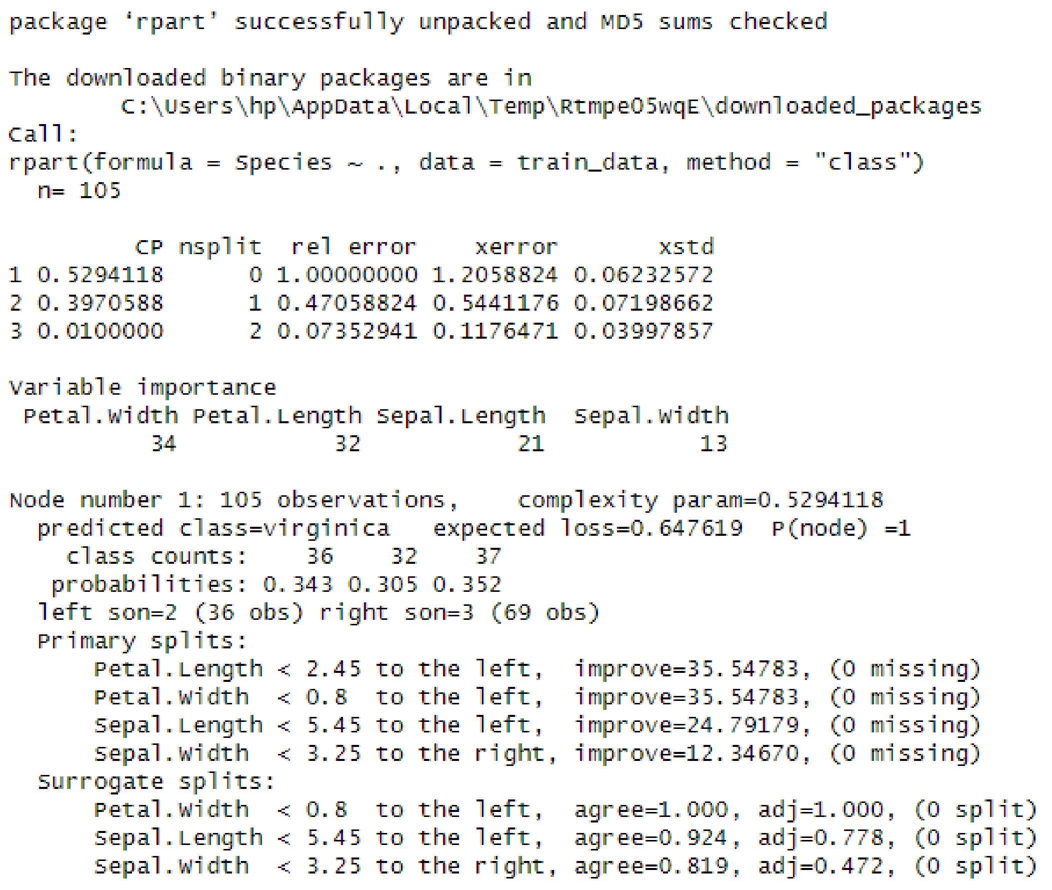
print(confusion_matrix)

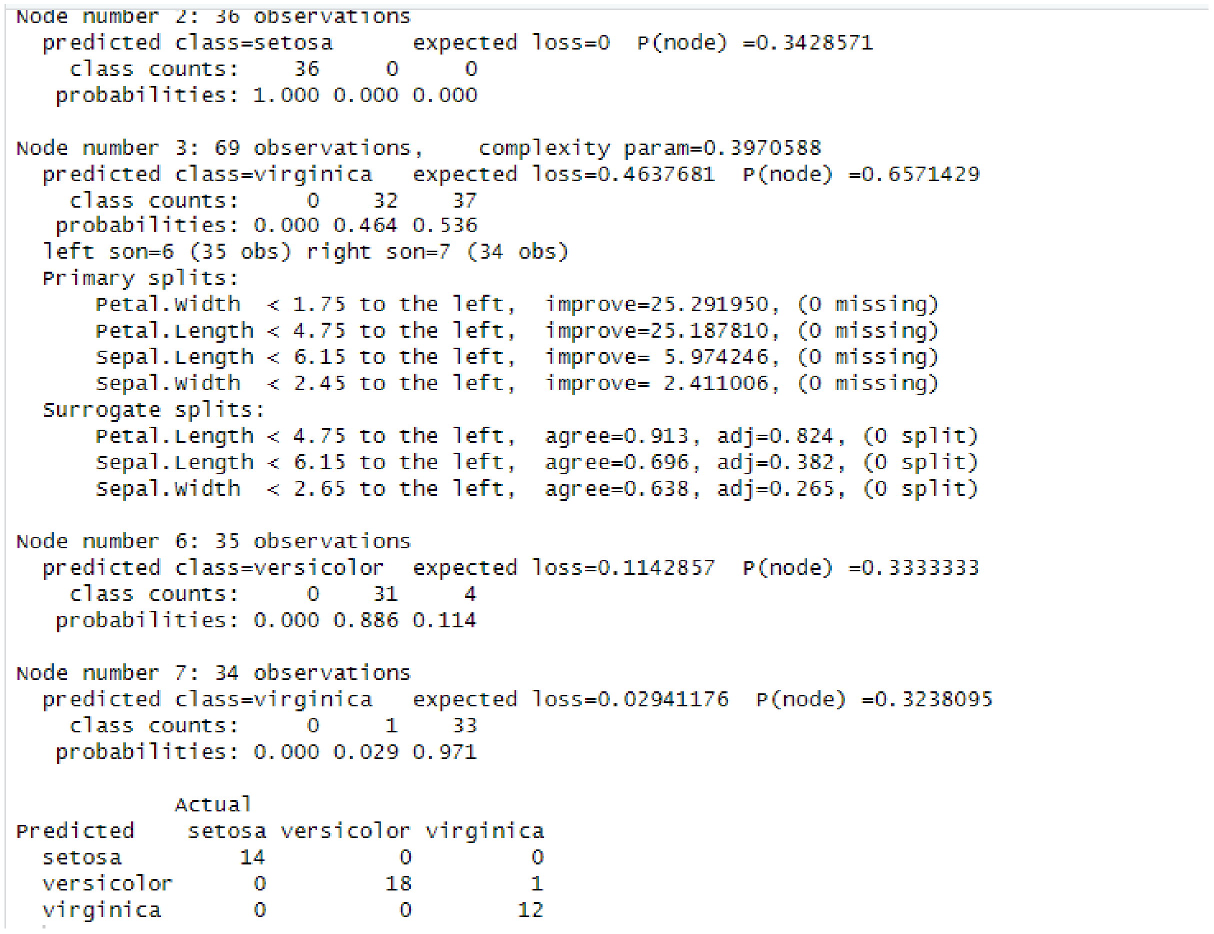
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

