Progressive Education Society's

Modern College of Engineering, Pune

**MCA Department A.Y.2024-25**

# (410907) Big Data Analytics Laboratory

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Class: SY-MCA Shift / Div : S3 / A Roll Number : 52062

Name: Laxman Shinde Assignment No: 1 Date of Implementation: 04 / 09 / 2024

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Q 1.1 Implement a simple linear regression model to predict house prices based on their size.

**>>>**

library(ggplot2)

library(caret)

cat("Preparing the data...\n")

data <- data.frame(

Size = c(1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400),

Price = c(300000, 320000, 340000, 360000, 380000, 400000, 420000, 440000, 460000, 480000)

)

set.seed(42)

data <- data[sample(nrow(data)), ]

cat("Splitting the data into training and test sets...\n")

train\_index <- createDataPartition(data$Price, p = 0.5, list = FALSE) # 50% training, 50% test

train\_data <- data[train\_index, ]

test\_data <- data[-train\_index, ]

cat("Training data size: ", nrow(train\_data), "\n")

cat("Test data size: ", nrow(test\_data), "\n")

cat("Unique values in test data Price column:\n")

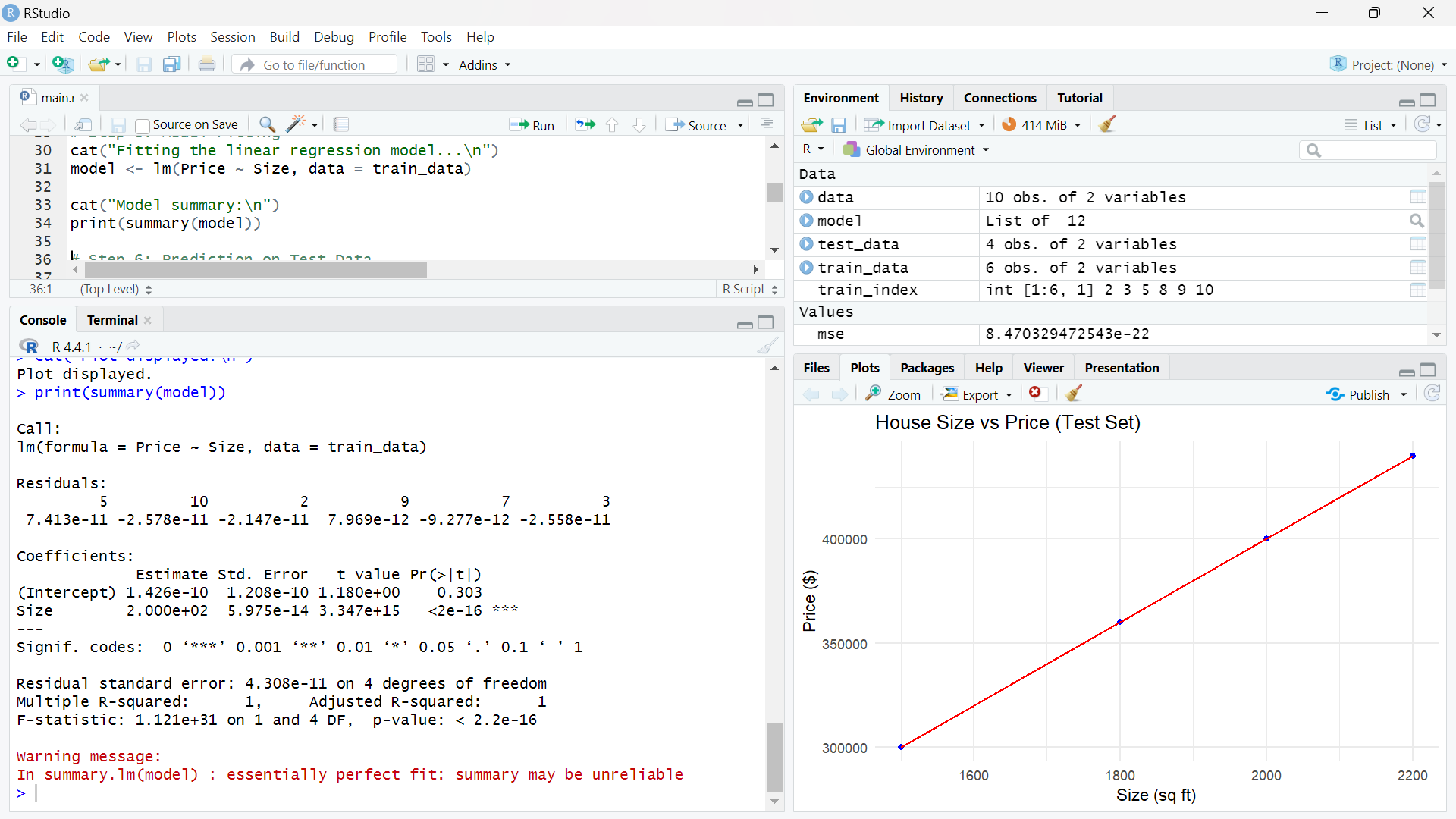
print(unique(test\_data$Price))

cat("Fitting the linear regression model...\n")

model <- lm(Price ~ Size, data = train\_data)

cat("Model summary:\n")

print(summary(model))



cat("Predicting house prices for the test data...\n")

y\_pred <- predict(model, newdata = test\_data)

cat("Predicted values:\n")

print(y\_pred)

cat("Actual values:\n")

print(test\_data$Price)

cat("Evaluating the model...\n")

cat("Test data values:\n")

print(test\_data)

test\_variance <- var(test\_data$Price, na.rm = TRUE)

cat("Variance in test data (Price):", test\_variance, "\n")

if (!is.na(test\_variance) && test\_variance > 0) {

mse <- mean((test\_data$Price - y\_pred)^2, na.rm = TRUE)

r\_squared <- 1 - (sum((test\_data$Price - y\_pred)^2, na.rm = TRUE) / sum((test\_data$Price - mean(test\_data$Price, na.rm = TRUE))^2, na.rm = TRUE))

cat("Mean Squared Error: ", mse, "\n")

cat("R-squared: ", r\_squared, "\n")

} else {

cat("Test data has zero or invalid variance, can't calculate MSE or R-squared.\n")

