**Sensor type2 (Soil Temperature data column of original dataset)**

library(Matrix)

library(GLDEX)

library(bayestestR)

options(scipen=999)

#X\_temp <- matrix(rnorm(100), ncol = 2) # 50x2 matrix with random values

#fun.zero.omit(c(0,1,2,3,4,0,2))

#x <- read.csv('D:/All\_Papers/Paper\_1\_O1/dataset details')

x<-read.csv('D:/R\_Experiments/Soil\_Temperature\_DS.csv')

# Convert the data frame to a matrix

x\_matrix <- as.matrix(x, nrow=100000,ncol=1)

# Convert the matrix to an array

x\_array <- array(x\_matrix, dim = dim(x\_matrix))

rankMatrix(X\_temp)

# Convert to a 2D matrix (e.g., 150 rows and 20 columns)

matrix1 <- matrix(x\_array, nrow = 1000, ncol = 100)

matrix2=matrix1

matrix3=matrix1

# Fetch and display all the index positions containing zero

zero\_indices <- which(matrix1 == 0, arr.ind = TRUE)

print("Index positions containing zero:")

print(zero\_indices)

# Calculate and display the total number of index positions containing zero

total\_zeros <- nrow(zero\_indices)

print(paste("Total number of index positions containing zero:", total\_zeros))

**OUTPUT:**

**"Total number of index positions containing zero: 2096"**

**To fill zero entries in original dataset (Approach 1)**

# Find the index positions of non-zero elements

non\_zero\_indices <- which(matrix1 != 0, arr.ind = TRUE)

# Display the index positions

print(non\_zero\_indices)

map\_estimate(matrix1[non\_zero\_indices])

rankMatrix(matrix1)

rankMatrix(matrix2)

constant\_value <- 45.00

matrix2[zero\_indices]<-constant\_value #map\_estimate() of non\_zero\_indices

matrix2

rankMatrix(matrix2)

mean(matrix2)

mean(matrix1)

mse(22.5361, 23.4793)

mape(22.5361, 23.4793)

rmse(22.5361, 23.4793)

mae(22.5361, 23.4793)

FuzzyR::fuzzyr.accuracy(22.5361, 23.4793)

|  |
| --- |
| > mse(22.5361, 23.4793)  [1] 0.8896262  > mape(22.5361, 23.4793)  [1] 4.185285  > rmse(22.5361, 23.4793)  [1] 0.9432  > mae(22.5361, 23.4793)  [1] 0.9432  > FuzzyR::fuzzyr.accuracy(22.5361, 23.4793)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.94320000 0.94320000 NA 0.04017156 0.04017156 4.01715554 4.09949712 0.04017156 |
|  |
| **To fill zero entries in original dataset (Second Approach )**  # Function to check which rows contain no zeros and show those rows  check\_rows\_without\_zeros <- function(matrix1) {  rows\_without\_zeros <- apply(matrix1, 1, function(row) all(row != 0))  no\_zero\_rows <- which(rows\_without\_zeros)    if (length(no\_zero\_rows) == 0) {  cat("All rows contain zero values.\n")  } else {  cat("Rows containing no zero values:\n")  print(no\_zero\_rows)  cat("Displaying rows with no zeros (showing first 10 rows for brevity):\n")  print(matrix[no\_zero\_rows[1:10], ])  }  }  # Check the rows and show the ones containing no zeros  check\_rows\_without\_zeros(matrix1)  matrix1[10,]  map\_estimate(matrix1[10,])  # fill the map\_estimate value of 10th row i.e. 16.86 into the places of the matrix where zero is present in the original matrix  # Define the constant value to replace zeros  constant\_value <- 16.85  matrix3[zero\_indices]<-constant\_value #map\_estimate() of [10,] th row  matrix3  rankMatrix(matrix3)  mean(matrix3)  mean(matrix1)  mse(22.5361, 22.88928)  mape(22.5361, 22.88928)  rmse(22.5361, 22.88928)  mae(22.5361, 22.88928)  FuzzyR::fuzzyr.accuracy(22.5361, 22.88928)# This provides best result so take it as filling value in zero entries  filled\_matrix=matrix3 # Filled matrix containing no missing values   |  | | --- | | > mean(matrix3)  [1] 22.88928  > mean(matrix1)  [1] 22.5361  >  > mse(22.5361, 22.88928)  [1] 0.1247361  > mape(22.5361, 22.88928)  [1] 1.567174  > rmse(22.5361, 22.88928)  [1] 0.35318  > mae(22.5361, 22.88928)  [1] 0.35318  > FuzzyR::fuzzyr.accuracy(22.5361, 22.88928)# This provides best result so take it as filling value in zero entries  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.35318000 0.35318000 NA 0.01542993 0.01542993 1.54299305 1.55498974 0.01542993 | |  | | |  | | --- | | > | |  |  | | --- | |  | |

Second approach provides good result, so select it as the better, so obtained matrix3 is select as filled matrix to perform the data recovery operation tastings.

