A close-up of a school of engineering

Description automatically generated

**LAB PROGRAMS (41-60)**

**ON**

**ITA0402-Statistics with R Programming for Data Visualization**

**SLOT B**

**Submitted by**

**192321093 – MURALI KRISHNAN.S**

**To**

**DR.P. Jesu Jayarin**

**Saveetha School of Engineering**

**SIMATS, Thandalam.**

41. Aim: To create a data frame in R with Name, Age, and Gender, and print it.

Algorithm:

1. Create vectors for Name, Age, and Gender.
2. Combine these vectors into a data frame using the data.frame() function.
3. Print the data frame using the print() function.

Code:

data <- data.frame(

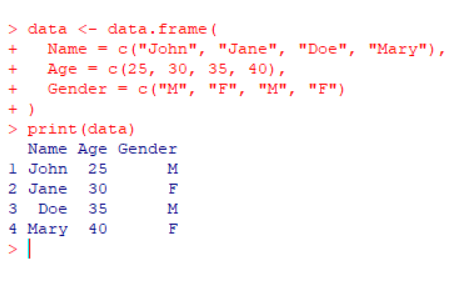
Name = c("John", "Jane", "Doe", "Mary"),

Age = c(25, 30, 35, 40),

Gender = c("M", "F", "M", "F")

)

print(data)



42. Aim: To create a data frame with Age and Height, and extract the Height column as a vector.

Algorithm:

1. Create vectors for Age and Height.
2. Combine these vectors into a data frame.
3. Extract the Height column as a vector using the $ operator.
4. Print the extracted vector.

Code:

df <- data.frame(

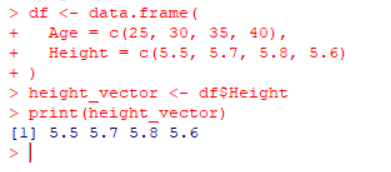
Age = c(25, 30, 35, 40),

Height = c(5.5, 5.7, 5.8, 5.6)

)

height\_vector <- df$Height

print(height\_vector)



43. Aim: To melt a data frame from wide format to long format.

Algorithm:

1. Load the reshape2 package.
2. Create a data frame with columns for region and sales in multiple quarters (Q1, Q2, Q3, Q4).
3. Use the melt() function from the reshape2 package to reshape the data frame to long format.
4. Print the melted data frame.

Code:

install.packages("reshape2")

library(reshape2)

sale <- data.frame(

region = c("North", "South", "East", "West"),

Q1 = c(100, 200, 150, 130),

Q2 = c(120, 210, 160, 140),

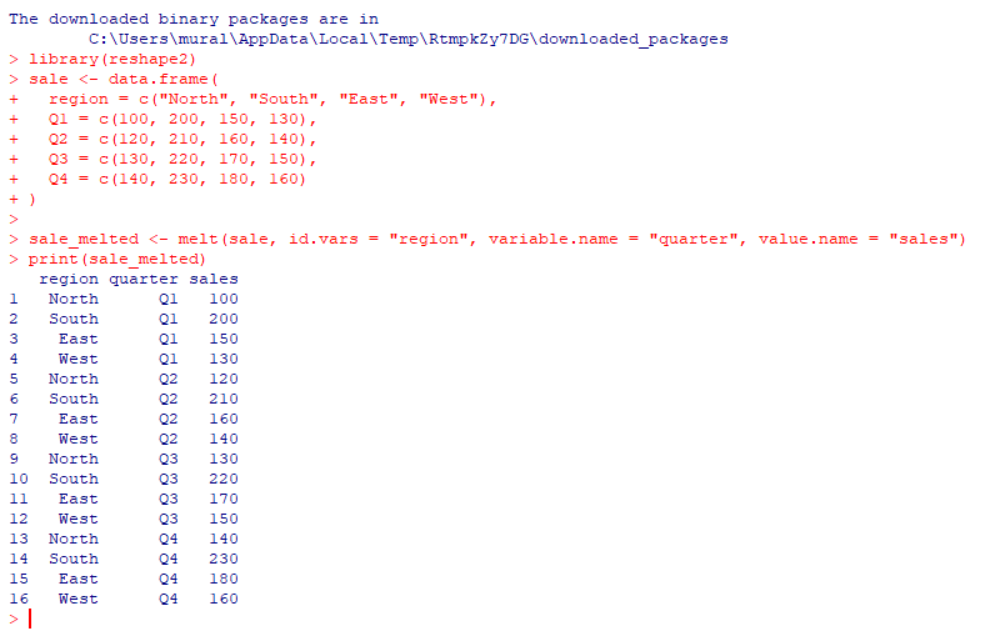
Q3 = c(130, 220, 170, 150),

Q4 = c(140, 230, 180, 160)