}

cat("Plotting actual vs predicted values...\n")

ggplot() +

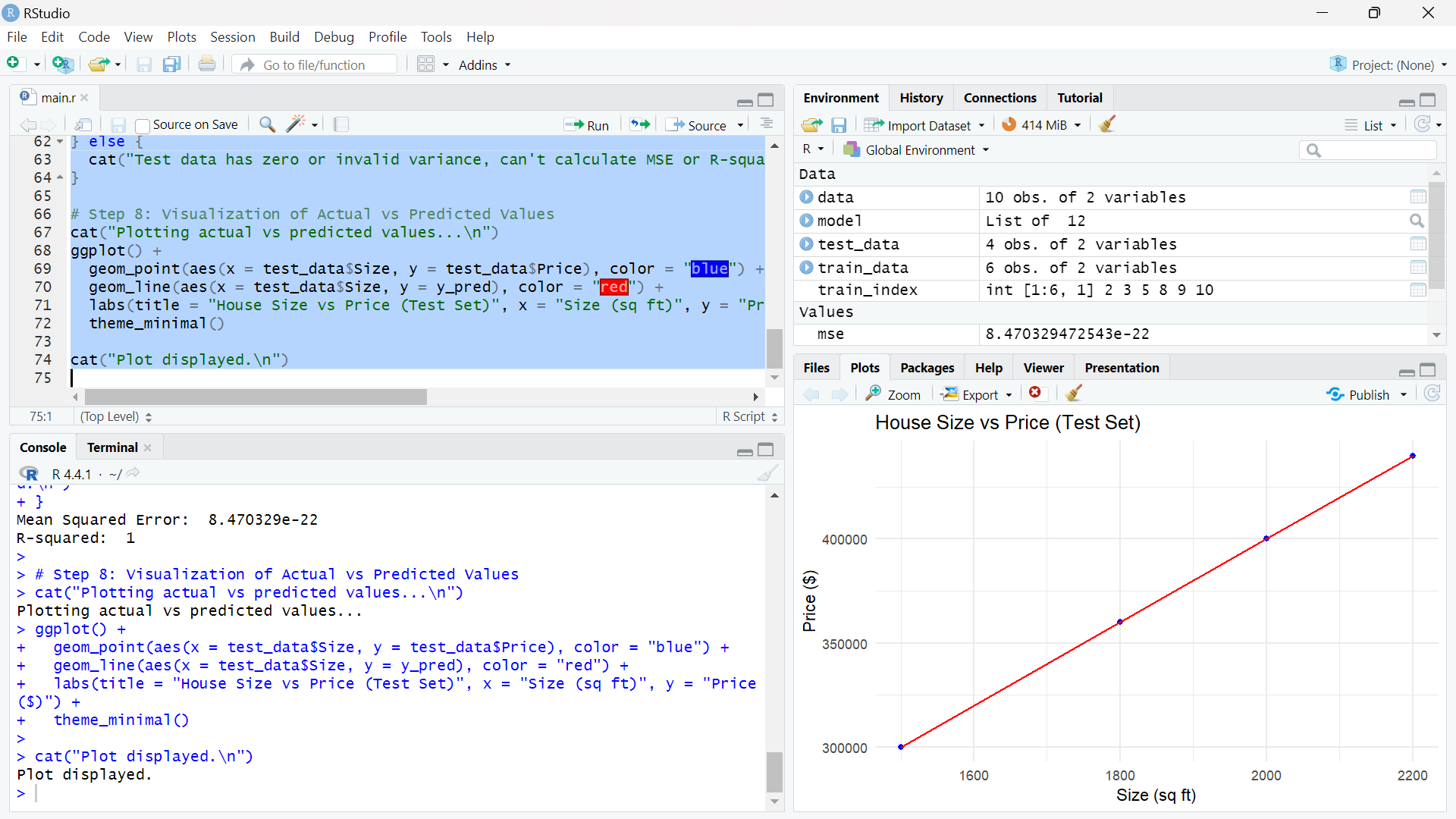
geom\_point(aes(x = test\_data$Size, y = test\_data$Price), color = "blue") +

geom\_line(aes(x = test\_data$Size, y = y\_pred), color = "red") +

labs(title = "House Size vs Price (Test Set)", x = "Size (sq ft)", y = "Price ($)") +

theme\_minimal()

cat("Plot displayed.\n")



# Q 1.2 Implement a logistic regression model to classify whether a student will pass or fail based on their study hours. Evaluate the model performance using accuracy.

**>>>**

library(caret)

library(ggplot2)

cat("Preparing the data...\n")

data <- data.frame(

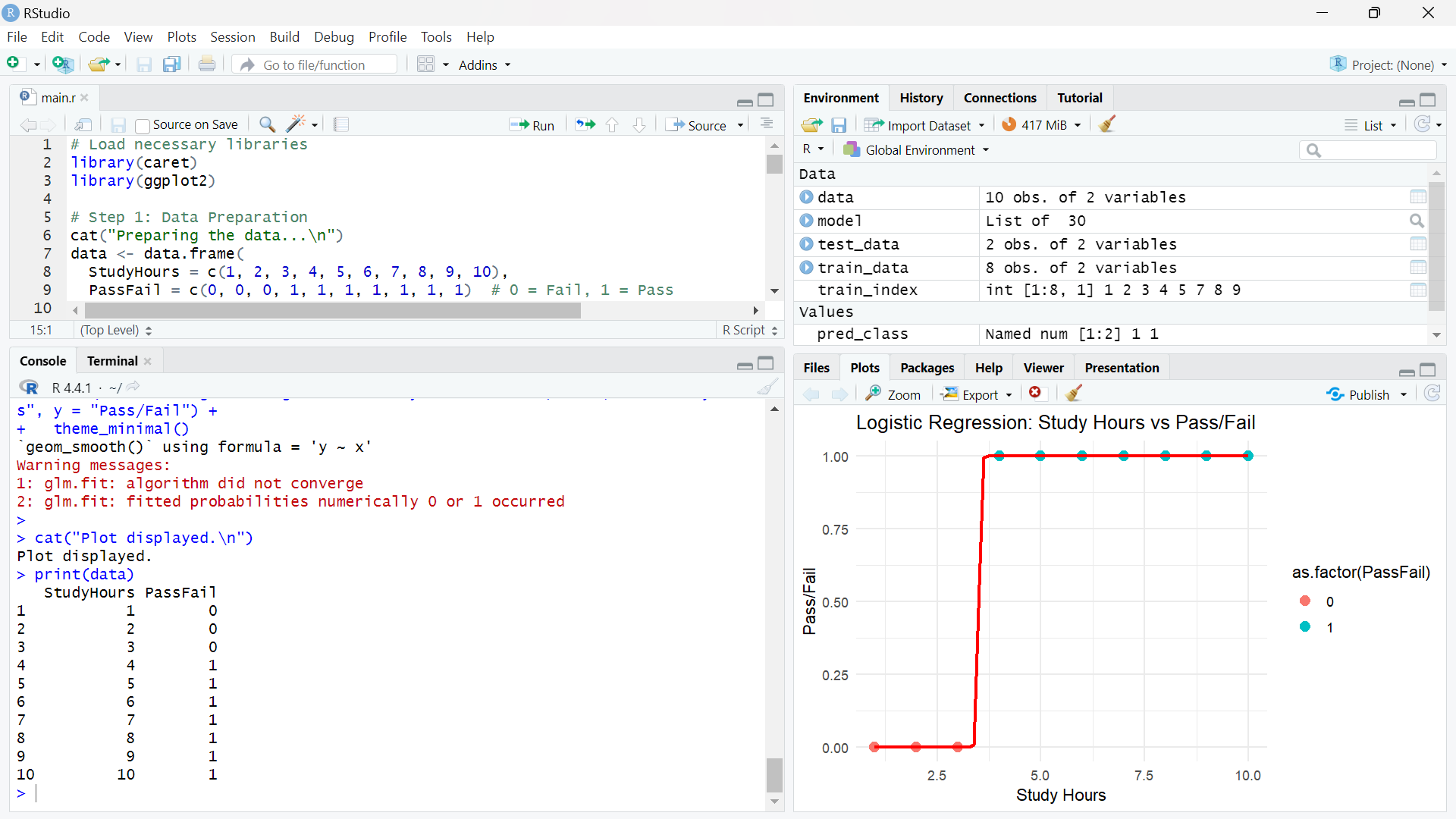
StudyHours = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),