**Case 1 :** Complete column missing (Let 50 th column of original dataset)

**Code:**

filled\_matrix=matrix3

matrix3\_NA=matrix3

rankMatrix(filled\_matrix) # out put 0.0000000000002220446

# Step 3: Set the values at a particular column position to 0 say column 50 (Let column 50 is missing)

matrix3\_NA[,50] <- 0

map\_estimate(filled\_matrix[,51])

matrix3\_NA[,50]<-17.90 #value of map\_estimate(filled\_matrix [,51])

mean(matrix3\_NA)

mean(filled\_matrix)

mse(22.88928, 22.83544)

mape(22.88928, 22.83544)

rmse(22.88928, 22.83544)

mae(22.88928, 22.83544)

FuzzyR::fuzzyr.accuracy(22.88928, 22.83544)

**Output:**

Rank of matrix filled\_matrix: 0.0000000000002220446

|  |
| --- |
| mean(matrix3\_NA)  [1] 22.88426  > mean(filled\_matrix)  [1] 22.88928  > mse(22.88928, 22.83544)  [1] 0.002898746  > mape(22.88928, 22.83544)  [1] 0.2352193  > rmse(22.88928, 22.83544)  [1] 0.05384  > mae(22.88928, 22.83544)  [1] 0.05384  > FuzzyR::fuzzyr.accuracy(22.88928, 22.83544)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.053840000 0.053840000 NA 0.002357739 0.002357739 0.235773867 0.235496248 0.002357739 |
|  |
| |  | | --- | |  |   Case-2 Let complete row missing:  # Step 4:Set the values at a particular row position to 0 say row 50 (Let complete row 50 is missing)  rankMatrix(filled\_matrix)  matrix4=filled\_matrix  matrix4\_NA=matrix4  matrix4\_NA[50,]<-0  map\_estimate(filled\_matrix[51,])  matrix4\_NA[50,]<-13.26  mean(matrix4\_NA)  mean(filled\_matrix)  mse(22.88928, 22.88017)  mape(22.88928, 22.88017)  rmse(22.88928, 22.88017)  mae(22.88928, 22.88017)  FuzzyR::fuzzyr.accuracy(22.88928, 22.88017) |
| > mean(matrix4\_NA)  [1] 22.88017  > mean(filled\_matrix)  [1] 22.88928  > mse(22.88928, 22.88017)  [1] 0.0000829921  > mape(22.88928, 22.88017)  [1] 0.03980029  > rmse(22.88928, 22.88017)  [1] 0.00911  > mae(22.88928, 22.88017)  [1] 0.00911  > FuzzyR::fuzzyr.accuracy(22.88928, 22.88017)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.0091100000 0.0091100000 NA 0.0003981614 0.0003981614 0.0398161377 0.0398082127 0.0003981614 |
|  |
| |  | | --- | | > | |

matrix4\_NA=matrix4

matrix4\_NA[50,]<-0

matrix4\_NA[50,]

map\_estimate(filled\_matrix[49,])

matrix4\_NA[50,]<-31.04

mean(matrix4\_NA)

mean(filled\_matrix)

mse(22.88928, 22.89795)

mape(22.88928, 22.89795)

rmse(22.88928, 22.89795)

mae(22.88928, 22.89795)

FuzzyR::fuzzyr.accuracy(22.88928, 22.89795)

**Output:**

|  |
| --- |
| > mean(matrix4\_NA)  [1] 22.89795  > mean(filled\_matrix)  [1] 22.88928  > mse(22.88928, 22.89795)  [1] 0.0000751689  > mape(22.88928, 22.89795)  [1] 0.03787799  > rmse(22.88928, 22.89795)  [1] 0.00867  > mae(22.88928, 22.89795)  [1] 0.00867  > FuzzyR::fuzzyr.accuracy(22.88928, 22.89795)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.0086700000 0.0086700000 NA 0.0003786365 0.0003786365 0.0378636515 0.0378708212 0.0003786365 |
|  |
| |  | | --- | | **Second Approach provides best result** | |