)

sale\_melted <- melt(sale, id.vars = "region", variable.name = "quarter", value.name = "sales")

print(sale\_melted)



44. Aim: To manipulate the Air Quality dataset by ordering and removing columns.

Algorithm:

1. Check if the airquality dataset is a data frame.
2. Order the data frame by the first two columns (Ozone and Solar.R).
3. Remove the columns Solar.R and Wind from the dataset.
4. Display the modified data frame.

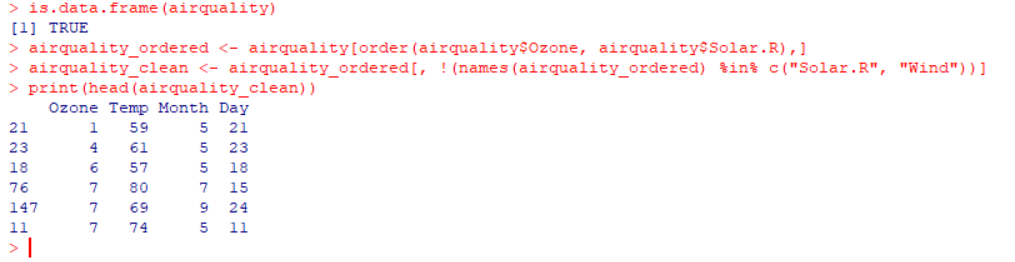
Code:

is.data.frame(airquality)

airquality\_ordered <- airquality[order(airquality$Ozone, airquality$Solar.R),]

airquality\_clean <- airquality\_ordered[, !(names(airquality\_ordered) %in% c("Solar.R", "Wind"))]

print(head(airquality\_clean))



45. Aim: To create a factor based on the height of women.

Algorithm:

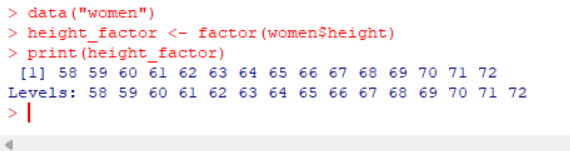
1. Load the built-in women dataset.
2. Convert the height data into a factor using the factor() function.
3. Print the factor.

Code:

data("women")

height\_factor <- factor(women$height)

print(height\_factor)



46. Aim: To perform exploratory data analysis (EDA) on the Iris dataset.

Algorithm:

1. Use built-in iris dataset.
2. Compute the dimension, structure, summary statistics, and standard deviation of the dataset.
3. Compute mean and standard deviation of features grouped by species.
4. Calculate the quantiles for Sepal.Width and Sepal.Length.
5. Create a new column Sepal.Length.Cate based on quantiles.
6. Compute averages of numerical variables based on species and Sepal.Length.Cate.
7. Create a pivot table using the dcast() function.

Code:

dim(iris)

str(iris)

summary(iris)

sd(iris[,1:4]) # Standard deviation of features

aggregate(iris[,1:4], by = list(iris$Species), FUN = mean)

aggregate(iris[,1:4], by = list(iris$Species), FUN = sd)

quantile(iris$Sepal.Width)

quantile(iris$Sepal.Length)

iris$Sepal.Length.Cate <- cut(iris$Sepal.Length, breaks = 3)

head(iris)