PassFail = c(0, 0, 0, 1, 1, 1, 1, 1, 1, 1) # 0 = Fail, 1 = Pass

)

cat("Sample data:\n")

print(data)



cat("Splitting the data into training and test sets...\n")

set.seed(42) # For reproducibility

train\_index <- createDataPartition(data$PassFail, p = 0.7, list = FALSE) # 70% training, 30% test

train\_data <- data[train\_index, ]

test\_data <- data[-train\_index, ]

cat("Training data size: ", nrow(train\_data), "\n")

cat("Test data size: ", nrow(test\_data), "\n")

cat("Classes in test data:\n")

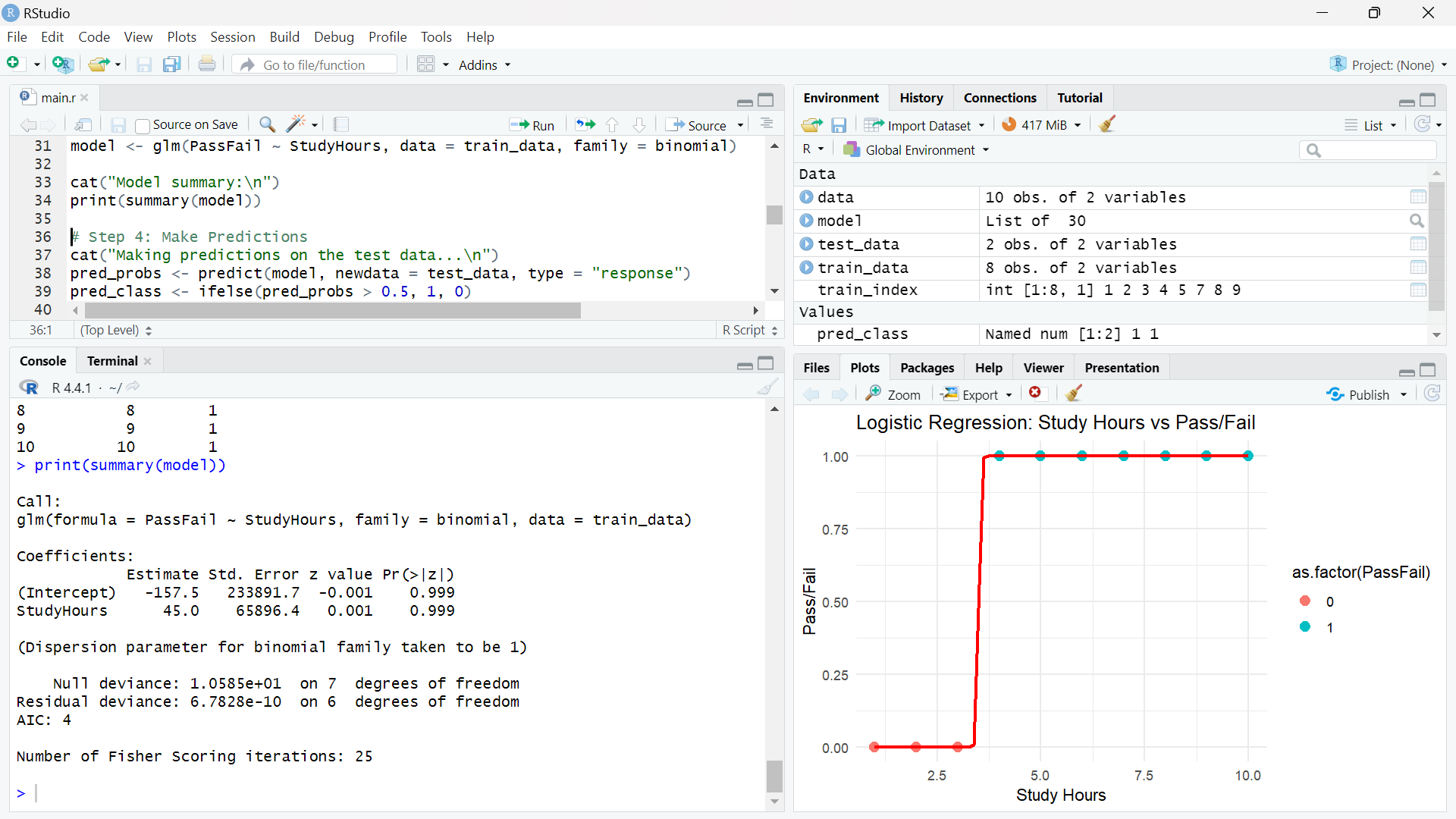
print(table(test\_data$PassFail))

cat("Fitting the logistic regression model...\n")

model <- glm(PassFail ~ StudyHours, data = train\_data, family = binomial)

cat("Model summary:\n")

print(summary(model))



cat("Making predictions on the test data...\n")

pred\_probs <- predict(model, newdata = test\_data, type = "response")

pred\_class <- ifelse(pred\_probs > 0.5, 1, 0)

cat("Predicted probabilities:\n")

print(pred\_probs)

cat("Predicted classes:\n")

print(pred\_class)

cat("Actual classes:\n")

print(test\_data$PassFail)

cat("Unique predicted classes:\n")

print(unique(pred\_class))

cat("Evaluating the model...\n")

if (length(unique(test\_data$PassFail)) > 1 && length(unique(pred\_class)) > 1) {

conf\_matrix <- confusionMatrix(as.factor(pred\_class), as.factor(test\_data$PassFail))

cat("Confusion Matrix:\n")

print(conf\_matrix)

cat("Accuracy:", conf\_matrix$overall['Accuracy'], "\n")

