Task 1: Check if a number is divisible by both 2 and 3 and Calculate Sum, Average, Minimum, and Maxim um of First 10 Natural Numbers

R

Copy code

# Check if a number is divisible by both 2 and 3 number <- 6

if (number %% 2 == 0 & number %% 3 == 0) { print("Number is divisible by both 2 and 3")

} else {

print("Number is not divisible by both 2 and 3")

}

# Calculate sum, average, minimum, and maximum of the first 10 natural numbers numbers <- 1:10

sum\_numbers <- sum(numbers) average <- mean(numbers) minimum <- min(numbers) maximum <- max(numbers)

print(paste("Sum:", sum\_numbers)) print(paste("Average:", average)) print(paste("Minimum:", minimum)) print(paste("Maximum:", maximum))

Task 2: Combine Multiple Logical Conditions and Calculate Sum, Average, Minimum, and Maximum of th e First 10 Natural Numbers using a For Loop

R

Copy code

# Combine logical conditions and perform calculations numbers <- 1:10

result <- c()

for (num in numbers) {

if (num %% 2 == 0 & num %% 3 == 0) { result <- c(result, num)

}

}

# Calculate sum, average, minimum, and maximum of the numbers divisible by both 2 and 3 sum\_result <- sum(result)

average\_result <- mean(result) minimum\_result <- min(result) maximum\_result <- max(result)

print(paste("Sum of numbers divisible by both 2 and 3:", sum\_result)) print(paste("Average of numbers divisible by both 2 and 3:", average\_result)) print(paste("Minimum of numbers divisible by both 2 and 3:", minimum\_result)) print(paste("Maximum of numbers divisible by both 2 and 3:", maximum\_result))

Task 3: Create a Dataframe, Filter Rows, and Compute Squared Values using For Loop R

Copy code

# Create a dataframe

data <- data.frame(numbers = 1:10)

# Filter rows based on a condition (e.g., squared values greater than 50) squared\_values <- c()

for (num in data$numbers) { if (num^2 > 50) {

squared\_values <- c(squared\_values, num^2)

}

}

# Display squared values greater than 50 print("Squared values greater than 50:") print(squared\_values)

Task 4: Create a Multidimensional Array, Perform Array Slicing, and Iterate Over a Numeric Vector R

Copy code

# Create a multidimensional array multidim\_array <- array(1:27, dim = c(3, 3, 3))

# Perform array slicing to extract a specific subarray (e.g., slice from first matrix) subarray <- multidim\_array[,,1]

print("Subarray:") print(subarray)

# Iterate over a numeric vector and create a new vector containing the addition of two vectors vector1 <- c(1, 2, 3)

vector2 <- c(4, 5, 6)

sum\_vector <- vector1 + vector2

print("Sum of two vectors:") print(sum\_vector)

Task 5: Create a List of Integers and Perform Operations R

Copy code

# Create a list of integers integer\_list <- list(3, 6, 9, 12, 15)

# Append a new element integer\_list[[6]] <- 18

# Insert a new element at a specific index

integer\_list <- c(integer\_list[1:3], list(7), integer\_list[4:length(integer\_list)])

# Delete an element integer\_list <- integer\_list[-4]

# Display min, max, sum, average minimum <- min(unlist(integer\_list)) maximum <- max(unlist(integer\_list)) sum\_values <- sum(unlist(integer\_list)) average <- mean(unlist(integer\_list))

print(paste("Minimum:", minimum)) print(paste("Maximum:", maximum)) print(paste("Sum:", sum\_values)) print(paste("Average:", average))

# Find a specific element specific\_element <- integer\_list[[3]]

print(paste("Specific Element:", specific\_element))

Task 6: Use If/Else Statement and Plot User Input Data R

Copy code

# Take user input for x and y vectors

x <- as.numeric(readline("Enter values for x (separated by space): ")) y <- as.numeric(readline("Enter values for y (separated by space): "))

