Biochemical Oxygen Demand

29-Oct-24

Name: Faizan

Registration # FA21-BSE-011

# Dataset Assigned

BOD (Biochemical Oxygen Demand).

## GitHub Link of Implementation

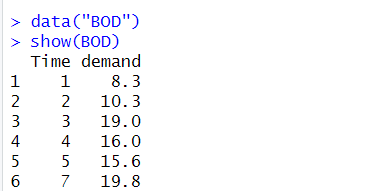
https://github.com/Faizan-Kh/Data-Science

# Minkowski Distance

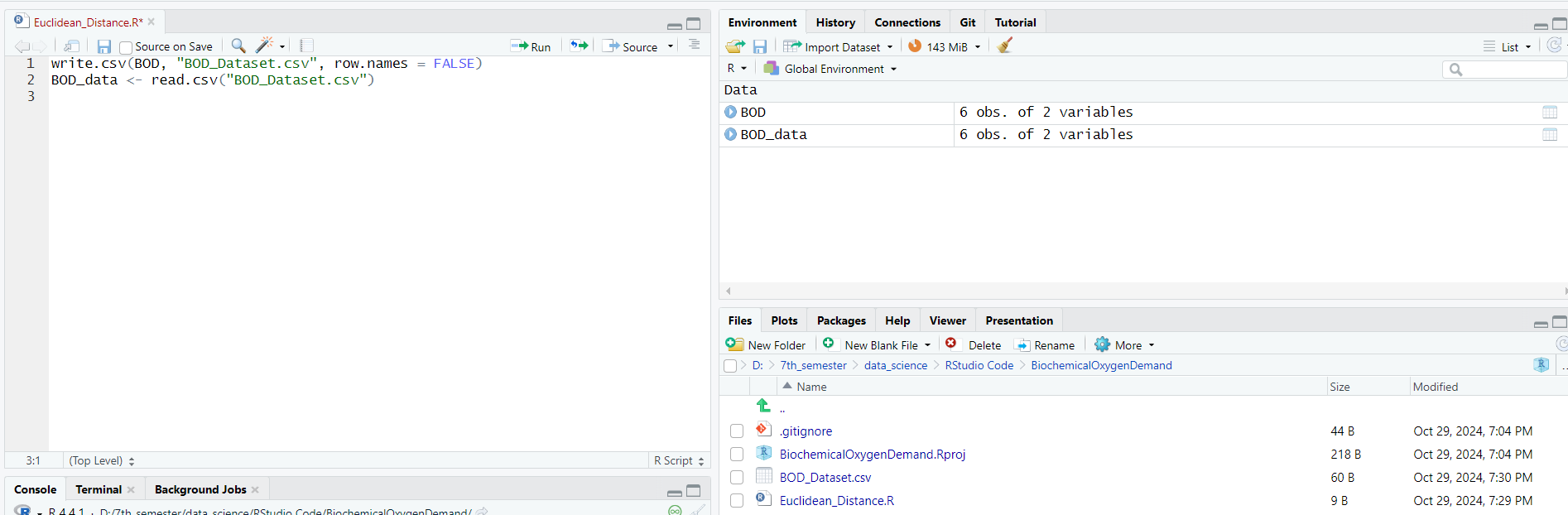
Minkowski distance is the generalization of Euclidean distance. Euclidean Distance Length of the line segment between two points in a Euclidean space is known as the Euclidean Distance [1].