} else {

cat("Not enough class variability in the test set or predictions for evaluation.\n")

}

cat("Plotting the data with the decision boundary...\n")

ggplot(data, aes(x = StudyHours, y = PassFail)) +

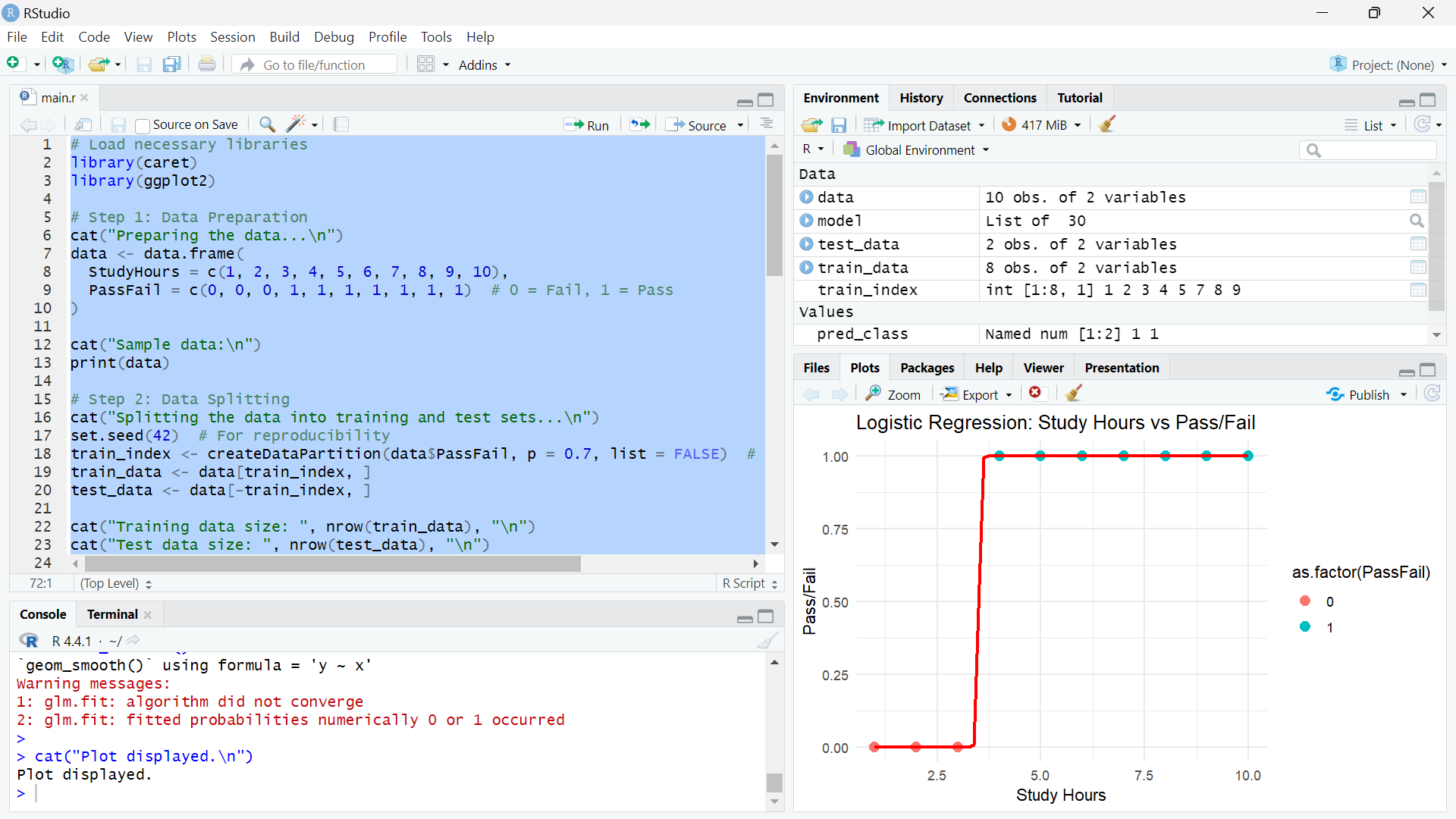
geom\_point(aes(color = as.factor(PassFail)), size = 3) +

geom\_smooth(method = "glm", method.args = list(family = "binomial"), color = "red", se = FALSE) +

labs(title = "Logistic Regression: Study Hours vs Pass/Fail", x = "Study Hours", y = "Pass/Fail") +

theme\_minimal()

cat("Plot displayed.\n")



**Q.1.3**. **iii.Implement an SVM model to classify the Iris dataset.**

if (!require(e1071)) install.packages("e1071", dependencies = TRUE)

if (!require(caret)) install.packages("caret", dependencies = TRUE)

if (!require(ggplot2)) install.packages("ggplot2", dependencies = TRUE)

library(e1071)

library(caret)

library(ggplot2)

cat("Loading the Iris dataset...\n")

data(iris)

set.seed(42)

train\_index <- createDataPartition(iris$Species, p = 0.7, list = FALSE)

train\_data <- iris[train\_index, ]

test\_data <- iris[-train\_index, ]

cat("Training data size: ", nrow(train\_data), "\n")

cat("Test data size: ", nrow(test\_data), "\n")

cat("Training the SVM model...\n")

svm\_model <- svm(Species ~ ., data = train\_data, kernel = "radial") # Radial kernel is default

cat("Model summary:\n")

print(summary(svm\_model))

cat("Making predictions on the test data...\n")