**Case 3**

Missing value at Random positions

# Set 100 random positions to zero in each column in the original matrix and recovery by map\_estimate()

# Initialize matrix4 with original\_matrix

matrix5=filled\_matrix

matrix5\_NA=matrix5

matrix5\_NA

# Initialize matrix1 with original\_matrix

# Set 10 random positions to zero in each column

set.seed(42) # Setting a seed for reproducibility

for (i in 1:ncol(matrix5\_NA)) {

zero\_positions <- sample(1:nrow(matrix5\_NA), 100) # Get 100 random positions

matrix5\_NA[zero\_positions, i] <- 0

}

matrix5\_NA

# Print the modified matrix (showing the first 10 rows and 10 columns for brevity)

print("Modified Matrix with 10 random zeros in each column (First 10 rows and 10 columns):")

print(matrix5\_NA[1:10, 1:10])

# Fetch and display all the index positions containing zero

zero\_indices <- which(matrix5\_NA == 0, arr.ind = TRUE)

print("Index positions containing zero:")

print(zero\_indices)

# Calculate and display the total number of index positions containing zero

total\_zeros <- nrow(zero\_indices)

print(paste("Total number of index positions containing zero:", total\_zeros))

non\_zero\_indices <- which(matrix5\_NA != 0, arr.ind = TRUE)

# Display number of index positions containing nonzeros

print(non\_zero\_indices)

total\_non\_zeros <- nrow(non\_zero\_indices)

print(paste("Total number of index positions containing nonzero:", total\_non\_zeros))

Output:

"Total number of index positions containing zero: 10000"

"Total number of index positions containing nonzero: 90000"

map\_estimate(matrix5\_NA[non\_zero\_indices])

matrix5\_NA[zero\_indices]

matrix5\_NA[zero\_indices]<-13.00 #map\_estimate() of nonzero indices

matrix5\_NA[zero\_indices]

mean(filled\_matrix)

mean(matrix5\_NA)

mse(22.88928, 21.89839)

mape(22.88928,21.89839)

rmse(22.88928, 21.89839)

mae(22.88928, 21.89839)

FuzzyR::fuzzyr.accuracy(22.88928, 21.89839)

|  |
| --- |
| mean(filled\_matrix)  [1] 22.88928  > mean(matrix5\_NA)  [1] 21.89839  > mse(22.88928, 21.89839)  [1] 0.981863  > mape(22.88928,21.89839)  [1] 4.329057  > rmse(22.88928, 21.89839)  [1] 0.99089  > mae(22.88928, 21.89839)  [1] 0.99089  > FuzzyR::fuzzyr.accuracy(22.88928, 21.89839)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.99089000 0.99089000 NA 0.04524945 0.04524945 4.52494453 4.42483389 0.04524945 |
|  |
| |  | | --- | | **Approach 2(for random fill)**  map\_estimate(filled\_matrix[,1])  matrix5\_NA[zero\_indices]  matrix5\_NA[zero\_indices]<-26.85 #map\_estimate() of [,1] st column  mean(filled\_matrix)  mean(matrix5\_NA)  mse(22.88928, 23.28339)  mape(22.88928,23.28339)  rmse(22.88928, 23.28339)  mae(22.88928, 23.28339)  FuzzyR::fuzzyr.accuracy(22.88928, 23.28339)  **Output:**  > mean(filled\_matrix)  [1] 22.88928  > mean(matrix5\_NA)  [1] 23.28339  > mse(22.88928, 23.28339)  [1] 0.1553227  > mape(22.88928,23.28339)  [1] 1.72181  > rmse(22.88928, 23.28339)  [1] 0.39411  > mae(22.88928, 23.28339)  [1] 0.39411  > FuzzyR::fuzzyr.accuracy(22.88928, 23.28339)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.39411000 0.39411000 NA 0.01692666 0.01692666 1.69266589 1.70711375 0.01692666 | |

**Approach 2**

map\_estimate(filled\_matrix[1,])

matrix5\_NA[zero\_indices]

matrix5\_NA[zero\_indices]<-20.35 #map\_estimate() of [1,] st row

matrix5\_NA[zero\_indices]

mean(filled\_matrix)

mean(matrix5\_NA)

mse(22.88928, 22.63339)

mape(22.88928,22.63339)

rmse(22.88928, 22.63339)

mae(22.88928, 22.63339)

FuzzyR::fuzzyr.accuracy(22.88928, 22.63339)

**Output:**

> mean(filled\_matrix)

[1] 22.88928

> mean(matrix5\_NA)