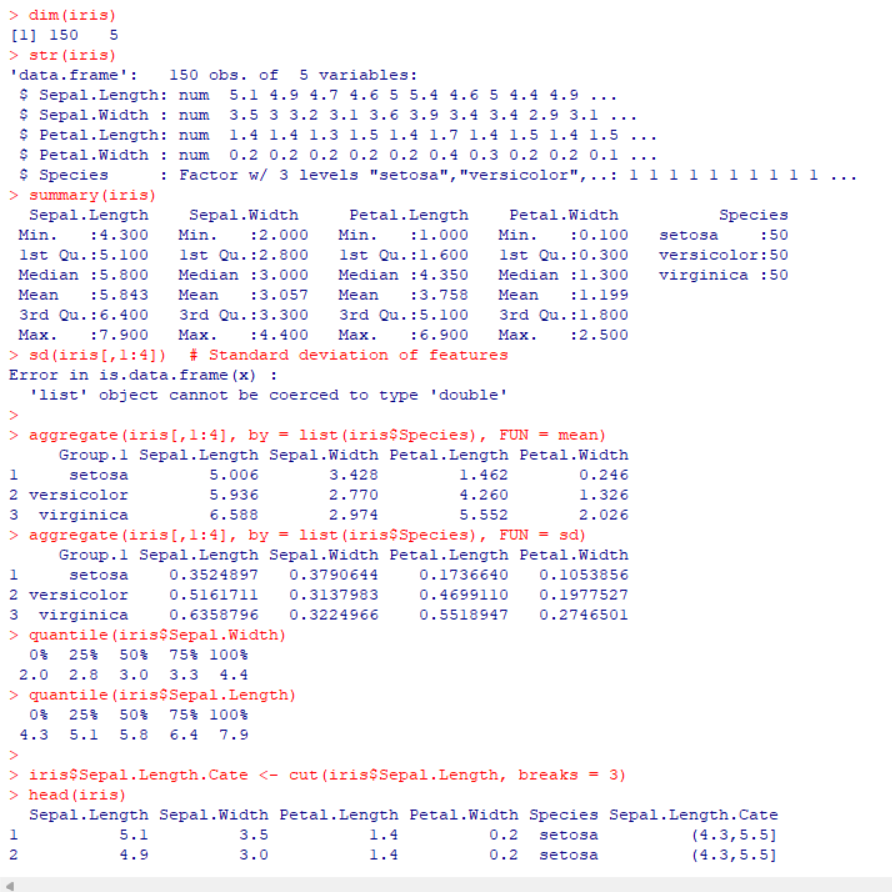
aggregate(iris[,1:4], by = list(iris$Species, iris$Sepal.Length.Cate), FUN = mean)

aggregate(iris[,1:4], by = list(iris$Species, iris$Sepal.Length.Cate), FUN = mean)

library(reshape2)

pivot\_table <- dcast(iris, Sepal.Length.Cate ~ Species, value.var = "Sepal.Width")

print(pivot\_table)



47. Aim: To perform logistic regression on the Iris dataset with training and testing splits.

Algorithm:

1. Split the iris dataset into training (80%) and testing (20%) datasets.
2. Build a logistic regression model using features Petal.Length and Petal.Width to predict Species.
3. Use the model to predict species on the test data and calculate probabilities.
4. Create a confusion matrix comparing the predicted species to actual species.

Code:

set.seed(123)

sample\_index <- sample(1:nrow(iris), 0.8 \* nrow(iris))

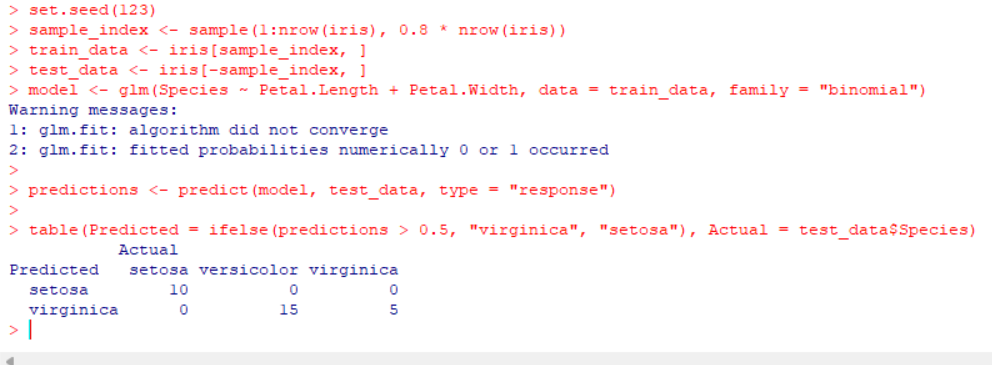
train\_data <- iris[sample\_index, ]