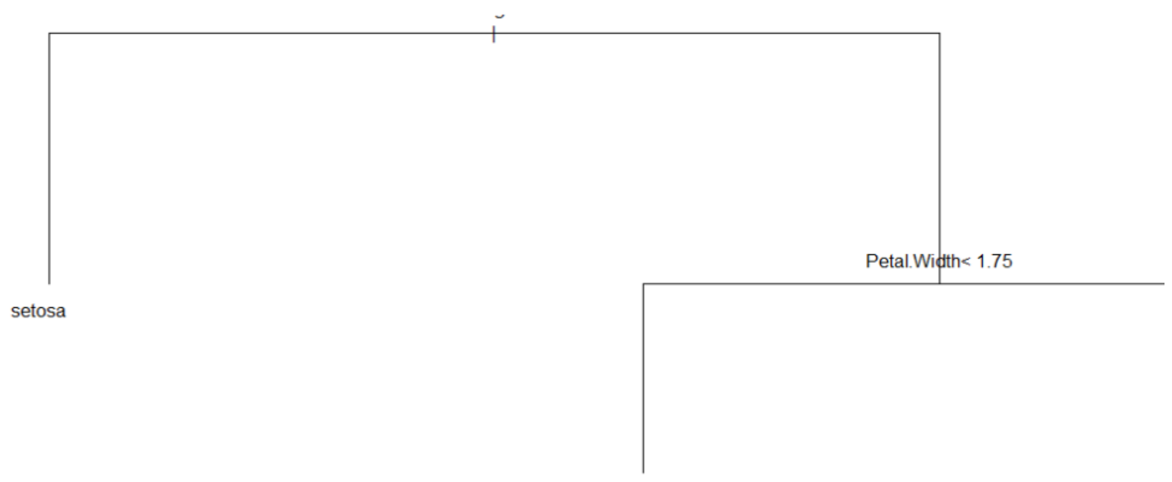
cat("Accuracy:", accuracy * 100, "%\n")



```
# Install and load the rpart package (if not already installed)
install.packages("rpart")
library(rpart)
# Load the iris dataset
data(iris)
# Split the data into training (70%) and testing (30%) sets
set.seed(123) # For reproducibility
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]
# Fit the Decision Tree model
tree_model <- rpart(Species ~ ., data = train_data, method = "class")
# Print the summary of the model
summary(tree_model)
# Plot the Decision Tree
plot(tree_model)
text(tree_model, pretty = 0)
# Predict the test set
predictions <- predict(tree_model, newdata = test_data, type = "class")
# Evaluate the model's performance
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
print(confusion_matrix)
```


```
package ‘rpart’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\hp\AppData\Local\Temp\Rtmpe05wqE\downloaded_packages
Call:
rpart(formula = Species ~ ., data = train_data, method = "class")
  n= 105

         CP nsplit  rel error    xerror       xstd
1 0.5294118      0 1.00000000 1.2058824 0.06232572
2 0.3970588      1 0.47058824 0.5441176 0.07198662
3 0.0100000      2 0.07352941 0.1176471 0.03997857

Variable importance
 Petal.Width Petal.Length Sepal.Length  Sepal.Width
         34           32           21           13

Node number 1: 105 observations,    complexity param=0.5294118
  predicted class=virginica   expected loss=0.647619  P(node) =1
    class counts:    36    32    37
   probabilities: 0.343 0.305 0.352
  left son=2 (36 obs) right son=3 (69 obs)
  Primary splits:
      Petal.Length < 2.45 to the left,  improve=35.54783, (0 missing)
      Petal.Width  < 0.8  to the left,  improve=35.54783, (0 missing)
      Sepal.Length < 5.45 to the left,  improve=24.79179, (0 missing)
      Sepal.Width  < 3.25 to the right, improve=12.34670, (0 missing)
  Surrogate splits:
      Petal.Width  < 0.8  to the left,  agree=1.000, adj=1.000, (0 split)
      Sepal.Length < 5.45 to the left,  agree=0.924, adj=0.778, (0 split)
      Sepal.Width  < 3.25 to the right, agree=0.819, adj=0.472, (0 split)
```

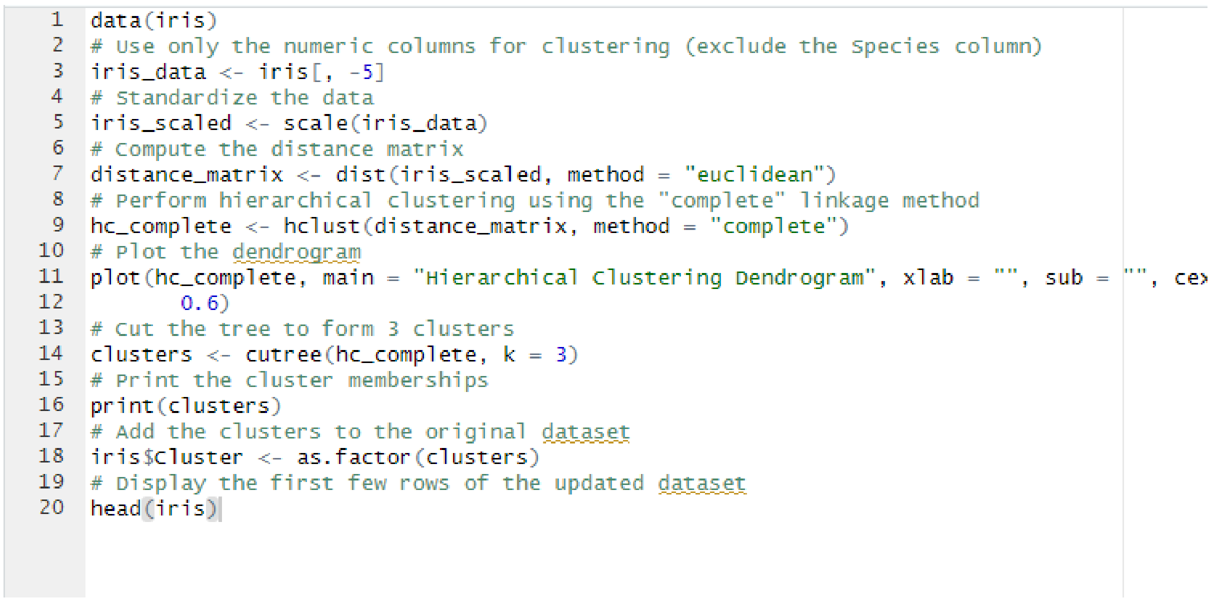

```
Node number 2: 36 observations
  predicted class=setosa      expected loss=0  P(node) =0.3428571
    class counts:    36     0     0
   probabilities: 1.000 0.000 0.000

Node number 3: 69 observations,    complexity param=0.3970588
  predicted class=virginica   expected loss=0.4637681  P(node) =0.6571429
    class counts:     0    32    37
   probabilities: 0.000 0.464 0.536
  left son=6 (35 obs) right son=7 (34 obs)
  Primary splits:
      Petal.Width  < 1.75 to the left,  improve=25.291950, (0 missing)
      Petal.Length < 4.75 to the left,  improve=25.187810, (0 missing)
      Sepal.Length < 6.15 to the left,  improve= 5.974246, (0 missing)
      Sepal.Width  < 2.45 to the left,  improve= 2.411006, (0 missing)
  Surrogate splits:
      Petal.Length < 4.75 to the left,  agree=0.913, adj=0.824, (0 split)
      Sepal.Length < 6.15 to the left,  agree=0.696, adj=0.382, (0 split)
      Sepal.Width  < 2.65 to the left,  agree=0.638, adj=0.265, (0 split)

Node number 6: 35 observations
  predicted class=versicolor  expected loss=0.1142857  P(node) =0.3333333
    class counts:     0    31     4
   probabilities: 0.000 0.886 0.114

Node number 7: 34 observations
  predicted class=virginica   expected loss=0.02941176  P(node) =0.3238095
    class counts:     0     1    33
   probabilities: 0.000 0.029 0.971

            Actual
Predicted    setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         18         1
  virginica       0          0        12
```