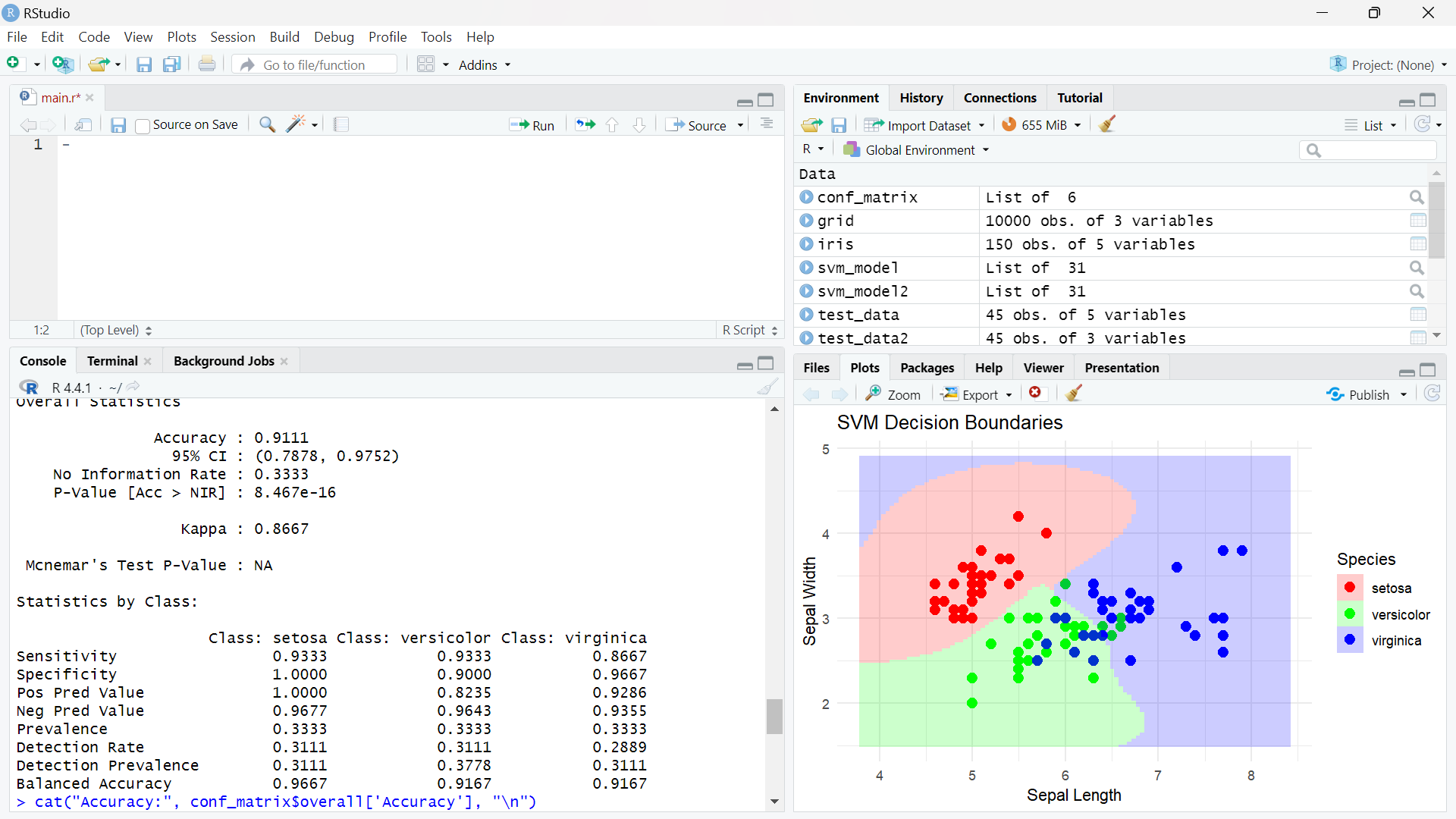
predictions <- predict(svm\_model, newdata = test\_data)

cat("Evaluating the model...\n")

conf\_matrix <- confusionMatrix(predictions, test\_data$Species)

cat("Confusion Matrix:\n")

print(conf\_matrix)



cat("Accuracy:", conf\_matrix$overall['Accuracy'], "\n")

cat("Visualizing decision boundaries...\n")

train\_data2 <- train\_data[, c("Sepal.Length", "Sepal.Width", "Species")]

test\_data2 <- test\_data[, c("Sepal.Length", "Sepal.Width", "Species")]

svm\_model2 <- svm(Species ~ ., data = train\_data2, kernel = "radial")

x\_seq <- seq(min(iris$Sepal.Length) - 0.5, max(iris$Sepal.Length) + 0.5, length.out = 100)

y\_seq <- seq(min(iris$Sepal.Width) - 0.5, max(iris$Sepal.Width) + 0.5, length.out = 100)

grid <- expand.grid(Sepal.Length = x\_seq, Sepal.Width = y\_seq)

grid$Species <- predict(svm\_model2, newdata = grid)

ggplot() +

geom\_point(data = train\_data2, aes(x = Sepal.Length, y = Sepal.Width, color = Species), size = 3) +

geom\_raster(data = grid, aes(x = Sepal.Length, y = Sepal.Width, fill = Species), alpha = 0.2) +

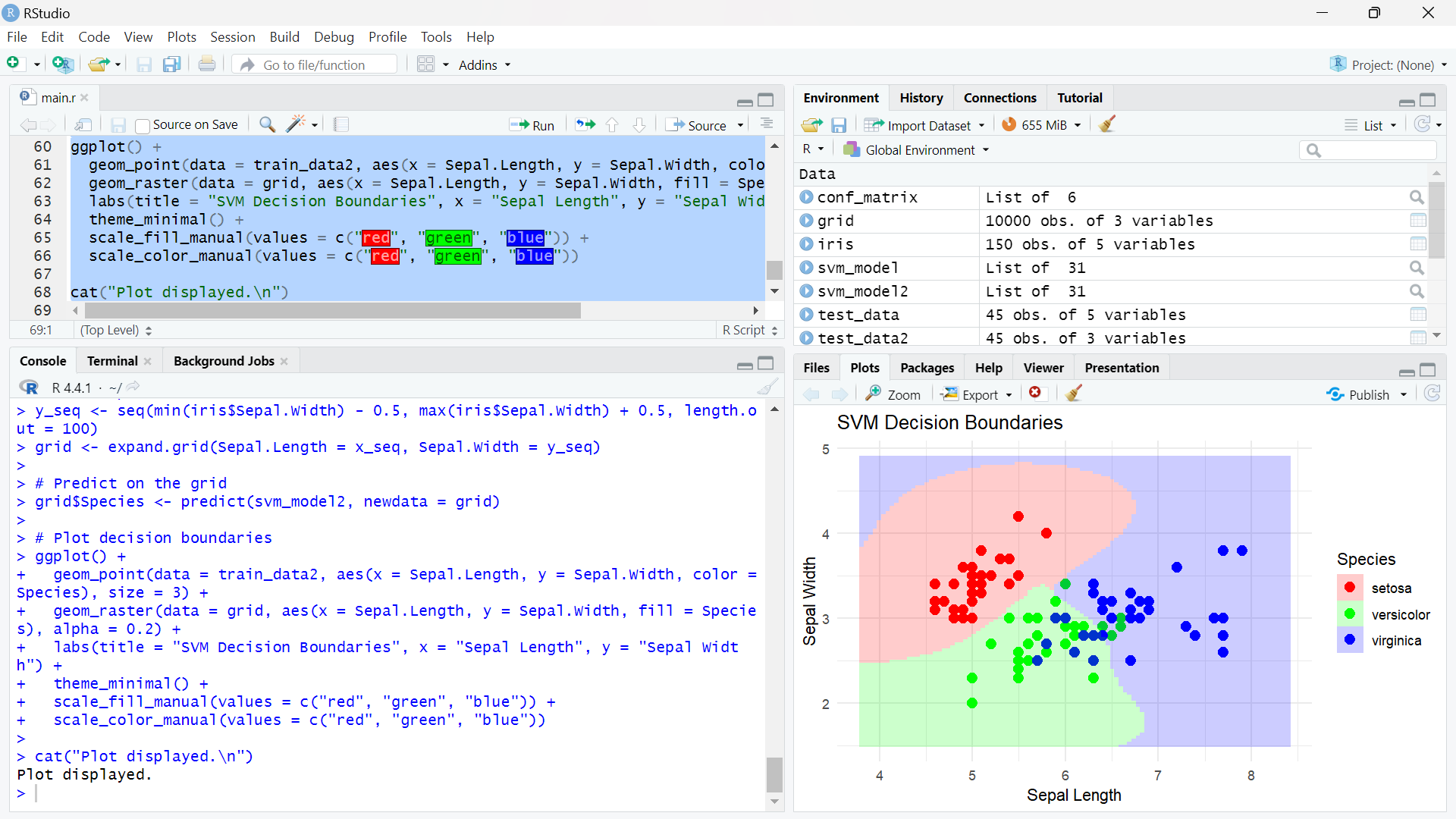
labs(title = "SVM Decision Boundaries", x = "Sepal Length", y = "Sepal Width") +