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

model <- glm(Species ~ Petal.Length + Petal.Width, data = train\_data, family = "binomial")

predictions <- predict(model, test\_data, type = "response")

table(Predicted = ifelse(predictions > 0.5, "virginica", "setosa"), Actual = test\_data$Species)



48. Aim: To explore the Air Quality dataset by performing various computations.

Algorithm:

1. Compute the mean temperature without using the built-in mean() function.
2. Extract the first five rows of the dataset.
3. Remove the Temp and Wind columns.
4. Find the coldest day based on the Temp column.
5. Count how many days the wind speed was greater than 17 mph.

Code:

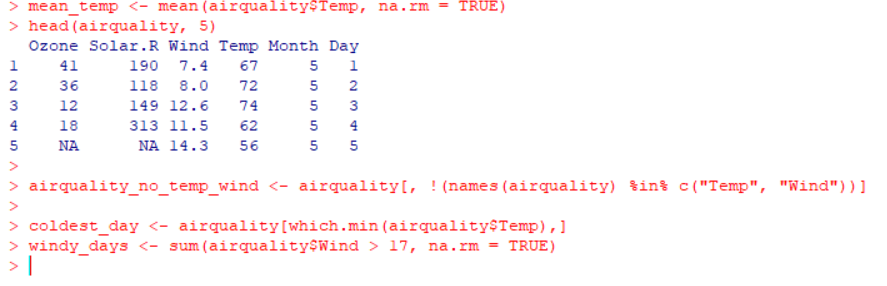
mean\_temp <- mean(airquality$Temp, na.rm = TRUE)

head(airquality, 5)

airquality\_no\_temp\_wind <- airquality[, !(names(airquality) %in% c("Temp", "Wind"))]

coldest\_day <- airquality[which.min(airquality$Temp),]

windy\_days <- sum(airquality$Wind > 17, na.rm = TRUE)



49. Aim: To create a multiple regression model predicting the weight of chickens based on Time and Diet.

Algorithm:

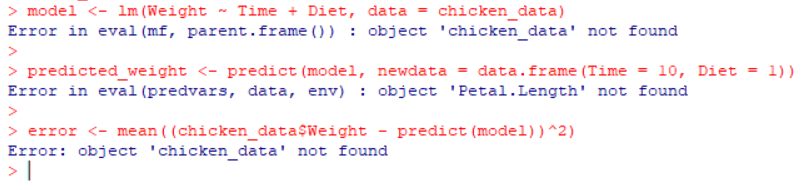
1. Create a data frame with variables Time, Diet, and Weight.
2. Build a multiple regression model with Weight as the dependent variable and Time and Diet as predictors.
3. Predict the weight for Time = 10 and Diet = 1.
4. Calculate the error for the model by computing the mean squared error.

Code:

model <- lm(Weight ~ Time + Diet, data = chicken\_data)

predicted\_weight <- predict(model, newdata = data.frame(Time = 10, Diet = 1))

error <- mean((chicken\_data$Weight - predict(model))^2)



50. Aim: To visualize data from the Titanic dataset.

Algorithm:

1. Create a bar chart of the Survived variable based on passengerClass.
2. Modify the bar chart to show Survived based on gender.
3. Create a histogram for the distribution of the Age variable.

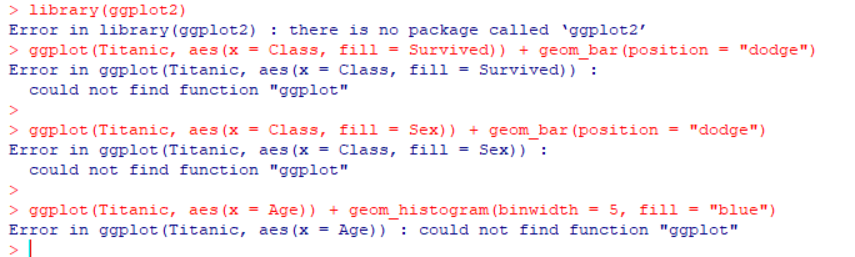
Code:

library(ggplot2)

ggplot(Titanic, aes(x = Class, fill = Survived)) + geom\_bar(position = "dodge")

ggplot(Titanic, aes(x = Class, fill = Sex)) + geom\_bar(position = "dodge")

ggplot(Titanic, aes(x = Age)) + geom\_histogram(binwidth = 5, fill = "blue")