## RStudio implementation

### Loading the dataset to memory:

****

### Exporting the dataset to a CSV file and importing it



### Calculating the Euclidean, city block and supremum distance by putting r= 1, 2, and ∞ on the imported dataset

# Load the BOD dataset and save it to a CSV

write.csv(BOD, "BOD\_Dataset.csv", row.names = FALSE)

# Read the dataset back into R

BOD\_data <- read.csv("BOD\_Dataset.csv")

# Define a generalized Minkowski distance function

minkowski\_dist <- function(x, y, r) {

if (r == Inf) {

return(max(abs(x - y))) # L∞ norm (Supremum distance)

} else {

return((sum(abs(x - y)^r))^(1/r)) # Lr norm (Minkowski distance)

}

}

# Get the number of demands

n <- length(BOD\_data$demand)

# Initialize matrices to hold the distances

manhattan\_distances <- matrix(NA, n, n)

euclidean\_distances <- matrix(NA, n, n)

supremum\_distances <- matrix(NA, n, n)

# Set values for r

r\_values <- c(1, 2, Inf) # 1 = Manhattan, 2 = Euclidean, Inf = Supremum

# Calculate the distances for each pair

for (i in 1:n) {

for (j in 1:n) {

manhattan\_distances[i, j] <- minkowski\_dist(BOD\_data$demand[i], BOD\_data$demand[j], r\_values[1])

euclidean\_distances[i, j] <- minkowski\_dist(BOD\_data$demand[i], BOD\_data$demand[j], r\_values[2])

supremum\_distances[i, j] <- minkowski\_dist(BOD\_data$demand[i], BOD\_data$demand[j], r\_values[3])

}

}

# Print the distance matrices

cat("Manhattan Distance Matrix (r=1):\n")

print(manhattan\_distances)

cat("\nEuclidean Distance Matrix (r=2):\n")

print(euclidean\_distances)

cat("\nSupremum Distance Matrix (r=∞):\n")

print(supremum\_distances)

print(distances)

#### Output

A screenshot of a computer

Description automatically generated

# SMC (Simple Matching Coefficient)

Also known as the statistic used for comparing the similarities between sample sets [2].

## Implementation

1. We initialize a threshold to convert the data into binary

threshold <- mean(BOD\_data$demand)

#Converting the data to binary

binary\_demand <- ifelse(BOD\_data$demand > threshold, 1, 0)

1. Treating both 1s and 0s

n <- length(binary\_demand)

P\_11 <- sum(binary\_demand %\*% t(binary\_demand)) # Both 1s

P\_00 <- sum((1 - binary\_demand) %\*% t(1 - binary\_demand)) #Both 0s

1. Calculating the total number of similarities and SMC

#Total number of comparisons

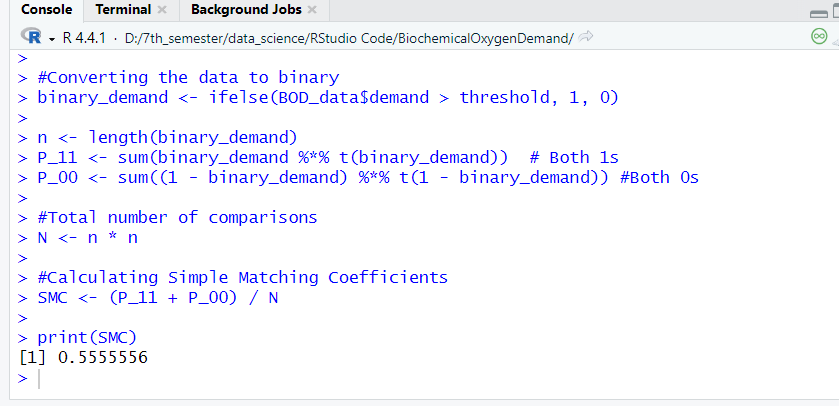
N <- n \* n

#Calculating Simple Matching Coefficients

SMC <- (P\_11 + P\_00) / N

print(SMC)

### Output



# Jaccard Coefficient

Also known as Jaccard index, is a statistic used to find out the similarity and diversity of sample sets [3].

