**1: Study of basic syntax in R**

# 1. Assigning values to variables

name <- "John"

age <- 25

score <- 87.5

passed <- TRUE

# 2. Printing values

print(paste("Name:", name))

print(paste("Age:", age))

print(paste("Score:", score))

print(paste("Passed:", passed))

# 3. Arithmetic operations

a <- 10

b <- 3

print(paste("Sum:", a + b))

print(paste("Product:", a \* b))

# 4. Using a vector

marks <- c(80, 85, 90)

print("Marks:")

print(marks)

# 5. Conditional statement

if (score > 50) {

print("You passed the test.")

} else {

print("You failed the test.")

}

# 6. Loop (for loop)

print("Counting from 1 to 5:")

for (i in 1:5) {

print(i)

}

# 7. Defining and using a function

greet <- function(person) {

return(paste("Hello,", person))

}

message <- greet(name)

print(message)

**Result:**

[1] "Name: John"

[1] "Age: 25"

[1] "Score: 87.5"

[1] "Passed: TRUE"

[1] "Sum: 13"

[1] "Product: 30"

[1] "Marks:"

[1] 80 85 90

[1] "You passed the test."

[1] "Counting from 1 to 5:

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

[1] "Hello, John"

**2: Implementation of Vector data objects operations R**

v1 <- c(10, 20, 30); v2 <- c(1, 2, 3)

cat("Add:", v1 + v2, "\n")

cat("Sub:", v1 - v2, "\n")

cat("Mul:", v1 \* v2, "\n")

cat("Div:", v1 / v2, "\n")

cat("v1[1]:", v1[1], "v2[2]:", v2[2], "\n")

v1[2] <- 50

cat("Modified v1:", v1, "\nLength:", length(v1), "\n")

**Result:**

Add: 11 22 33

Sub: 9 18 27

Mul: 10 40 90

Div: 10 10 10

v1[1]: 10 v2[2]: 2

Modified v1: 10 50 30

Length: 3

**3: Implementation of matrix ,array and factors and perform various operations in R**

mat <- matrix(1:9, 3, 3); mat2 <- matrix(9:1, 3, 3)

cat("mat:\n"); print(mat)

cat("mat+mat2:\n"); print(mat + mat2)

cat("mat %\*% mat2:\n"); print(mat %\*% mat2)

cat("mat[2,3]:", mat[2,3], "\n\n")

arr <- array(1:12, c(2,3,2))

cat("arr:\n"); print(arr)

cat("arr[1,2,2]:", arr[1,2,2], "Sum:", sum(arr), "\n\n")

colors <- factor(c("red","blue","green","red","green","blue"))

cat("Levels:", levels(colors), "\nFreq:\n"); print(table(colors))

**Result:**

mat:

[,1] [,2] [,3]

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

mat+mat2:

[,1] [,2] [,3]

[1,] 10 10 10

[2,] 10 10 10

[3,] 10 10 10

mat %\*% mat2:

[,1] [,2] [,3]

[1,] 90 54 18

[2,] 114 69 24

[3,] 138 84 30

mat[2,3]: 8

arr:

, , 1

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

, , 2

[,1] [,2] [,3]

[1,] 7 9 11

[2,] 8 10 12

arr[1,2,2]: 9 Sum: 78

Levels: blue green red

Freq:

colors

blue green red

2 2 2

**4: Implementation and use of data frames in R**

names <- c("Alice", "Bob", "Charlie", "David")

ages <- c(25, 30, 22, 35)

scores <- c(85.5, 90, 78.5, 88)

students\_df <- data.frame(Name=names, Age=ages, Score=scores)

print("Students:")

print(students\_df)

print("Names:")

print(students\_df$Name)

students\_df$Passed <- students\_df$Score >= 80

print("With Passed column:")

print(students\_df)

print("Passed students:")

print(subset(students\_df, Passed))

print("Summary:")

print(summary(students\_df))

**Result:**

[1] "Students:"

Name Age Score

1 Alice 25 85.5

2 Bob 30 90.0

3 Charlie 22 78.5

4 David 35 88.0

[1] "Names:"

[1] "Alice" "Bob" "Charlie" "David"

[1] "With Passed column:"

Name Age Score Passed

1 Alice 25 85.5 TRUE

2 Bob 30 90.0 TRUE

3 Charlie 22 78.5 FALSE

4 David 35 88.0 TRUE

[1] "Passed students:"