# Determine if the numbers in x and y are positive, negative, or zero for (i in 1:length(x)) {

if (x[i] > 0) { print("Positive")

} else if (x[i] < 0) { print("Negative")

} else { print("Zero")

}

}

# Plotting the data

plot(x, y, main = "Plot of X and Y", xlab = "X Values", ylab = "Y Values")

Task 7: Read CSV File, Display Data Information, and Plotting R

Copy code

# Read CSV file into a dataframe data <- read.csv("filename.csv")

# Display the first few rows of the dataframe head(data)

# Summary of dataset summary(data)

# Information about the dataset str(data)

# Line plot plot(data$column\_name, type = "l")

# Scatter plot

plot(data$column\_name1, data$column\_name2)

# Histogram hist(data$column\_name)

# Dot plot dotchart(data$column\_name)

Task 8: Implement Nested If/Else Statement and Data Visualization R

Copy code

# Assuming ’data’ dataframe and ’column\_name’ as the column of interest

# Function to classify a number classify\_number <- function(x) { if (x %% 2 == 0) {

print("Even")

if (x > 0) { print("Positive")

} else if (x < 0) { print("Negative")

} else { print("Zero")

}

} else { print("Odd")

if (x > 0) { print("Positive")

} else if (x < 0) { print("Negative")

} else { print("Zero")

}

}

}

# Apply function to column sapply(data$column\_name, classify\_number)

# Plotting # Line plot

plot(data$column\_name, type = "l")

# Scatter plot

plot(data$column\_name1, data$column\_name2)

# Histogram hist(data$column\_name)

# Dot plot dotchart(data$column\_name)

Task 9: Read Data from Excel File, Print Summary, Box-Whisker Plot, and Dot Plot R

Copy code

# Load necessary package for reading Excel files library(readxl)

# Read data from Excel file excel\_data <- read\_excel("data.xlsx")

# Print 5-point summary summary(excel\_data)

# Box-whisker plot boxplot (excel\_data)

Dot plot dotchart(excel\_data)

bash

Copy code

### Task 10: Read Data from CSV File, Calculate Average, If/Else Statement, and Plotting ‘‘‘R

# Read CSV file into a dataframe data <- read.csv("filename.csv")

# Calculate average of a specific column (assuming ’column\_name’) average <- mean(data$column\_name)

# Determine if the average is above or below a certain threshold (assuming ’threshold’) if (average > threshold) {

print("Above threshold")

} else {

print("Below threshold")

}

# Plotting # Line plot

plot(data$column\_name, type = "l")

# Scatter plot

plot(data$column\_name1, data$column\_name2) # Histogram

hist(data$column\_name)

# Dot plot dotchart(data$column\_name)

Task 11: Create Z-scores and Data Visualization Using Drinks.csv File R

Copy code

# Assuming ’Drinks.csv’ contains necessary data

# Read Drinks.csv file into a dataframe drinks\_data <- read.csv("Drinks.csv")

# Create standardized (Z-) scores for several variables standardized\_scores <- scale(drinks\_data)

# Display data using line graph, scatter plot, histogram, and dot plot # Line plot

plot(drinks\_data$column\_name, type = "l")

# Scatter plot

plot(drinks\_data$column\_name1, drinks\_data$column\_name2)

# Histogram hist(drinks\_data$column\_name)

# Dot plot dotchart(drinks\_data$column\_name)

Task 12: Run Frequencies and Display Data Visualization Using Census.csv File R

Copy code

# Assuming ’Census.csv’ contains necessary data

# Read Census.csv file into a dataframe census\_data <- read.csv("Census.csv")

# Run Frequencies to explore distributions of several variables

# Display data using line graph, scatter plot, histogram, and dot plot # Line plot

plot(census\_data$column\_name, type = "l")

# Scatter plot

plot(census\_data$column\_name1, census\_data$column\_name2)