Petal.Width< 1.75
setosa

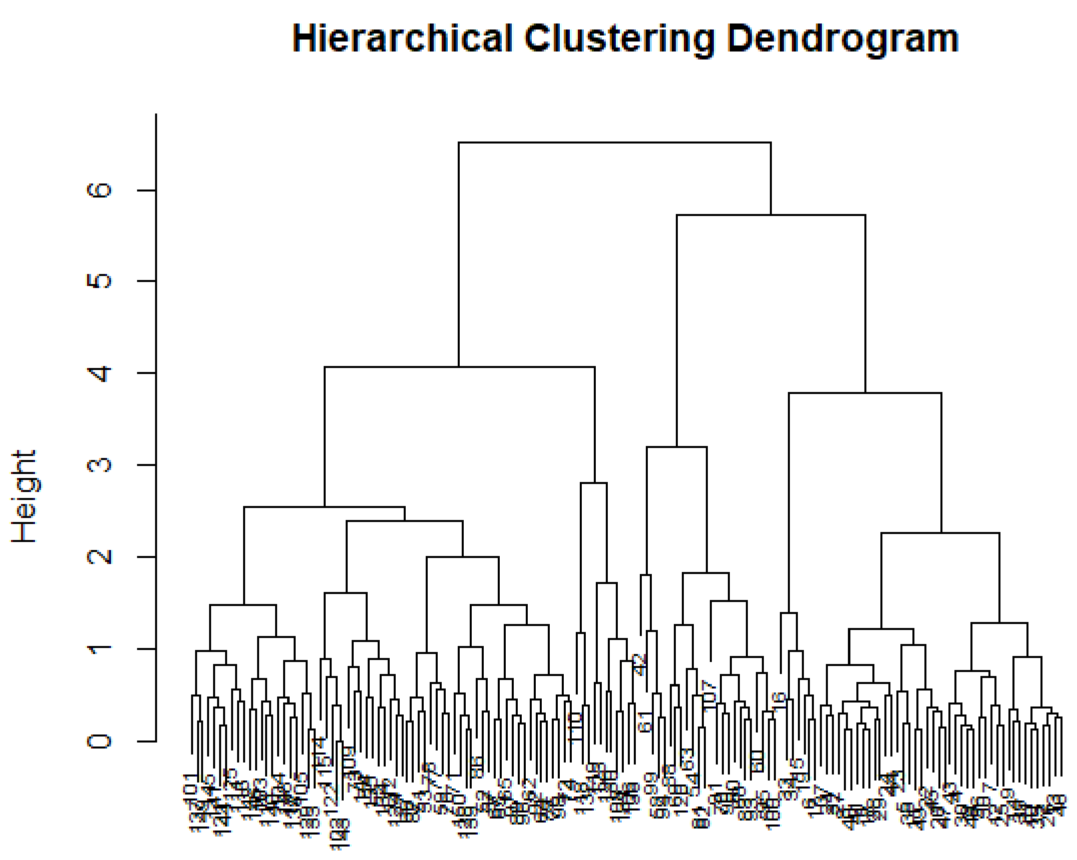
Exp:9 Implement clustering techniques – Hierarchical and K-Means

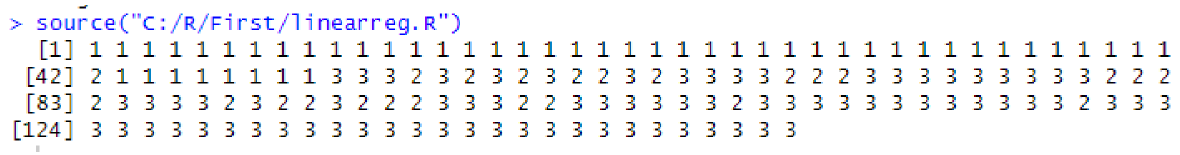
a) HIERARCHIAL CLUSTERING

```
# Load the iris dataset

data(iris)

# Use only the numeric columns for clustering (exclude the Species column)

iris_data <- iris[, -5]

# Standardize the data

iris_scaled <- scale(iris_data)

# Compute the distance matrix

distance_matrix <- dist(iris_scaled, method = "euclidean")

# Perform hierarchical clustering using the "complete" linkage method

hc_complete <- hclust(distance_matrix, method = "complete")

# Plot the dendrogram

plot(hc_complete, main = "Hierarchical Clustering Dendrogram", xlab = "", sub = "", cex =

0.6)

# Cut the tree to form 3 clusters

clusters <- cutree(hc_complete, k = 3)

# Print the cluster memberships

print(clusters)

# Add the clusters to the original dataset

iris$Cluster <- as.factor(clusters)

# Display the first few rows of the updated dataset

head(iris)
```



```r
data(iris)
# Use only the numeric columns for clustering (exclude the Species column)
iris_data <- iris[, -5]
# Standardize the data
iris_scaled <- scale(iris_data)
# Compute the distance matrix
distance_matrix <- dist(iris_scaled, method = "euclidean")
# Perform hierarchical clustering using the "complete" linkage method
hc_complete <- hclust(distance_matrix, method = "complete")
# Plot the dendrogram
plot(hc_complete, main = "Hierarchical Clustering Dendrogram", xlab = "", sub = "", ce
       0.6)
# Cut the tree to form 3 clusters
clusters <- cutree(hc_complete, k = 3)
# Print the cluster memberships
print(clusters)
# Add the clusters to the original dataset
iris$cluster <- as.factor(clusters)
# Display the first few rows of the updated dataset
head(iris)
```