Name Age Score Passed

1 Alice 25 85.5 TRUE

2 Bob 30 90.0 TRUE

4 David 35 88.0 TRUE

[1] "Summary:"

Name Age Score Passed

Length:4 Min. :22.00 Min. :78.50 Mode :logical

Class :character 1st Qu.:24.25 1st Qu.:83.75 FALSE:1

Mode :character Median :27.50 Median :86.75 TRUE :3

Mean :28.00 Mean :85.50

3rd Qu.:31.25 3rd Qu.:88.50

Max. :35.00 Max. :90.00

**5: Create sample (Dummy) data in R and perform data manipulation with R**

employees <- data.frame(

ID = 1:5,

Name = c("Alice", "Bob", "Charlie", "Diana", "Ethan"),

Department = c("HR", "IT", "Finance", "IT", "HR"),

Salary = c(50000, 60000, 55000, 62000, 48000),

Experience = c(2,4,3,5,1),

stringsAsFactors=FALSE

)

cat("Original Data:\n"); print(employees)

cat("\nSalary > 55000:\n"); print(subset(employees, Salary > 55000))

employees$Bonus <- employees$Salary \* 0.05

cat("\nWith Bonus:\n"); print(employees)

cat("\nAvg Salary by Dept:\n"); print(aggregate(Salary ~ Department, employees, mean))

cat("\nSorted by Experience:\n"); print(employees[order(-employees$Experience), ])

**Result:**

Original Employee Data:

ID Name Department Salary Experience

1 1 Alice HR 50000 2

2 2 Bob IT 60000 4

3 3 Charlie Finance 55000 3

4 4 Diana IT 62000 5

5 5 Ethan HR 48000 1

Employees with Salary > 55000:

ID Name Department Salary Experience

2 2 Bob IT 60000 4

4 4 Diana IT 62000 5

Data with Bonus Column:

ID Name Department Salary Experience Bonus

1 1 Alice HR 50000 2 2500

2 2 Bob IT 60000 4 3000

3 3 Charlie Finance 55000 3 2750

4 4 Diana IT 62000 5 3100

5 5 Ethan HR 48000 1 2400

**6: Study and Implement of various control structures in R**

# 1. if and if...else

x <- 10

if (x > 0) {

cat("x is positive\n")

} else {

cat("x is zero or negative\n")

}

# 2. ifelse (vectorized)

marks <- c(85, 45, 70, 30, 90)

result <- ifelse(marks >= 50, "Pass", "Fail")

cat("\nResults using ifelse():\n")

print(data.frame(Marks = marks, Result = result))

# 3. for loop

cat("\nPrinting numbers using for loop:\n")

for (i in 1:5) {

cat(i, " ")

}

# 4. while loop

cat("\n\nWhile loop (counting up to 3):\n")

count <- 1

while (count <= 3) {

cat("Count:", count, "\n")

count <- count + 1

}

# 5. repeat loop with break

cat("\nRepeat loop example:\n")

y <- 1

repeat {

cat("y =", y, "\n")

y <- y + 1

if (y > 3) {

break

}

}

# 6. switch statement

cat("\nSwitch statement example:\n")

day <- 3

day\_name <- switch(day,

"Sunday",

"Monday",

"Tuesday",

"Wednesday",

"Thursday",

"Friday",

"Saturday")

cat("Day 3 is:", day\_name, "\n")

**Result:**

x is positive

Results using ifelse():

Marks Result

1 85 Pass

2 45 Fail

3 70 Pass

4 30 Fail

5 90 Pass

Printing numbers using for loop:

1. 2 3 4 5

While loop (counting up to 3):

Count: 1

Count: 2

Count: 3

Repeat loop example:

y = 1

y = 2

y = 3

Switch statement example:

Day 3 is: Tuesday

**7: Study and Implementation of Data visualization with ggplot2**

library(ggplot2)

# Sample data

data <- data.frame(Category = c("A", "B", "C", "D"), Values = c(10, 23, 17, 9))

scatter\_data <- data.frame(x = rnorm(50), y = rnorm(50), Group = sample(c("G1", "G2"), 50, TRUE))

line\_data <- data.frame(Time = 1:10, Measurement = cumsum(rnorm(10)))