51. Aim: To find quartiles of a dataset and draw a box plot.

Algorithm:

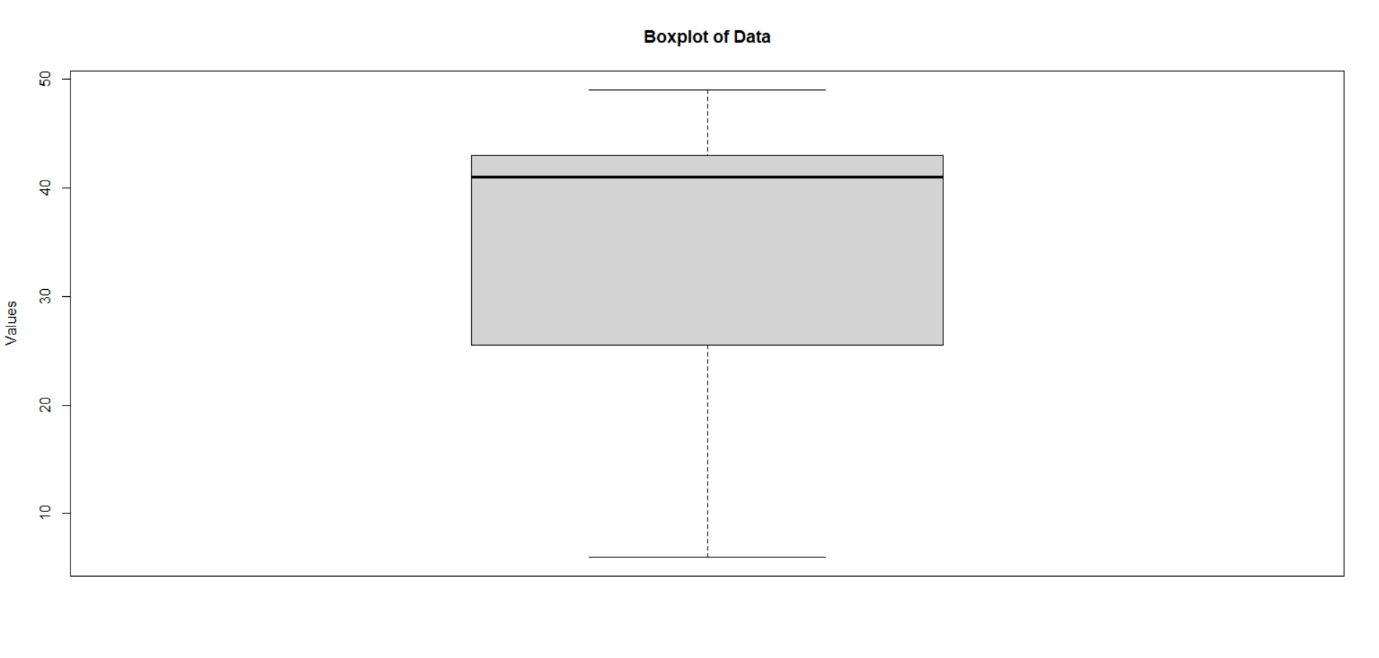
1. Find the quartiles of a given dataset using the quantile() function.
2. Draw a box plot for the given dataset using the boxplot() function.

Code:

data <- c(6, 47, 49, 15, 43, 41, 7, 39, 43, 41, 36)

quantile(data)

boxplot(data, main = "Boxplot of Data", ylab = "Values")



52. Aim: To analyze the contents of a CSV file and extract specific details.

Algorithm:

1. Read the CSV file into a data frame.
2. Find the maximum salary from the data.
3. Extract details of the person with the maximum salary.
4. Filter out all people working in the IT department.
5. Find IT department employees whose salary is greater than 600.