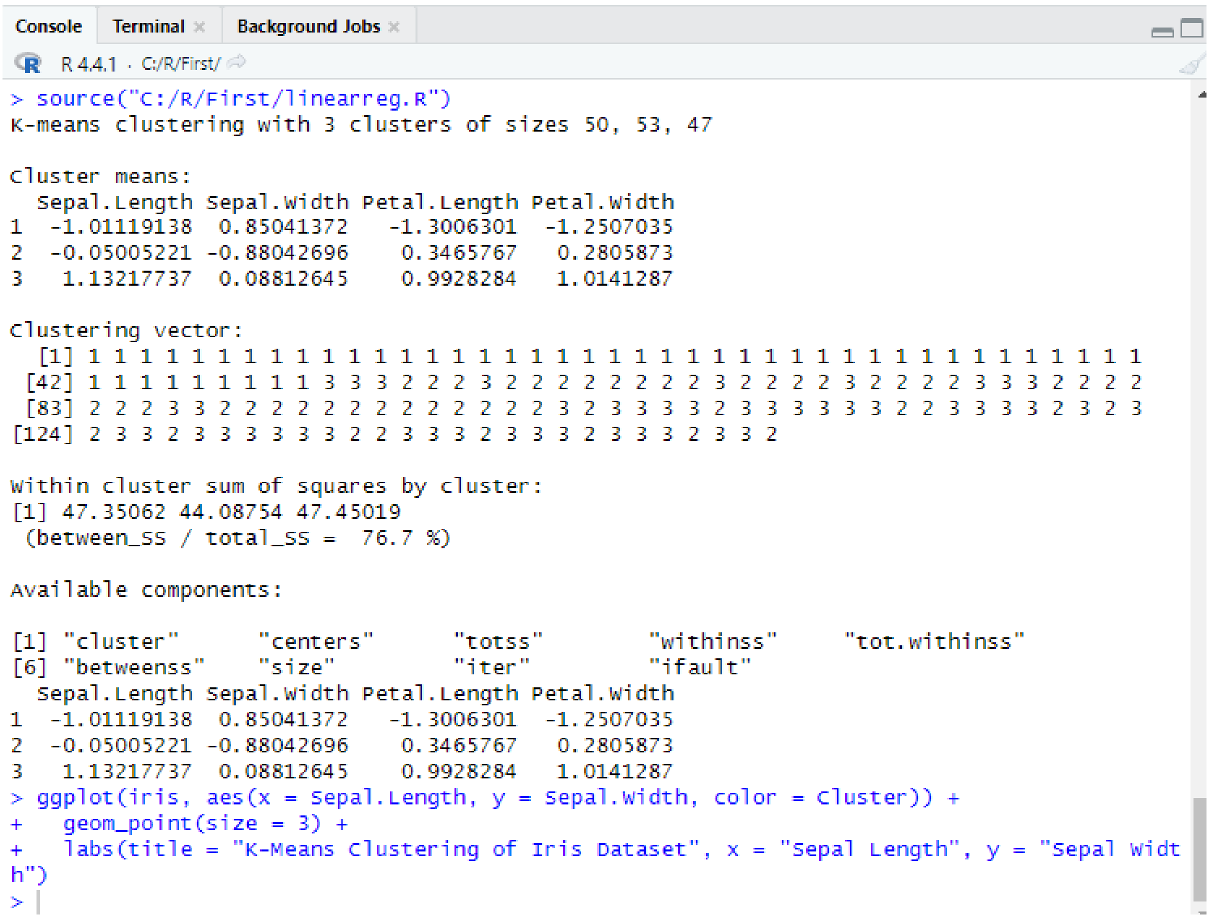




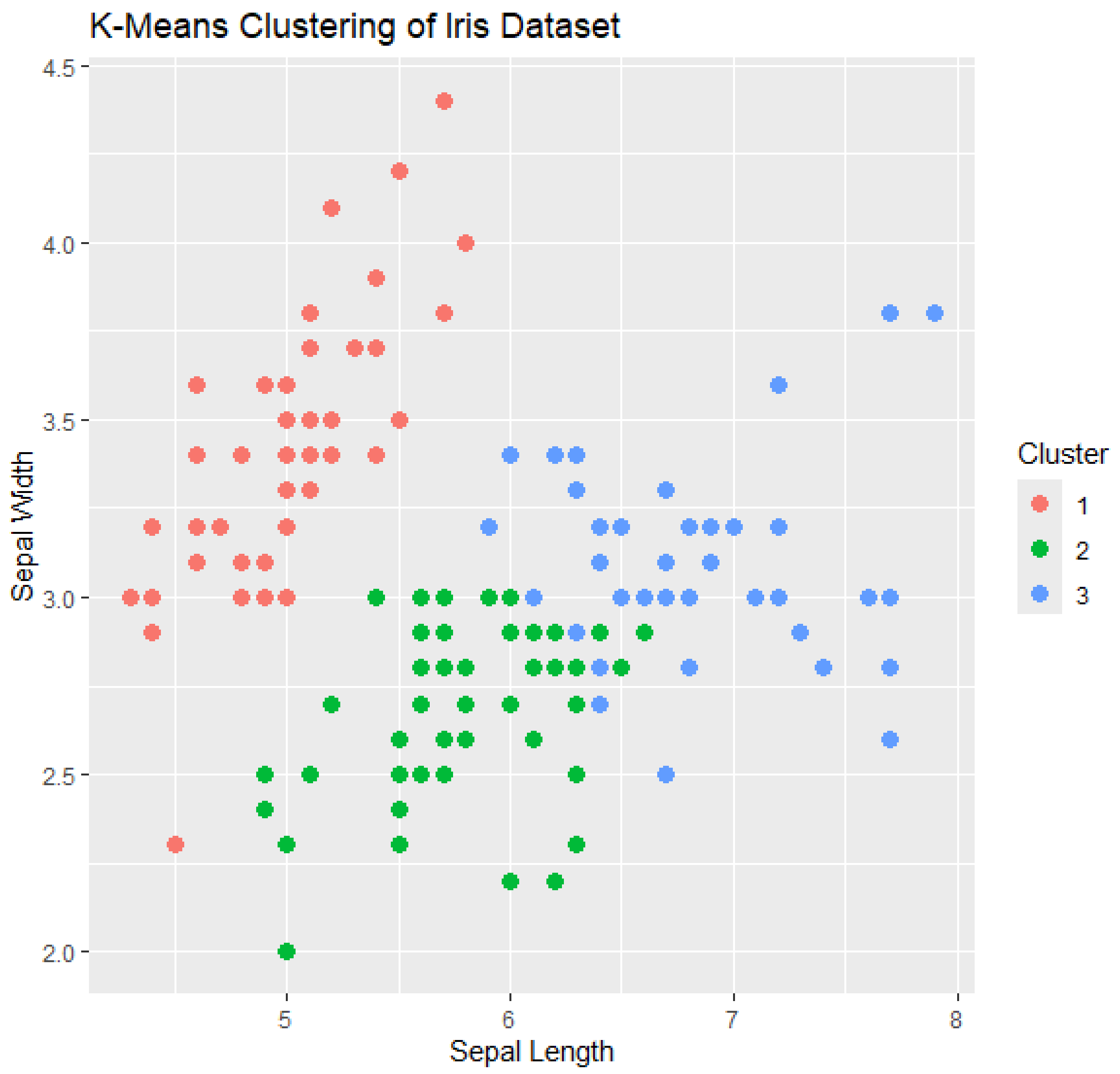
```
> source("C:/R/First/linearreg.R")
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [42] 2 1 1 1 1 1 1 1 1 3 3 3 2 3 2 3 2 3 2 2 3 2 3 3 3 3 2 2 2 3 3 3 3 3 3 3 3 3 2 2 2
 [83] 2 3 3 3 3 2 3 2 2 3 2 2 2 3 3 3 2 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
[124] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

b) K-MEANS CLUSTERING