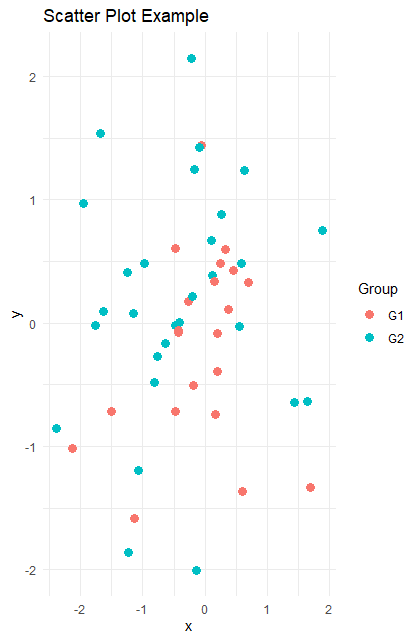
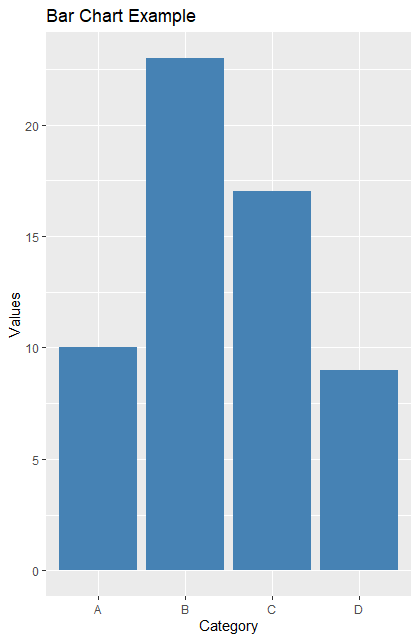
# Plots

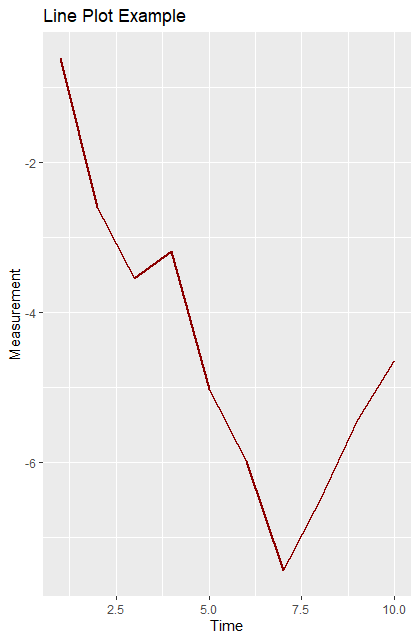
print(ggplot(data, aes(Category, Values)) + geom\_bar(stat="identity", fill="steelblue") + ggtitle("Bar Chart Example"))

print(ggplot(scatter\_data, aes(x, y, color = Group)) + geom\_point(size=3) + ggtitle("Scatter Plot Example") + theme\_minimal())

print(ggplot(line\_data, aes(Time, Measurement)) + geom\_line(color="darkred", size=1) + ggtitle("Line Plot Example"))

**Result:**

**** ****

****

**8: Implement Apriori algorithm to extract association rule of data mining**

# Transactions

transactions <- list(

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

c("bread", "butter"),

c("milk", "bread"),

c("milk", "butter"),

c("bread", "butter", "jam"),

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

)

# Minimum thresholds

min\_support <- 0.3

min\_conf <- 0.7

# Support function

support <- function(items, trans) {

sum(sapply(trans, function(t) all(items %in% t))) / length(trans)

}

# Unique items

items <- unique(unlist(transactions))

# Frequent 1-itemsets

cat("Frequent 1-itemsets:\n")

for (i in items) {

s <- support(i, transactions)

if (s >= min\_support) cat(sprintf("{%s} support: %.2f\n", i, s))

}

# Association rules from 2-itemsets

cat("\nAssociation Rules:\n")

for (i in 1:(length(items)-1)) {

for (j in (i+1):length(items)) {

A <- items[i]; B <- items[j]

sAB <- support(c(A, B), transactions)

cA\_B <- sAB / support(A, transactions)

cB\_A <- sAB / support(B, transactions)

if (sAB >= min\_support && cA\_B >= min\_conf)

cat(sprintf("{%s} => {%s} (support: %.2f, conf: %.2f)\n", A, B, sAB, cA\_B))

if (sAB >= min\_support && cB\_A >= min\_conf)

cat(sprintf("{%s} => {%s} (support: %.2f, conf: %.2f)\n", B, A, sAB, cB\_A))