Code:

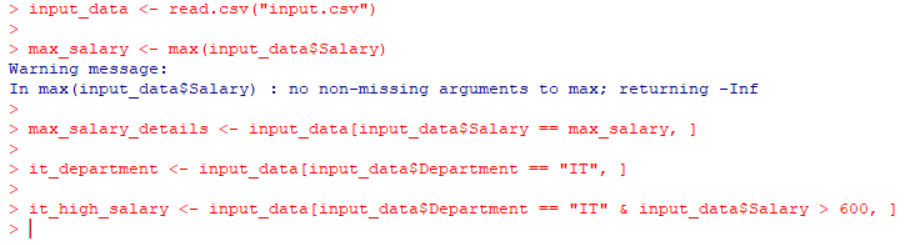
input\_data <- read.csv("input.csv")

max\_salary <- max(input\_data$Salary)

max\_salary\_details <- input\_data[input\_data$Salary == max\_salary, ]

it\_department <- input\_data[input\_data$Department == "IT", ]

it\_high\_salary <- input\_data[input\_data$Department == "IT" & input\_data$Salary > 600, ]



53. Aim: To merge two data frames (student personal information and scores).

Algorithm:

1. Create two data frames: one for student personal information and the other for student scores.
2. Merge the data frames based on the common StudentID column.
3. Print the merged data frame.

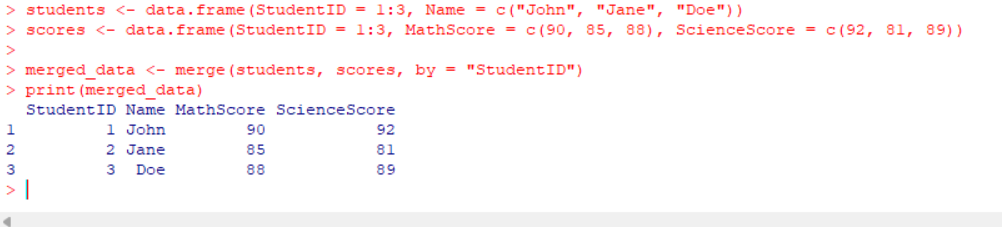
Code:

students <- data.frame(StudentID = 1:3, Name = c("John", "Jane", "Doe"))

scores <- data.frame(StudentID = 1:3, MathScore = c(90, 85, 88), ScienceScore = c(92, 81, 89))

merged\_data <- merge(students, scores, by = "StudentID")

print(merged\_data)



54. Aim: To visualize the Titanic dataset by passenger class, gender, and age.

Algorithm:

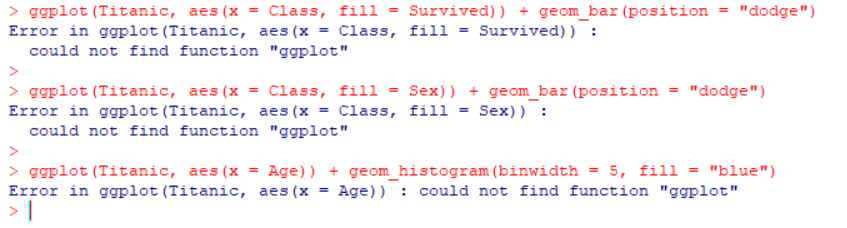
1. Create a bar chart showing the details of Survived by passengerClass.
2. Modify the bar chart to break down the survival count by gender.
3. Create a histogram for the distribution of the Age variable.

Code:

ggplot(Titanic, aes(x = Class, fill = Survived)) + geom\_bar(position = "dodge")

ggplot(Titanic, aes(x = Class, fill = Sex)) + geom\_bar(position = "dodge")

ggplot(Titanic, aes(x = Age)) + geom\_histogram(binwidth = 5, fill = "blue")



55. Aim: To create a product dataset and compute the average price.

Algorithm:

1. Create a data frame for product details including name, price, and quantity.
2. Calculate and print the average price of all products in the dataset.