[1] 22.63339

> mse(22.88928, 22.63339)

[1] 0.06547969

> mape(22.88928,22.63339)

[1] 1.117947

> rmse(22.88928, 22.63339)

[1] 0.25589

> mae(22.88928, 22.63339)

[1] 0.25589

> FuzzyR::fuzzyr.accuracy(22.88928, 22.63339)

MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE

0.25589000 0.25589000 NA 0.01130586 0.01130586 1.13058627 1.12423107 0.01130586

**(This approach provides better result)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MAE | RMSE | MASE | MRAE | GMRAE | MAPE | sMAPE | uMbRAE |
| 0.106300000 | 0.106300000 | NA | 0.002342557 | 0.002342557 | 0.234255672 | 0.233981614 | 0.002342557 |

**Case 2** Complete row missing (50 th row of original dataset)

1. Filling with the map estimate of the right column to the missing column of original matrix dataset like: map\_estimate(filled\_matrix[,51])
2. Filling with the mean of the right column to the missing column of original matrix dataset like mean(filled\_matrix[,51])

Compare the Evaluation Results

1. First solution approach provides the best result

**Code**

matrix3=matrix1

matrix3

matrix3\_NA=matrix3

matrix3\_NA[50,]<-0

matrix3\_NA[50,]

map\_estimate(matrix3[51,])

matrix3\_NA[50,]<-42.32

mean(matrix3\_NA)

mean(matrix1)

mse(45.48407, 45.4833)

mape(45.48407, 45.4833)

rmse(45.48407, 45.4833)

mae(45.48407, 45.4833)

FuzzyR::fuzzyr.accuracy(45.48407, 45.4833)

**Output**

|  |
| --- |
| >map\_estimate(matrix3[51,])  MAP Estimate  Parameter | MAP\_Estimate  ------------------------  x | 42.32  > matrix3\_NA[50,]<-42.32  > mean(matrix3\_NA)  [1] 45.4833  > mean(matrix1)  [1] 45.48407  > mse(45.48407, 45.4833)  [1] 0.0000005929  > mape(45.48407, 45.4833)  [1] 0.0016929  > rmse(45.48407, 45.4833)  [1] 0.00077  > mae(45.48407, 45.4833)  [1] 0.00077  > FuzzyR::fuzzyr.accuracy(45.48407, 45.4833)  MAE RMSE MASE MRAE GMRAE MAPE  0.00077000000 0.00077000000 NA 0.00001692929 0.00001692929 0.00169292905  sMAPE uMbRAE  0.00169291472 0.00001692929 |
|  |
| |  | | --- | | > | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MAE | RMSE | MASE | MRAE | GMRAE | MAPE | sMAPE | uMbRAE |
| 0.00077000000 | 0.00077000000 | NA | 0.00001692929 | 0.00001692929 | 0.00169292905 | 0.00169291472 | 0.00001692929 |

**Case 3**

At Random Position

map\_estimate(matrix1[,1])

map\_estimate(matrix4[,50])

map\_estimate(matrix4[,1])

map\_estimate(matrix4[,100])

matrix4\_NA[zero\_indices]<-51.46 #map\_estimate() of [,50] th column

matrix4\_NA

mean(matrix1)

mean(matrix4\_NA)

mse(45.48407, 45.55337)

mape(45.48407,45.55337)

rmse(45.48407, 45.55337)

mae(45.48407, 45.55337)

FuzzyR::fuzzyr.accuracy(45.48407, 45.55337)

**Output**

|  |
| --- |
| > mean(matrix1)  [1] 45.48407  > mean(matrix4\_NA)  [1] 45.55337  > mse(45.48407, 45.55337)  [1] 0.00480249  > mape(45.48407,45.55337)  [1] 0.152361  > rmse(45.48407, 45.55337)  [1] 0.0693  > mae(45.48407, 45.55337)  [1] 0.0693  > FuzzyR::fuzzyr.accuracy(45.48407, 45.55337)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.069300000 0.069300000 NA 0.001521292 0.001521292 0.152129250 0.152245054 0.001521292 |
|  |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | >   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | MAE | RMSE | MASE | MRAE | GMRAE | MAPE | sMAPE | uMbRAE | | 0. 069300000 | 0. 069300000 | NA | 0. 001521292 | 0. 001521292 | 0. 152129250 | 0. 152245054 | 0. 001521292 | | |

map\_estimate(matrix4[,100])

matrix5\_NA

matrix5\_NA[zero\_indices]<-54.94 #map\_estimate() of [,100] th column

matrix5\_NA

mean(matrix1)

mean(matrix5\_NA)

mse(45.48407, 45.58817)

mape(45.48407,45.58817)

rmse(45.48407, 45.58817)

mae(45.48407, 45.58817)