theme\_minimal() +

scale\_fill\_manual(values = c("red", "green", "blue")) +

scale\_color\_manual(values = c("red", "green", "blue"))

cat("Plot displayed.\n")



# Q 2.1 . Use the Apriori algorithm to find frequent item sets in a transaction dataset. Generate association rules and evaluate them using confidence and lift.

**>>>**

if (!require(arules)) install.packages("arules", dependencies = TRUE)

if (!require(arulesViz)) install.packages("arulesViz", dependencies = TRUE)

library(arules)

library(arulesViz)

cat("Creating sample transaction data...\n")

transaction\_list <- list(

c("milk", "bread"),

c("milk", "bread", "butter"),

c("bread", "butter"),

c("milk", "bread", "butter"),

c("bread", "butter", "eggs"),

c("milk", "eggs"),

c("milk", "bread", "butter", "eggs")

)

transactions <- as(transaction\_list, "transactions")

cat("Sample transactions:\n")

inspect(transactions)

cat("Applying the Apriori algorithm...\n")

frequent\_itemsets <- apriori(transactions, parameter = list(support = 0.3, target = "frequent itemsets"))

cat("Frequent itemsets:\n")

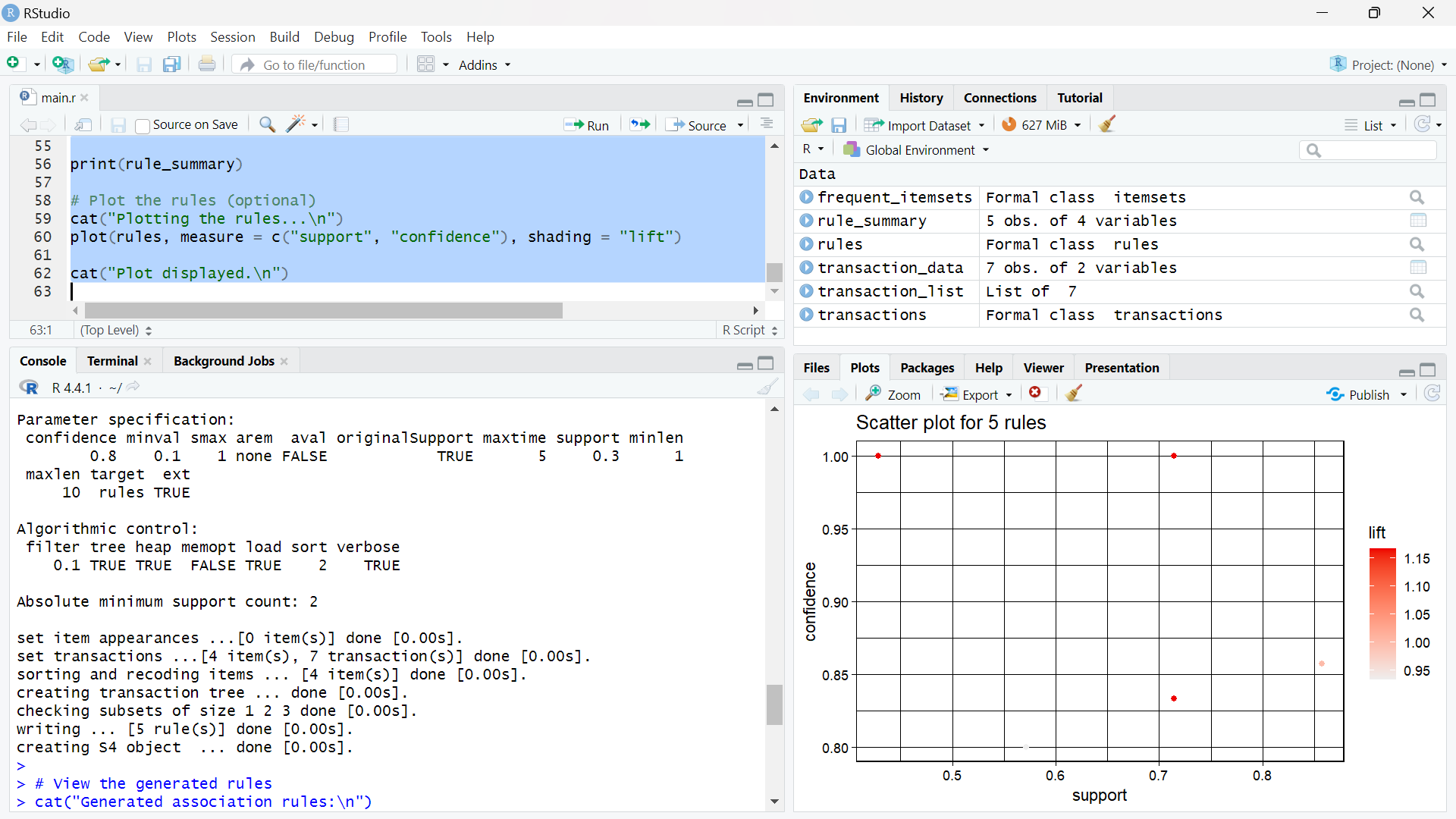
inspect(frequent\_itemsets)

cat("Generating association rules...\n")