## Implementation

threshold <- 15 # Example threshold

binary\_demand <- ifelse(BOD$demand > threshold, 1, 0)

n <- length(binary\_demand)

# Initialize counts for M11, M01, and M10

M\_11 <- 0

M\_01 <- 0

M\_10 <- 0

# Loop through each pair to count M11, M01, and M10

for (i in 1:n) {

for (j in 1:n) {

if (i != j) {

if (binary\_demand[i] == 1 && binary\_demand[j] == 1) {

M\_11 <- M\_11 + 1

} else if (binary\_demand[i] == 0 && binary\_demand[j] == 1) {

M\_01 <- M\_01 + 1

} else if (binary\_demand[i] == 1 && binary\_demand[j] == 0) {

M\_10 <- M\_10 + 1

}

}

}

}

# Calculate Jaccard Coefficient

jaccard\_coefficient <- M\_11 / (M\_11 + M\_01 + M\_10)

print(jaccard\_coefficient)

### Output

A screenshot of a computer code

Description automatically generated

# Cosine Similarity

The cosine measure computes the *angle* between the two documents, which is **insensitive** to the absolute length of the document

## Implementation

# Load the BOD dataset and save it to a CSV

write.csv(BOD, "BOD\_Dataset.csv", row.names = FALSE)

#or load it instead

#load("BOD")

# Read the dataset back into R

BOD\_data <- read.csv("BOD\_Dataset.csv")

# Define a function to calculate cosine similarity

cosine\_similarity <- function(x, y) {

dot\_product <- sum(x \* y)

magnitude\_x <- sqrt(sum(x^2))

magnitude\_y <- sqrt(sum(y^2))

# Calculate cosine similarity

if (magnitude\_x == 0 || magnitude\_y == 0) {

return(NA) # Avoid division by zero

} else {

return(dot\_product / (magnitude\_x \* magnitude\_y))

}

}

# Get the number of demands

n <- length(BOD\_data$demand)

# Initialize a matrix to hold the cosine similarities

cosine\_similarities <- matrix(NA, n, n)

# Calculate the cosine similarity for each pair

for (i in 1:n) {

for (j in 1:n) {

cosine\_similarities[i, j] <- cosine\_similarity(BOD\_data$demand[i], BOD\_data$demand[j])

}

}

# Print the cosine similarity matrix

cat("Cosine Similarity Matrix:\n")

print(cosine\_similarities)

### Output

A screenshot of a computer code

Description automatically generated

# Correlation

* Correlation measures the **linear relationship** between the attributes of the objects yk=axk+b for binary or continuous variable
* Measure of linear dependency b/w two variables x and y

## Implementation

# Load the BOD dataset

BOD\_data <- read.csv("BOD\_Dataset.csv")

# Extract the Time and Demand columns

time <- BOD\_data$Time

demand <- BOD\_data$demand

# Calculate means

mean\_time <- mean(time)

mean\_demand <- mean(demand)

# Calculate covariance

covariance <- sum((time - mean\_time) \* (demand - mean\_demand)) / length(time)

# Calculate standard deviations

std\_dev\_time <- sqrt(sum((time - mean\_time)^2) / length(time))

std\_dev\_demand <- sqrt(sum((demand - mean\_demand)^2) / length(demand))

# Calculate Pearson correlation coefficient

correlation\_coefficient <- covariance / (std\_dev\_time \* std\_dev\_demand)

# Print results

cat("Mean Time:", mean\_time, "\n")

cat("Mean Demand:", mean\_demand, "\n")

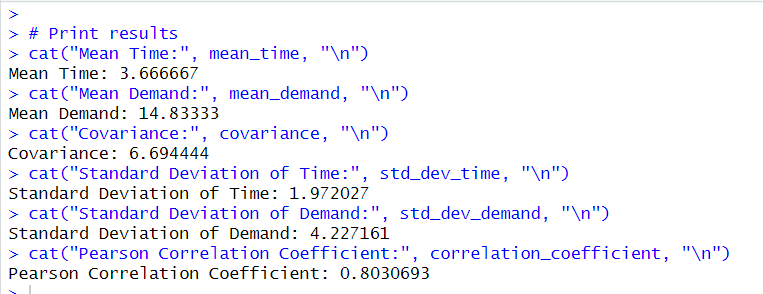
cat("Covariance:", covariance, "\n")

cat("Standard Deviation of Time:", std\_dev\_time, "\n")

cat("Standard Deviation of Demand:", std\_dev\_demand, "\n")

cat("Pearson Correlation Coefficient:", correlation\_coefficient, "\n")