Code:

product\_data <- data.frame(

Name = c("Product A", "Product B", "Product C"),

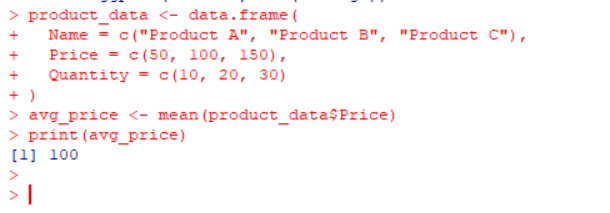
Price = c(50, 100, 150),

Quantity = c(10, 20, 30)

)

avg\_price <- mean(product\_data$Price)

print(avg\_price)



56. Aim: To merge customer and purchase data frames based on a common customer ID.

Algorithm:

1. Create two data frames: one for customer details and another for purchase history.
2. Merge the data frames based on the CustomerID column.
3. Display the merged data.

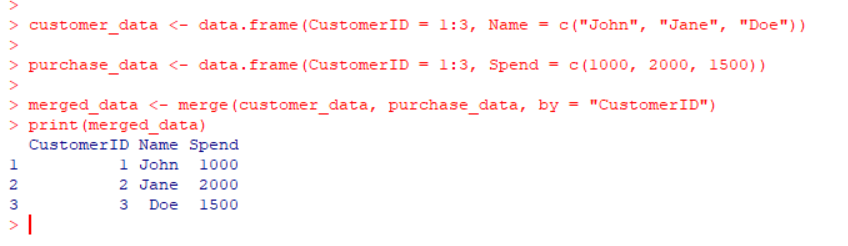
Code:

customer\_data <- data.frame(CustomerID = 1:3, Name = c("John", "Jane", "Doe"))

purchase\_data <- data.frame(CustomerID = 1:3, Spend = c(1000, 2000, 1500))

merged\_data <- merge(customer\_data, purchase\_data, by = "CustomerID")

print(merged\_data)



57. Aim: To create a regression model that predicts sales based on advertising spends.

Algorithm:

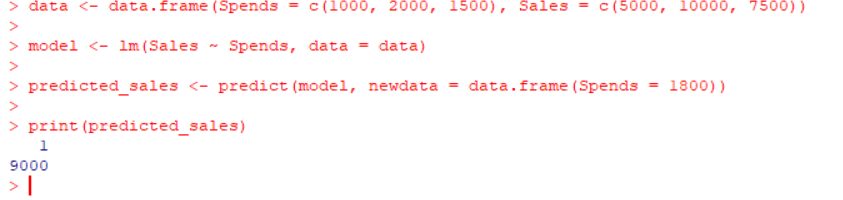
1. Create a data frame with columns Spends and Sales.
2. Build a linear regression model where Sales is the dependent variable and Spends is the independent variable.
3. Use the model to predict sales for a new value of Spends.

Code:

data <- data.frame(Spends = c(1000, 2000, 1500), Sales = c(5000, 10000, 7500))

model <- lm(Sales ~ Spends, data = data)

predicted\_sales <- predict(model, newdata = data.frame(Spends = 1800))



58. Aim: To perform linear regression between height and weight.

Algorithm:

1. Create vectors for Height and Weight.
2. Fit a linear regression model to predict Weight from Height.
3. Predict the weight for a given height of 170 cm.
4. Visualize the regression line by plotting the data and the regression model.

Code:

height <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

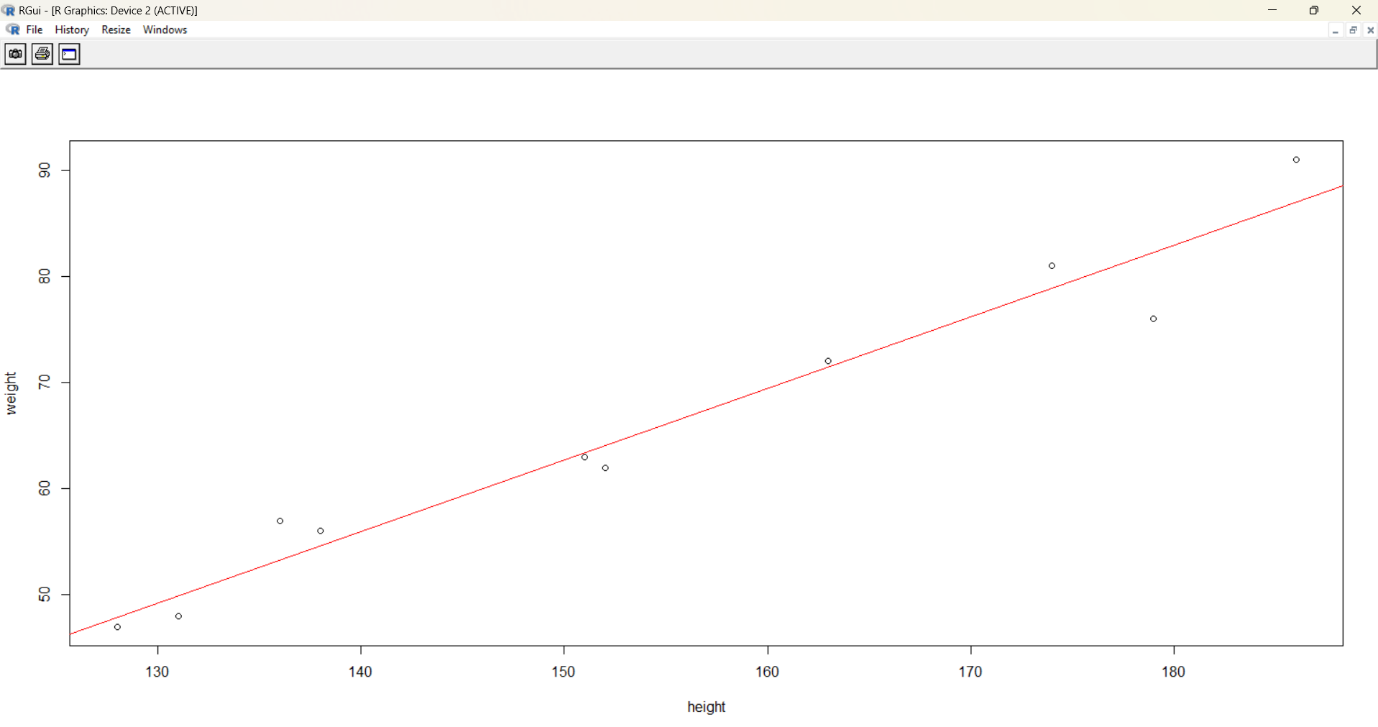
weight <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

model <- lm(weight ~ height)

predicted\_weight <- predict(model, newdata = data.frame(height = 170))

plot(height, weight)

abline(model, col = "red")



59. Aim: To calculate and interpret the mean, median, and mode of employee salaries.

Algorithm:

1. Create a data frame containing salary values.
2. Compute the mean, median, and mode of the salary data.
3. Print and interpret the results.

Code:

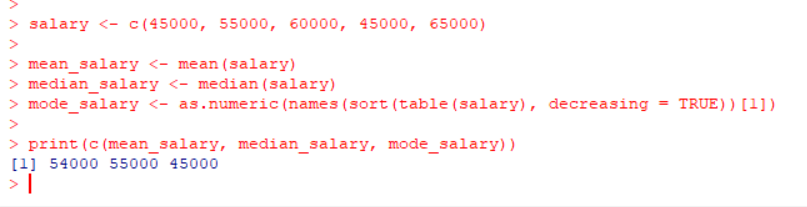
salary <- c(45000, 55000, 60000, 45000, 65000)

mean\_salary <- mean(salary)

median\_salary <- median(salary)

mode\_salary <- as.numeric(names(sort(table(salary), decreasing = TRUE))[1])

print(c(mean\_salary, median\_salary, mode\_salary))



60. Aim: To perform univariate exploratory data analysis (EDA) on daily temperature data.

Algorithm:

1. Create a vector of daily temperature data.
2. Calculate the mean, median, and mode of the temperature values.
3. Compute the range, interquartile range (IQR), and standard deviation of the temperatures.
4. Print the results of the EDA.

Code:

daily\_temps <- c(25, 28, 30, 29, 27, 26, 25, 24, 22, 23, 27, 28, 29, 30, 26, 27, 25, 24, 26, 28, 29, 30, 31, 32, 30, 29, 28, 27, 26, 25)

mean\_temp <- mean(daily\_temps)

median\_temp <- median(daily\_temps)

range\_temp <- range(daily\_temps)

iqr\_temp <- IQR(daily\_temps)

sd\_temp <- sd(daily\_temps)

print(c(mean\_temp, median\_temp, range\_temp, iqr\_temp, sd\_temp))