```
# Load the iris dataset

data(iris)

# Use only the numeric columns for clustering (exclude the Species column)

iris_data <- iris[, -5]


# Standardize the data

iris_scaled <- scale(iris_data)

# Set the number of clusters

set.seed(123) # For reproducibility

k <- 3 # Number of clusters

# Perform K-Means clustering

kmeans_result <- kmeans(iris_scaled, centers = k, nstart = 25)

# Print the K-Means result

print(kmeans_result)

# Print the cluster centers

print(kmeans_result$centers)

# Add the cluster assignments to the original dataset

iris$Cluster <- as.factor(kmeans_result$cluster)

# Display the first few rows of the updated dataset

head(iris)

# Plot the clusters

library(ggplot2)

ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster)) +

geom_point(size = 3) +

labs(title = "K-Means Clustering of Iris Dataset", x = "Sepal Length", y = "Sepal Width")
```



```
> source("C:/R/First/linearreg.R")
K-means clustering with 3 clusters of sizes 50, 53, 47

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2  -0.05005221 -0.88042696    0.3465767   0.2805873
3   1.13217737  0.08812645    0.9928284   1.0141287

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [42] 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2 2 3 3 3 2 2 2 2
 [83] 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3
[124] 2 3 3 2 3 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 47.35062 44.08754 47.45019
 (between_SS / total_SS =  76.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2  -0.05005221 -0.88042696    0.3465767   0.2805873
3   1.13217737  0.08812645    0.9928284   1.0141287
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Cluster)) +
+   geom_point(size = 3) +
+   labs(title = "K-Means Clustering of Iris Dataset", x = "Sepal Length", y = "Sepal Widt
h")
> 
```



K-Means Clustering of Iris Dataset
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
5
6
7
8
Cluster
1
2
3

Exp:10 VISUALIZE DATA USING ANY PLOTTING FRAMEWORK

1) SCATTER PLOT

# Install ggplot2 (if not already installed)

install.packages("ggplot2")

# Load the ggplot2 package

library(ggplot2)

Scatter plot of Sepal.Length vs Sepal.Width, colored by Species

ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +

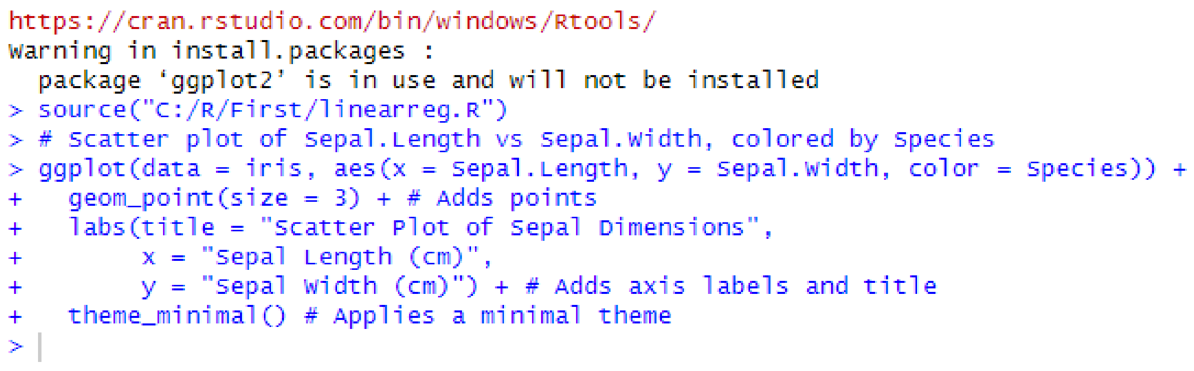
geom_point(size = 3) + # Adds points

labs(title = "Scatter Plot of Sepal Dimensions",

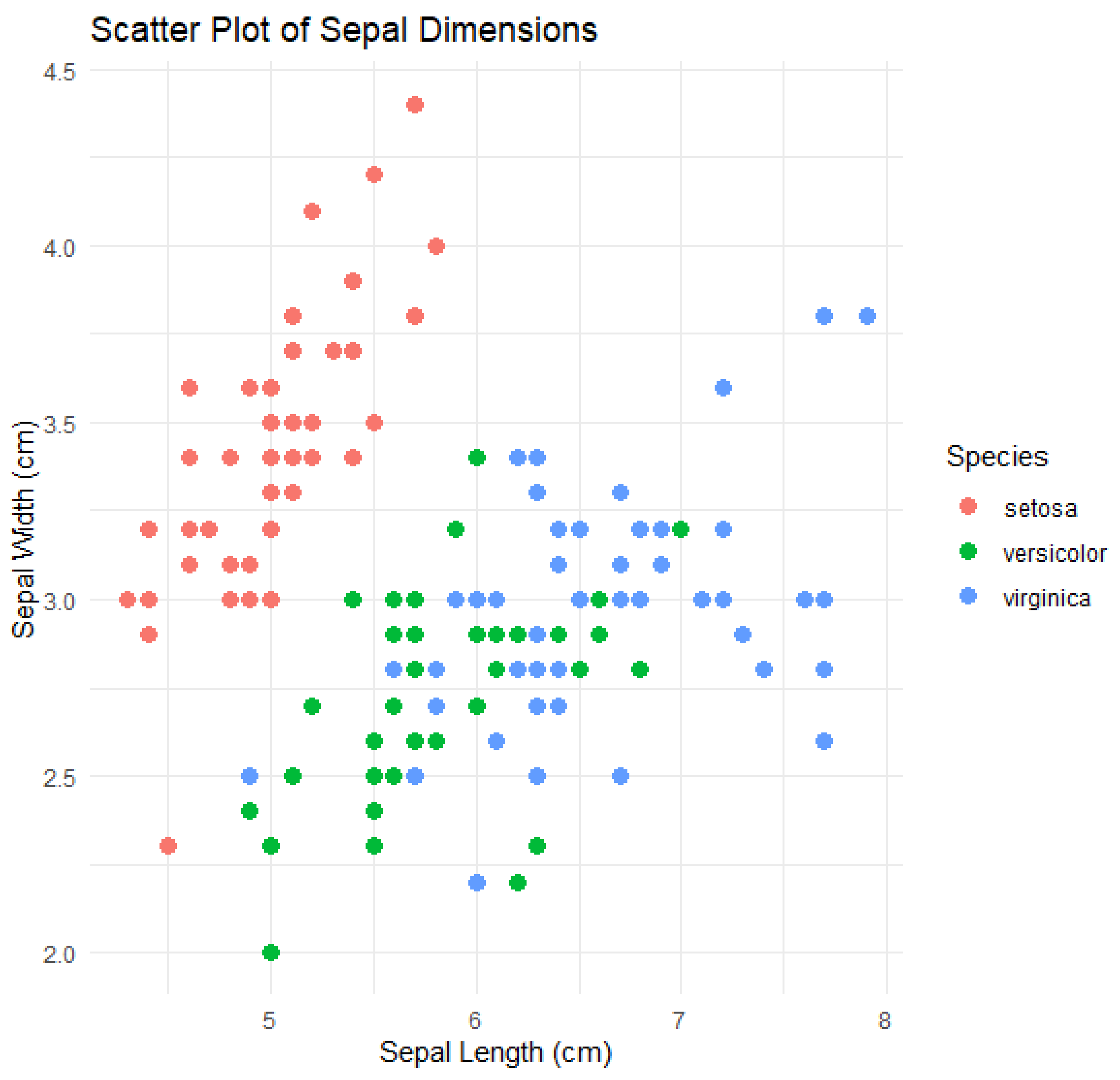
x = "Sepal Length (cm)",

y = "Sepal Width (cm)") + # Adds axis labels and title

theme_minimal() # Applies a minimal theme