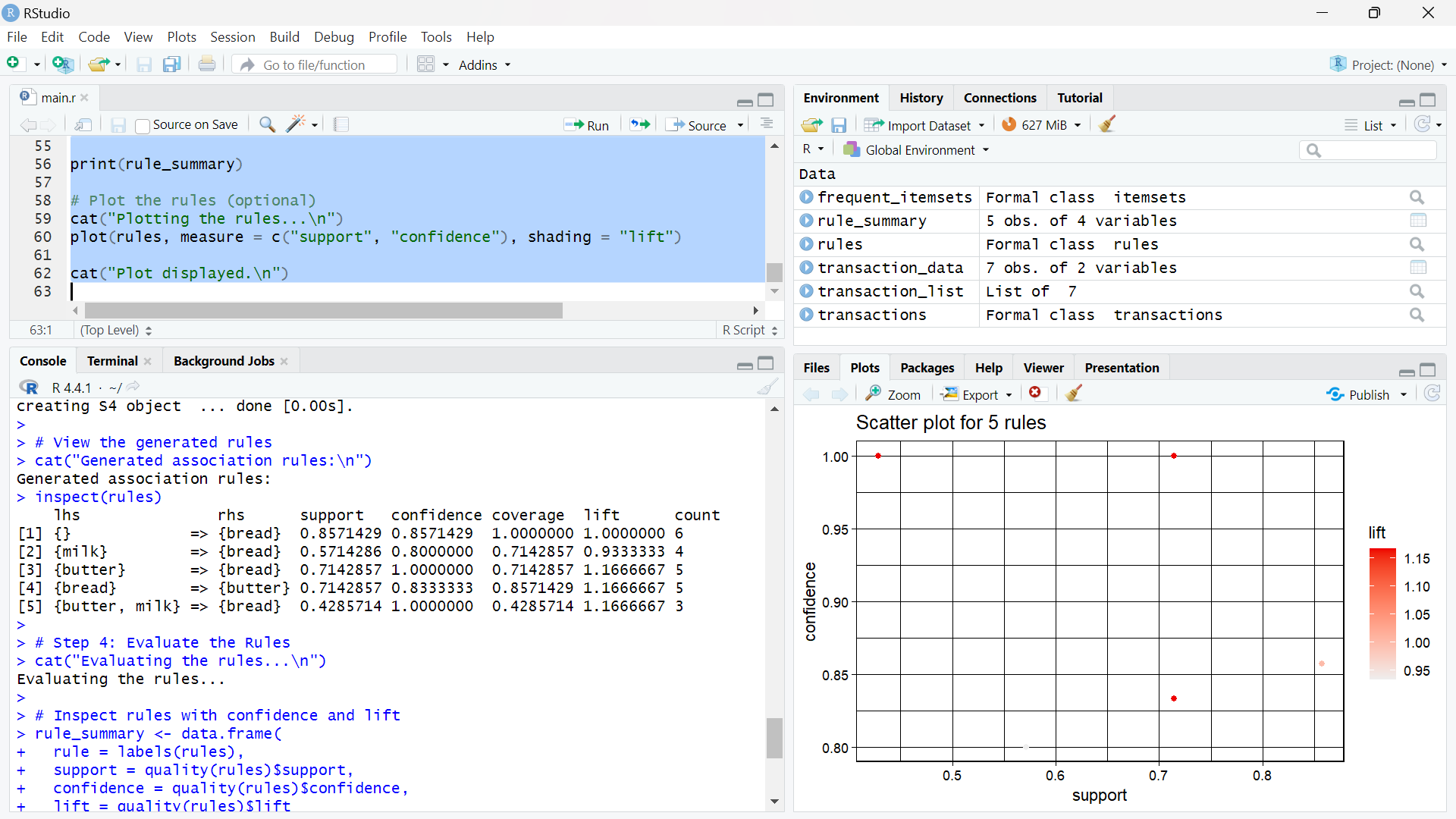
rules <- apriori(transactions, parameter = list(support = 0.3, confidence = 0.8, target = "rules"))

cat("Generated association rules:\n")

inspect(rules)



cat("Evaluating the rules...\n")



rule\_summary <- data.frame(

rule = labels(rules),

support = quality(rules)$support,

confidence = quality(rules)$confidence,

lift = quality(rules)$lift

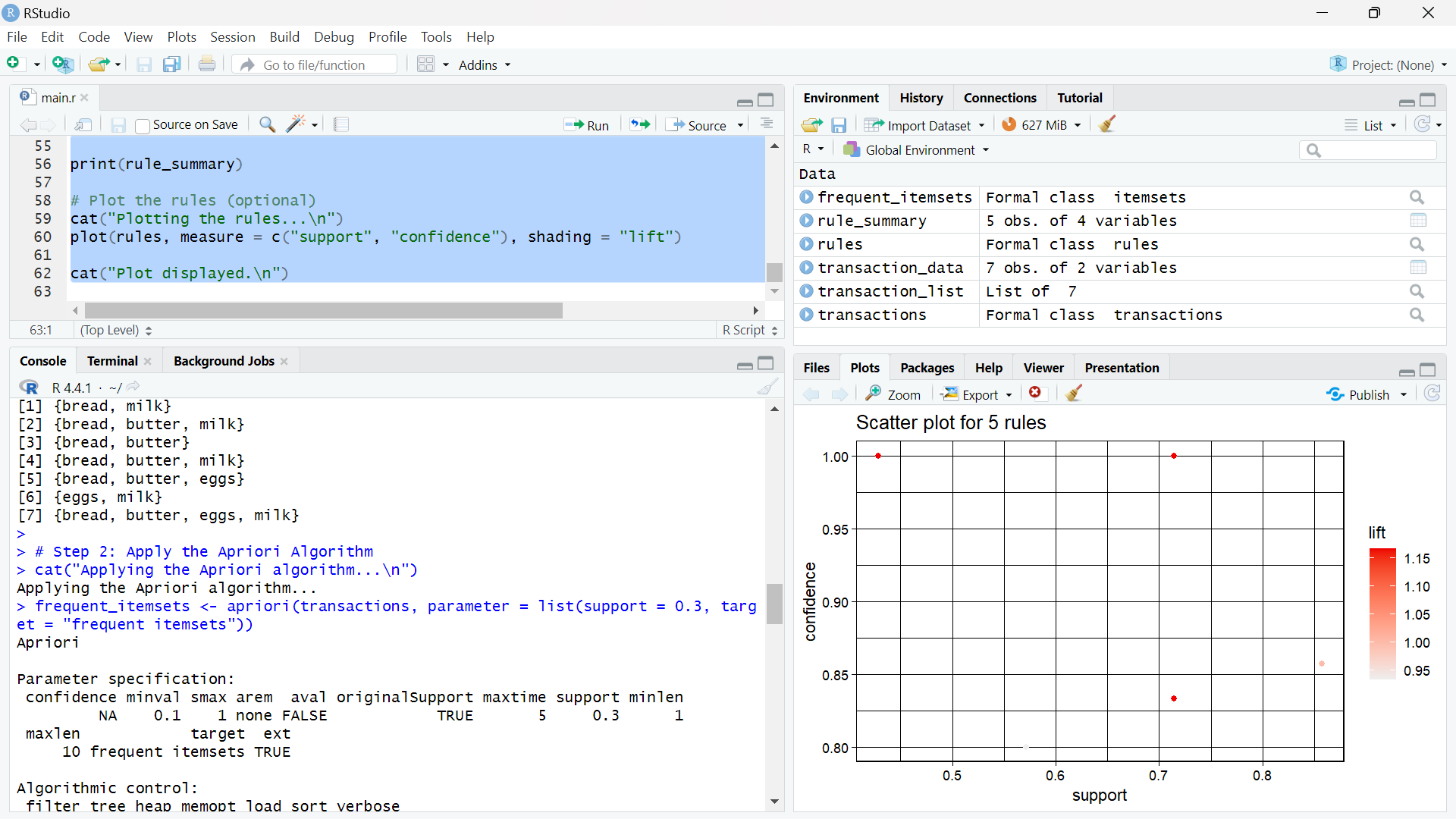
)