}

}

**Result:**

Frequent 1-itemsets:

{milk} support: 0.67

{bread} support: 0.83

{butter} support: 0.83

{jam} support: 0.33

Association Rules:

{milk} => {bread} (support: 0.50, conf: 0.75)

{milk} => {butter} (support: 0.50, conf: 0.75)

{bread} => {butter} (support: 0.67, conf: 0.80)

{butter} => {bread} (support: 0.67, conf: 0.80)

{jam} => {bread} (support: 0.33, conf: 1.00)

{jam} => {butter} (support: 0.33, conf: 1.00)

**9: Implement K-means clustering technique**

data(iris)

set.seed(123)

kmeans\_result <- kmeans(iris[, 1:4], centers = 3)

print(kmeans\_result)

iris$Cluster <- kmeans\_result$cluster

plot(iris[, 1:2], col = iris$Cluster, main = "K-Means Clustering",

xlab = "Sepal Length", ylab = "Sepal Width", pch = 19)

**Result:**

K-means clustering with 3 clusters of sizes 50, 62, 38

Cluster means:

Sepal.Length Sepal.Width Petal.Length Petal.Width

1 5.006000 3.428000 1.462000 0.246000

2 5.901613 2.748387 4.393548 1.433871

3 6.850000 3.073684 5.742105 2.071053

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

[38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

[75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3

[112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3

[149] 3 2

Within cluster sum of squares by cluster:

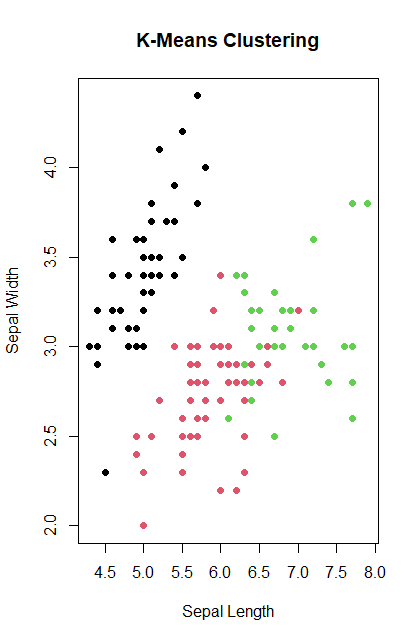
[1] 15.15100 39.82097 23.87947

(between\_SS / total\_SS = 88.4 %)

Available components:

[1] "cluster" "centers" "totss" "withinss" "tot.withinss"

[6] "betweenss" "size" "iter" "ifault"



**10: Implement any one Hierarchal clustering**

data(iris)

dist\_matrix <- dist(iris[, 1:4])

hc <- hclust(dist\_matrix, method = "complete")

plot(hc, main = "Hierarchical Clustering", labels = FALSE)

clusters <- cutree(hc, 3)

print(clusters)

iris$Cluster <- clusters

**Result:**

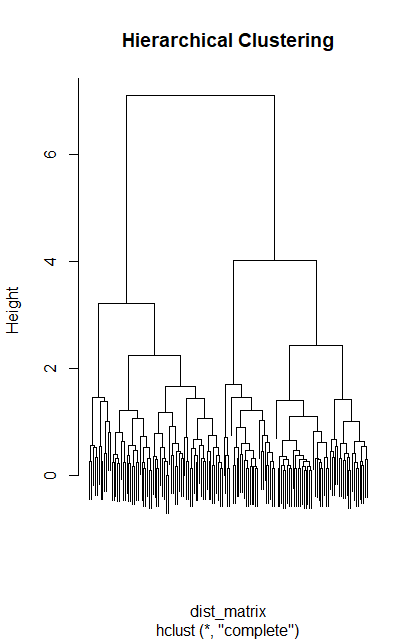
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 2 3 2 3 2 3 3 3 3 2 3 2 3 3 2 3 2 3 2 2

[75] 2 2 2 2 2 3 3 3 3 2 3 2 2 2 3 3 3 2 3 3 3 3 3 2 3 3 2 2 2 2 2 2 3 2 2 2 2

[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

[149] 2 2



**11: Implement Classification algorithm**

library(rpart)

data(iris)

model <- rpart(Species ~ ., iris, method = "class")

print(model)

plot(model); text(model, use.n = TRUE)

pred <- predict(model, iris, type = "class")

conf\_matrix <- table(pred, iris$Species)

print(conf\_matrix)

cat("Accuracy:", round(sum(diag(conf\_matrix)) / sum(conf\_matrix) \* 100, 2), "%\n")