```
https://cran.rstudio.com/bin/windows/Rtools/
Warning in install.packages :
  package ‘ggplot2’ is in use and will not be installed
> source("C:/R/First/linearreg.R")
> # Scatter plot of Sepal.Length vs Sepal.Width, colored by Species
> ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
+   geom_point(size = 3) + # Adds points
+   labs(title = "Scatter Plot of Sepal Dimensions",
+        x = "Sepal Length (cm)",
+        y = "Sepal Width (cm)") + # Adds axis labels and title
+   theme_minimal() # Applies a minimal theme
> 
```



2) BAR CHART

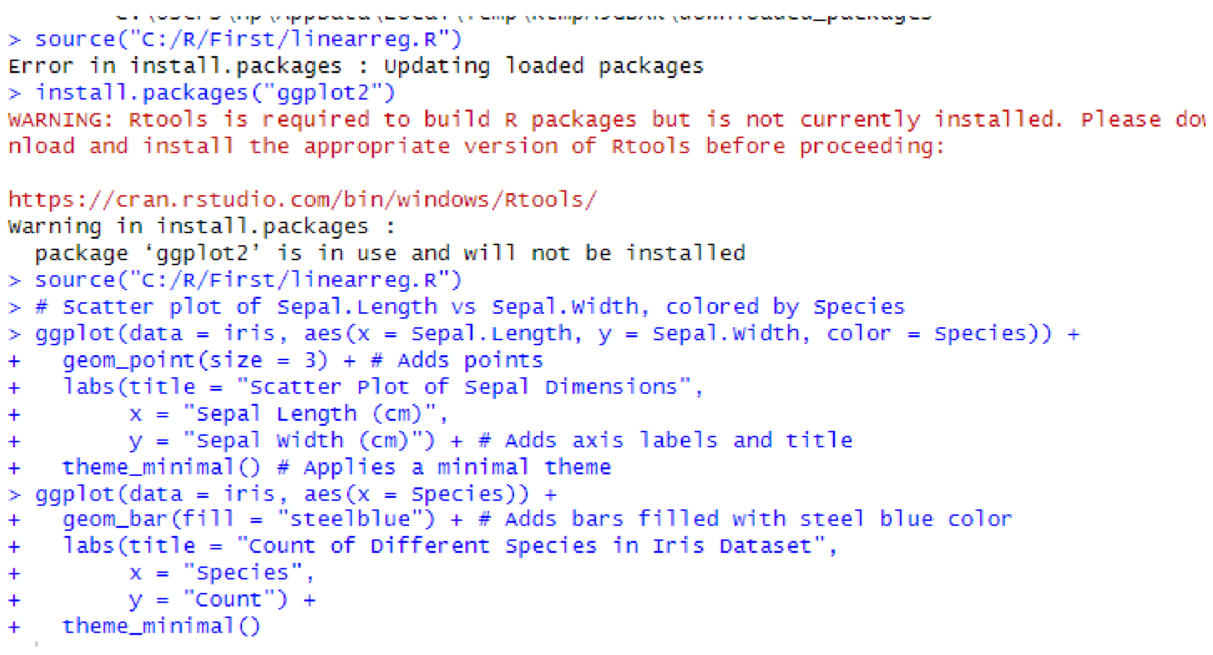
```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")
# Load the ggplot2 package
library(ggplot2)
# Bar plot of Species counts
ggplot(data = iris, aes(x = Species)) +
geom_bar(fill = "steelblue") + # Adds bars filled with steel blue color
```

labs(title = "Count of Different Species in Iris Dataset",

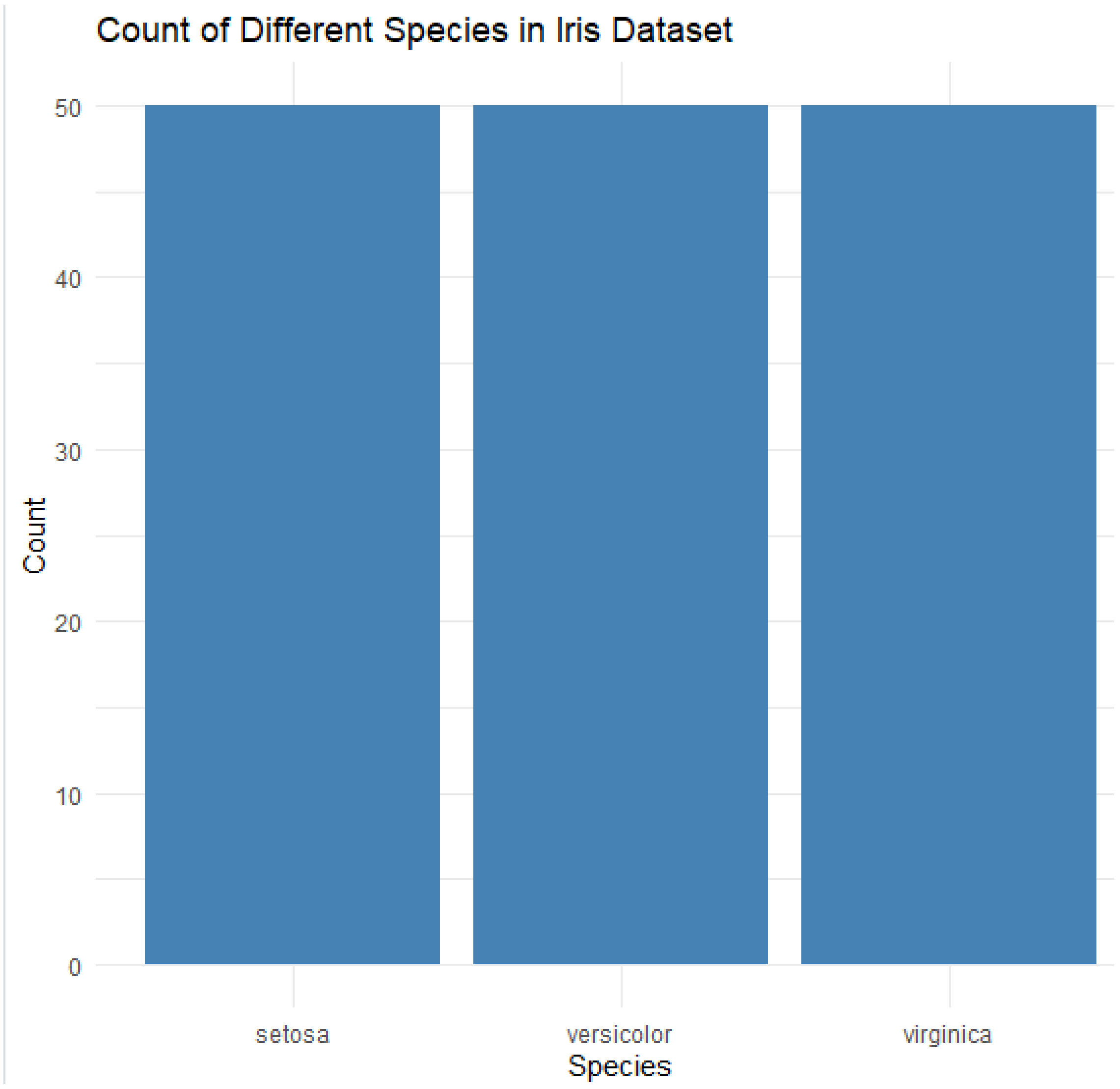
x = "Species",

y = "Count") +

theme_minimal()