print(rule\_summary)

cat("Plotting the rules...\n")

plot(rules, measure = c("support", "confidence"), shading = "lift")

cat("Plot displayed.\n")



# Q 2.2. Implement k-means clustering on a dataset. Visualize the clusters.

**>>>**

if (!require(ggplot2)) install.packages("ggplot2", dependencies = TRUE)

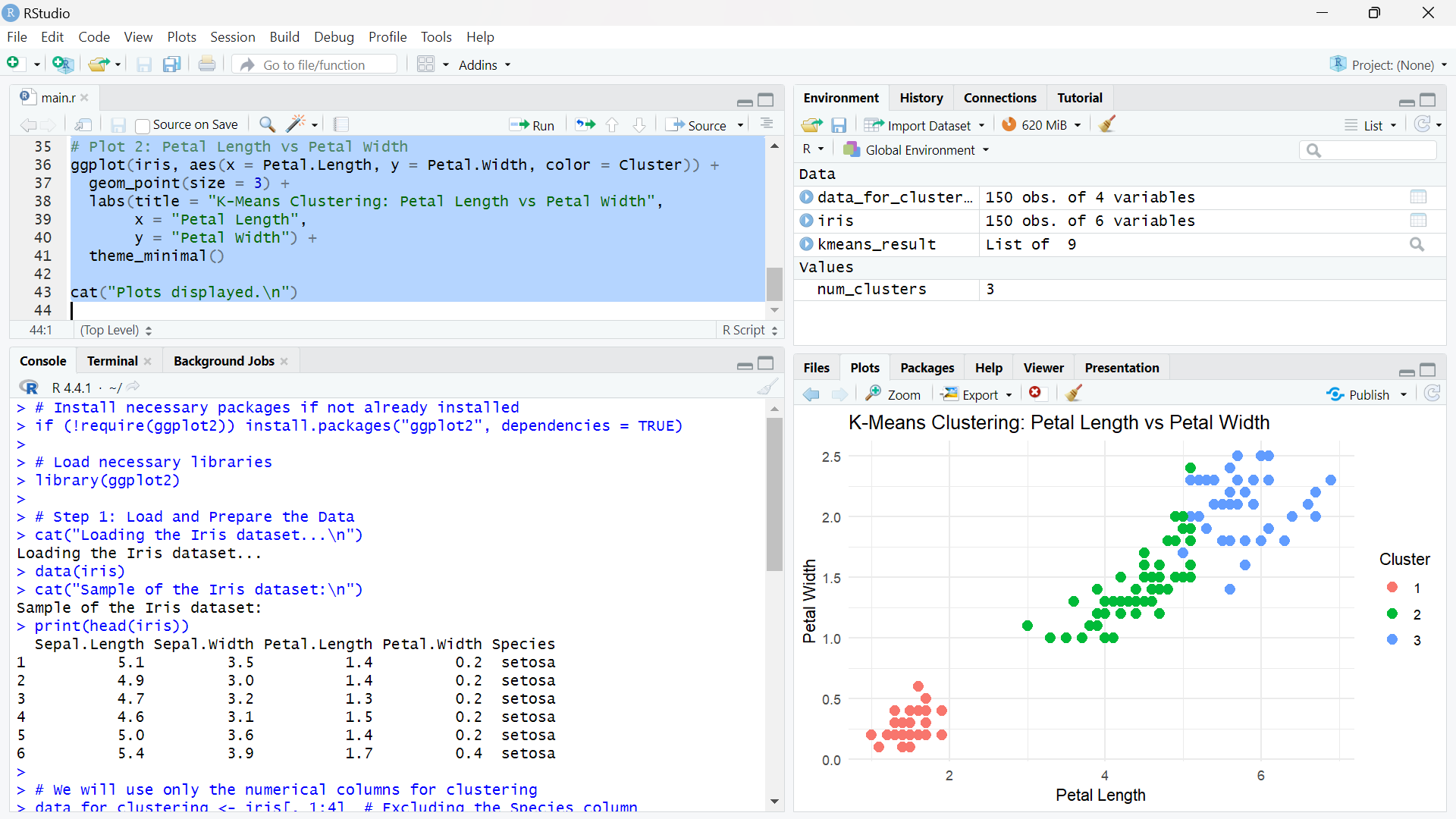
library(ggplot2)

cat("Loading the Iris dataset...\n")

data(iris)

cat("Sample of the Iris dataset:\n")

print(head(iris))



data\_for\_clustering <- iris[, 1:4]

set.seed(42)

num\_clusters <- 3

kmeans\_result <- kmeans(data\_for\_clustering, centers = num\_clusters)

iris$Cluster <- as.factor(kmeans\_result$cluster)

cat("Visualizing the clusters...\n")

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

geom\_point(size = 3) +

labs(title = "K-Means Clustering: Sepal Length vs Sepal Width",

x = "Sepal Length",

y = "Sepal Width") +

theme\_minimal()

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

geom\_point(size = 3) +

labs(title = "K-Means Clustering: Petal Length vs Petal Width",

x = "Petal Length",

y = "Petal Width") +

theme\_minimal()

cat("Plots displayed.\n")