# Histogram hist(census\_data$column\_name)

# Dot plot dotchart(census\_data$column\_name)

Task 13: Create Two-way Cross Tabulations and Visualize Using Chart Builder Using Census.csv File R

Copy code

# Assuming ’Census.csv’ contains necessary data

# Read Census.csv file into a dataframe census\_data <- read.csv("Census.csv")

# Create two-way cross tabulations to explore relationships between variables # Use Chart Builder to visualize the relationship

# Example:

# Create a cross-tabulation

cross\_table <- table(census\_data$column\_name1, census\_data$column\_name2)

# Visualize using Chart Builder or custom plot functions # For example, bar plot

barplot(cross\_table)

Task 14: Visualize Relationship Between Two Scale Variables Using Scatter Plots and Correlation Coeffic ient Using Census.csv File

R

Copy code

# Assuming ’Census.csv’ contains necessary data

# Read Census.csv file into a dataframe census\_data <- read.csv("Census.csv")

# Visualize the relationship between two scale variables using scatter plots # Assuming ’variable1’ and ’variable2’ are columns of interest plot(census\_data$variable1, census\_data$variable2)

# Quantify the relationship with the correlation coefficient correlation <- cor(census\_data$variable1, census\_data$variable2) print(correlation)

Task 15: Independent-Samples T Test, Interpretation, and Visualization R

Copy code

# Assuming ’Census.csv’ contains the necessary columns

# Read dataset

census\_data <- read.csv("Census.csv")

# Run Independent-Samples T Test

t\_test\_result <- t.test(variable1 ~ categorical\_variable, data = census\_data)

# Interpretation of T-test output print(t\_test\_result)

# Visualize results with an error bar chart library(ggplot2)

ggplot(census\_data, aes(x = categorical\_variable, y = variable1, fill = categorical\_variable)) + geom\_bar(stat = "summary", fun = "mean", position = "dodge") +

geom\_errorbar(stat = "summary", fun.data = "mean\_cl\_normal", position = "dodge", width = 0.5) + labs(x = "Categorical Variable", y = "Variable 1 Mean") +

ggtitle("Error Bar Chart")

Task 16: One-Way ANOVA with Post Hoc Tests and Visualization R

Copy code

# Assuming ’Census.csv’ contains necessary columns for ANOVA

# Perform One-Way ANOVA library(car)

anova\_result <- aov(variable1 ~ categorical\_variable, data = census\_data) summary(anova\_result)

# Post hoc tests (e.g., Tukey’s HSD) posthoc <- TukeyHSD(anova\_result) print(posthoc)

# Visual representation using suitable graphs # For instance, boxplot and bar plot

boxplot(variable1 ~ categorical\_variable, data = census\_data) barplot(table(census\_data$categorical\_variable))

Task 17: Filtering Rows, Iterating Over Numeric Vectors, and Creating New Vectors R

Copy code

# Create a dataframe and filter rows based on a specific condition

filtered\_data <- subset(census\_data, categorical\_variable == "desired\_condition")

# Iterate over two numeric vectors and create a new vector containing the sum values of each element numeric\_vector1 <- c(1, 2, 3, 4, 5)

numeric\_vector2 <- c(6, 7, 8, 9, 10)

sum\_vector <- numeric\_vector1 + numeric\_vector2

# Represent the new vector using suitable graphs (e.g., line plot and histogram) plot(sum\_vector, type = "l", xlab = "Index", ylab = "Sum", main = "Line Plot") hist(sum\_vector, xlab = "Sum", main = "Histogram")

Task 18: Creating a Multidimensional Array, Array Slicing, and Visualization R

Copy code

# Create a multidimensional array

multidim\_array <- array(data = c(1:27), dim = c(3, 3, 3))

# Perform array slicing to extract a specific subarray subarray <- multidim\_array[,,2]

# Visual representation of matrix values using line or scatter plot plot(subarray, type = "l", xlab = "Index", ylab = "Value", main = "Line Plot") plot(subarray, xlab = "Column", ylab = "Value", main = "Scatter Plot")