```
> source("C:/R/First/linearreg.R")
Error in install.packages : Updating loaded packages
> install.packages("ggplot2")
WARNING: Rtools is required to build R packages but is not currently installed. Please dow
nload and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Warning in install.packages :
  package ‘ggplot2’ is in use and will not be installed
> source("C:/R/First/linearreg.R")
> # Scatter plot of Sepal.Length vs Sepal.Width, colored by Species
> ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
+   geom_point(size = 3) + # Adds points
+   labs(title = "Scatter Plot of Sepal Dimensions",
+        x = "Sepal Length (cm)",
+        y = "Sepal Width (cm)") + # Adds axis labels and title
+   theme_minimal() # Applies a minimal theme
> ggplot(data = iris, aes(x = Species)) +
+   geom_bar(fill = "steelblue") + # Adds bars filled with steel blue color
+   labs(title = "Count of Different Species in Iris Dataset",
+        x = "Species",
+        y = "Count") +
+   theme_minimal()
```



3) HISTOGRAM

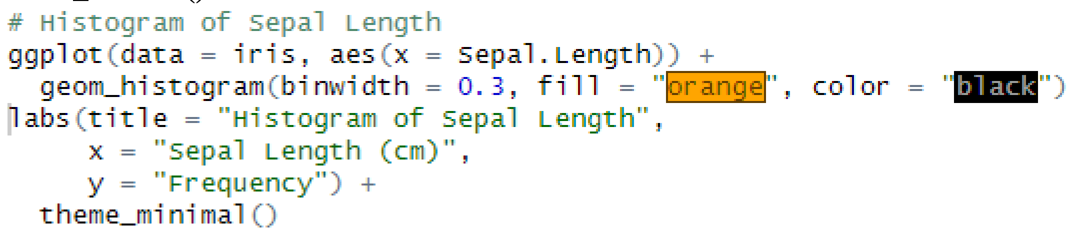
```
# Install ggplot2 (if not already installed)
install.packages("ggplot2")
# Load the ggplot2 package
library(ggplot2)