**Result:**

n= 150

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)

2) Petal.Length< 2.45 50 0 setosa (1.00000000 0.00000000 0.00000000) \*

3) Petal.Length>=2.45 100 50 versicolor (0.00000000 0.50000000 0.50000000)

6) Petal.Width< 1.75 54 5 versicolor (0.00000000 0.90740741 0.09259259) \*

7) Petal.Width>=1.75 46 1 virginica (0.00000000 0.02173913 0.97826087) \*

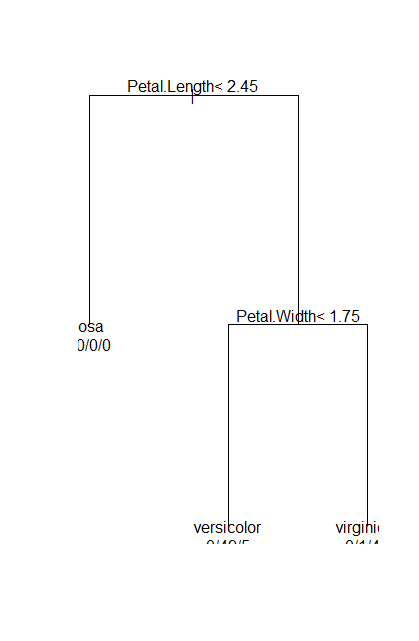
pred setosa versicolor virginica

setosa 50 0 0

versicolor 0 49 5

virginica 0 1 45

Accuracy: 96 %



**12: Implement Decision Tree**

# Load necessary library

library(rpart)

# Load the iris dataset

data(iris)

# Build the decision tree model

model <- rpart(Species ~ ., data = iris, method = "class")

# Print the model

print(model)

# Plot the decision tree (basic)

plot(model)

text(model, use.n = TRUE)

library(rpart.plot)

rpart.plot(model)

**Result:**

n= 150

node), split, n, loss, yval, (yprob)

\* denotes terminal node

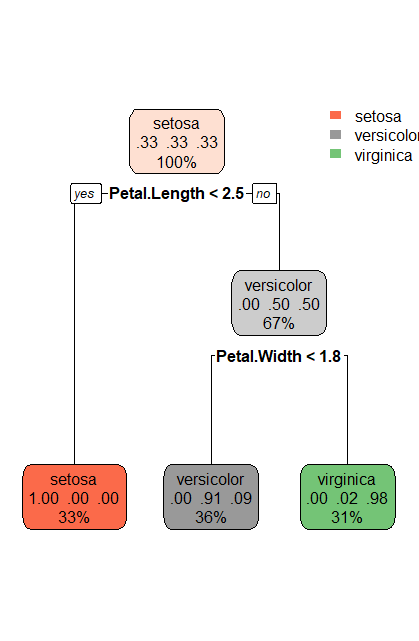
1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)

2) Petal.Length< 2.45 50 0 setosa (1.00000000 0.00000000 0.00000000) \*

3) Petal.Length>=2.45 100 50 versicolor (0.00000000 0.50000000 0.50000000)

6) Petal.Width< 1.75 54 5 versicolor (0.00000000 0.90740741 0.09259259) \*

7) Petal.Width>=1.75 46 1 virginica (0.00000000 0.02173913 0.97826087) \*



**13: Implement Linear regression**

data(mtcars)

model <- lm(mpg ~ hp, mtcars)

summary(model)

plot(mtcars$hp, mtcars$mpg, pch=19, col="blue", main="MPG vs Horsepower", xlab="Horsepower", ylab="MPG")

abline(model, col="red", lwd=2)

print(predict(model, data.frame(hp=c(100,150,200))))

**Result:**

Call:

lm(formula = mpg ~ hp, data = mtcars)

Residuals:

Min 1Q Median 3Q Max

-5.7121 -2.1122 -0.8854 1.5819 8.2360

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 30.09886 1.63392 18.421 < 2e-16 \*\*\*

hp -0.06823 0.01012 -6.742 1.79e-07 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.863 on 30 degrees of freedom

Multiple R-squared: 0.6024, Adjusted R-squared: 0.5892

F-statistic: 45.46 on 1 and 30 DF, p-value: 1.788e-07

1 2 3

23.27603 19.86462 16.45320