### Output



# Clustering using the distance matrix

# Load the BOD dataset and save it to a CSV

write.csv(BOD, "BOD\_Dataset.csv", row.names = FALSE)

# Read the dataset back into R

BOD\_data <- read.csv("BOD\_Dataset.csv")

# Binarize the demand data based on a threshold

threshold <- 15

BOD\_data$binarized\_demand <- ifelse(BOD\_data$demand > threshold, 1, 0)

# Define the distance functions

euclidean\_dist <- function(x, y) {

sqrt(sum((x - y) ^ 2))

}

minkowski\_dist <- function(x, y, r) {

sum(abs(x - y) ^ r)^(1/r)

}

cosine\_similarity <- function(x, y) {

dot\_product <- sum(x \* y)

magnitude\_x <- sqrt(sum(x^2))

magnitude\_y <- sqrt(sum(y^2))

if (magnitude\_x == 0 || magnitude\_y == 0) {

return(NA) # Avoid division by zero

} else {

return(dot\_product / (magnitude\_x \* magnitude\_y))

}

}

# Get the number of demands

n <- nrow(BOD\_data)

demand\_matrix <- as.matrix(BOD\_data$demand)

# Calculate distance matrices

euclidean\_matrix <- matrix(NA, n, n)

minkowski\_matrix <- matrix(NA, n, n)

cosine\_matrix <- matrix(NA, n, n)

# Calculate the distance matrices

for (i in 1:n) {

for (j in 1:n) {

# Using demand values for distance measures

euclidean\_matrix[i, j] <- euclidean\_dist(demand\_matrix[i, , drop = FALSE], demand\_matrix[j, , drop = FALSE])

minkowski\_matrix[i, j] <- minkowski\_dist(demand\_matrix[i, , drop = FALSE], demand\_matrix[j, , drop = FALSE], r = 3) # r can be 1, 2, or 3

cosine\_matrix[i, j] <- cosine\_similarity(demand\_matrix[i, , drop = FALSE], demand\_matrix[j, , drop = FALSE])

}

}

# Print the distance matrices

cat("Euclidean Distance Matrix:\n")

print(euclidean\_matrix)

cat("\nMinkowski Distance Matrix:\n")

print(minkowski\_matrix)

cat("\nCosine Similarity Matrix:\n")

print(cosine\_matrix)

# Clustering using K-means based on Euclidean distance

set.seed(123) # For reproducibility

k <- 2 # Number of clusters

# Since K-means uses Euclidean distance, we will use the original demand matrix

kmeans\_result <- kmeans(demand\_matrix, centers = k)

# Add cluster assignment to the original dataset

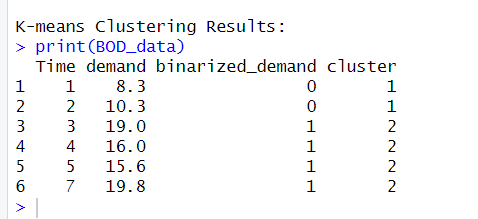
BOD\_data$cluster <- kmeans\_result$cluster

# Print the clustering result

cat("\nK-means Clustering Results:\n")

print(BOD\_data)

## Output



# References

1. <https://en.wikipedia.org/wiki/Euclidean_distance>
2. <https://en.wikipedia.org/wiki/Simple_matching_coefficient#:~:text=The%20simple%20matching%20coefficient%20(SMC,and%20diversity%20of%20sample%20sets.&text=value%201%2C%20and-,is%20the%20total%20number%20of%20attributes%20where%20A%20has,and%20B%20has%20value%200>.
3. https://en.wikipedia.org/wiki/Jaccard\_index