# Histogram of Sepal Length
ggplot(data = iris, aes(x = Sepal.Length)) +
geom_histogram(binwidth = 0.3, fill = "orange", color = "black") + # Adds
histogram bars
```

labs(title = "Histogram of Sepal Length",

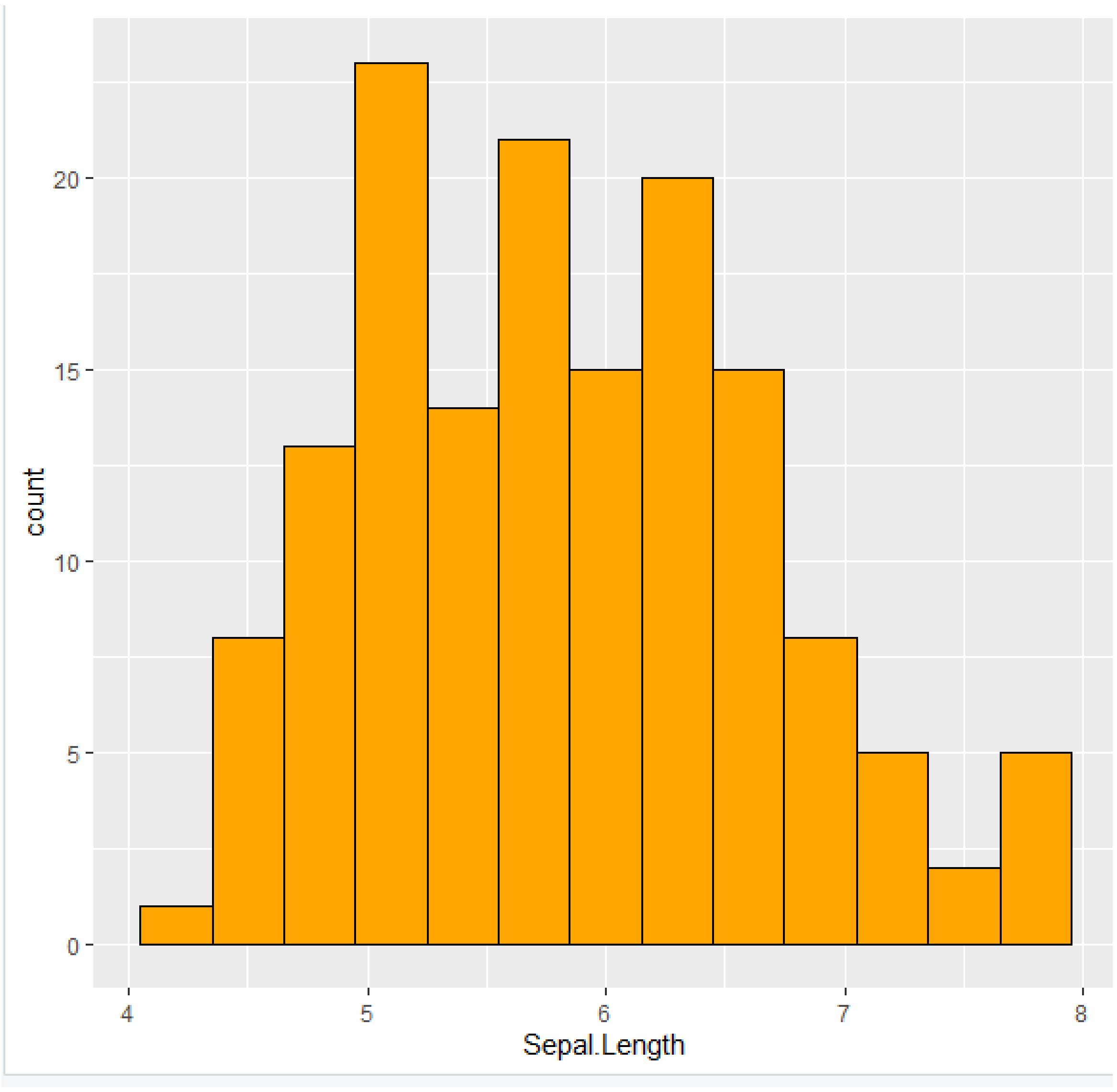
x = "Sepal Length (cm)",

y = "Frequency") +

theme_minimal()

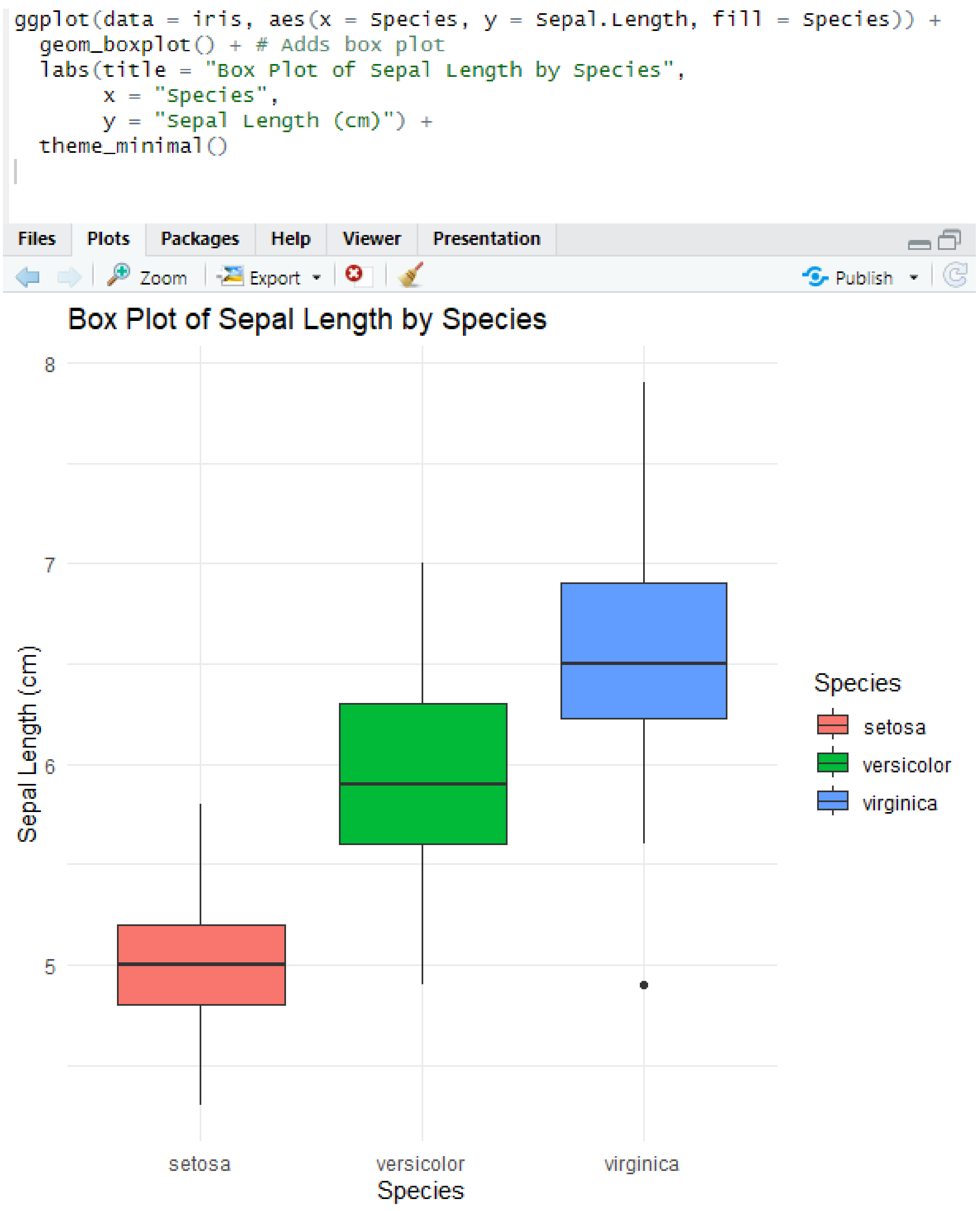


```
# Histogram of Sepal Length
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram(binwidth = 0.3, fill = "orange", color = "black")
labs(title = "Histogram of Sepal Length",
      x = "Sepal Length (cm)",
      y = "Frequency") +
  theme_minimal()
```



4)BOX PLOT

```
# Install ggplot2 (if not already installed)

install.packages("ggplot2")

# Load the ggplot2 package

library(ggplot2)

# Box plot of Sepal Length for each Species

ggplot(data = iris, aes(x = Species, y = Sepal.Length, fill = Species)) +

geom_boxplot() + # Adds box plot

labs(title = "Box Plot of Sepal Length by Species",

x = "Species",

y = "Sepal Length (cm)") +

theme_minimal()
```



```
ggplot(data = iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() + # Adds box plot
  labs(title = "Box Plot of Sepal Length by Species",
       x = "Species",
       y = "Sepal Length (cm)") +
  theme_minimal()
```