Task 19: Stacked Dotplots for a Categorical Variable and a Quantitative Variable R

Copy code

# Make stacked dotplots based on categorical variable values for a quantitative variable library(ggplot2)

ggplot(census\_data, aes(x = quantitative\_variable, fill = categorical\_variable)) + geom\_dotplot(binaxis = "y", stackdir = "center", position = "dodge") +

labs(x = "Quantitative Variable", y = "Count") + ggtitle("Stacked Dotplots")

Task 20: Calculate Summary Measures for a Variable R

Copy code

# Calculate summary measures for the entire dataset summary\_measures <- summary(census\_data$variable)

# Calculate summary measures by different groups formed by a categorical variable library(dplyr)

summary\_by\_group <- census\_data %>% group\_by(categorical\_variable) %>%

summarise(mean = mean(variable), sd = sd(variable), q1 = quantile(variable, probs = 0.25), q3 = quantile(variable, probs = 0.75), iqr = IQR(variable))

Task 21: Create Histogram, Dotplot, and Box-and-Whisker Plot R

Copy code

# Create histogram and dotplot of the data hist(census\_data$variable, xlab = "Variable", main = "Histogram")

dotchart(census\_data$variable, labels = rownames(census\_data), cex = 0.7)

# Create box-and-whisker plot for the entire dataset boxplot(census\_data$variable, xlab = "Variable", main = "Boxplot")

# Task 27: Create standardized (Z-) scores for several variables from Drinks.csv and display data

# Question: How can Z-scores be calculated for multiple variables and visualized using different plots?

# Read data from Drinks.csv drinks\_data <- read.csv("Drinks.csv")

# Calculating Z-scores for several variables

z\_scores <- scale(drinks\_data[, c("variable1", "variable2", "variable3")])

# Displaying data using different plots

plot(drinks\_data$variable1, type = "l", col = "blue", xlab = "X", ylab = "Y") plot(drinks\_data$variable1, drinks\_data$variable2, xlab = "Var1", ylab = "Var2") hist(drinks\_data$variable3, xlab = "Var3")

dotchart(drinks\_data$variable1, labels = rownames(drinks\_data), cex = 0.7)

# Task 28: Run Frequencies to explore distributions of several variables from a provided CSV file # Question: How can frequency distributions be explored visually using various plots?

# Assuming ’csv\_data’ is the preexisting CSV data read into a dataframe # Plot line graph

plot(csv\_data$variable1, type = "l", col = "blue", xlab = "X", ylab = "Y")

# Scatter plot

plot(csv\_data$variable1, csv\_data$variable2, xlab = "Var1", ylab = "Var2")

# Histogram

hist(csv\_data$variable3, xlab = "Var3")

# Dot plot

dotchart(csv\_data$variable1, labels = rownames(csv\_data), cex = 0.7)

# Task 29: Obtain summary statistics for scale variables from Drinks.csv and visualize using boxplots # Question: How can summary statistics be obtained and displayed for specific variables?

# Assuming ’drinks\_data’ is the preexisting Drinks.csv data read into a dataframe # Summary statistics for scale variables

summary\_stats <- summary(drinks\_data[, c("variable1", "variable2")])

# Visualize summary statistics using boxplots

boxplot(drinks\_data$variable1, drinks\_data$variable2, names = c("Var1", "Var2"), col = c("blue", "red"))

# Task 30: Visualize relationship between two scale variables from a provided data file and calculate corr elation coefficient

# Question: How can the relationship between two variables be visualized and quantified? # Assuming ’census\_data’ is the preexisting census.csv data read into a dataframe

# Scatter plot for two scale variables

plot(census\_data$variable1, census\_data$variable2, xlab = "Var1", ylab = "Var2")

# Calculate correlation coefficient

correlation\_coeff <- cor(census\_data$variable1, census\_data$variable2) print(paste("Correlation Coefficient:", correlation\_coeff))