FuzzyR::fuzzyr.accuracy(45.48407, 45.58817)

**Output**

|  |
| --- |
| > mean(matrix1)  [1] 45.48407  > mean(matrix5\_NA)  [1] 45.58817  > mse(45.48407, 45.58817)  [1] 0.01083681  > mape(45.48407,45.58817)  [1] 0.2288713  > rmse(45.48407, 45.58817)  [1] 0.1041  > mae(45.48407, 45.58817)  [1] 0.1041  > FuzzyR::fuzzyr.accuracy(45.48407, 45.58817)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.104100000 0.104100000 NA 0.002283487 0.002283487 0.228348714 0.228609728 0.002283487 |
|  |
| |  | | --- | | > | |

**Code**

map\_estimate(matrix4[,1])

matrix6\_NA[zero\_indices]<-36.41 #map\_estimate() of [,1] th column

matrix6\_NA

mean(matrix1)

mean(matrix6\_NA)

mse(45.48407, 45.40287)

mape(45.48407,45.40287)

rmse(45.48407, 45.40287)

mae(45.48407, 45.40287)

FuzzyR::fuzzyr.accuracy(45.48407, 45.40287)

**Output**

|  |
| --- |
| > mean(matrix1)  [1] 45.48407  > mean(matrix6\_NA)  [1] 45.40287  > mse(45.48407, 45.40287)  [1] 0.00659344  > mape(45.48407,45.40287)  [1] 0.178524  > rmse(45.48407, 45.40287)  [1] 0.0812  > mae(45.48407, 45.40287)  [1] 0.0812  > FuzzyR::fuzzyr.accuracy(45.48407, 45.40287)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.081200000 0.081200000 NA 0.001788433 0.001788433 0.178843320 0.178683538 0.001788433 |
|  |
| |  | | --- | | > | |

**Sensor-2 (Soil Temperature)**

The original data set contains 0 values at various positions initially. So, to fill those 0 values we have 2 cases.

**Case 1**

In first case calculate the map estimate of the positions of the original matrix, having non zero values. so, calculate map estimate of these as (map\_estimate(matrix1[non\_zero\_indices]))

map\_estimate(matrix1[non\_zero\_indices])

MAP Estimate

Parameter | MAP\_Estimate

------------------------

x | 45.00

**Code**

library(Matrix)

library(GLDEX)

options(scipen=999)

#X\_temp <- matrix(rnorm(100), ncol = 2) # 50x2 matrix with random values

rankMatrix(X\_temp)

library(bayestestR)

#x <- read.csv('D:/All\_Papers/Paper\_1\_O1/dataset details')

x<-read.csv('D:/R\_Experiments/Soil\_Temperature\_DS.csv')

# Convert the data frame to a matrix

x\_matrix <- as.matrix(x, nrow=100000,ncol=1)

# Convert the matrix to an array

x\_array <- array(x\_matrix, dim = dim(x\_matrix))

# Display the array

print(x\_array)

print(x\_matrix)

# Convert to a 2D matrix (e.g., 150 rows and 20 columns)

matrix1 <- matrix(x\_array, nrow = 1000, ncol = 100)

matrix1

matrix2=matrix1

matrix2

# Fetch and display all the index positions containing zero

zero\_indices <- which(matrix1 == 0, arr.ind = TRUE)

print("Index positions containing zero:")

print(zero\_indices)

# Calculate and display the total number of index positions containing zero

total\_zeros <- nrow(zero\_indices)

print(paste("Total number of index positions containing zero:", total\_zeros))

#fun.zero.omit(matrix1)

# Find the index positions of non-zero elements

non\_zero\_indices <- which(matrix1 != 0, arr.ind = TRUE)

# Display the index positions

print(non\_zero\_indices)

map\_estimate(matrix1[non\_zero\_indices])

rankMatrix(matrix1)

rankMatrix(matrix2)

constant\_value <- 45.00

matrix2[zero\_indices]<-constant\_value #map\_estimate() of non\_zero\_indices

matrix2

rankMatrix(matrix2)

mean(matrix2) # 23.4793

mean(matrix1) #22.5361

mse(22.5361, 23.4793)

mape(22.5361, 23.4793)

rmse(22.5361, 23.4793)

mae(22.5361, 23.4793)

FuzzyR::fuzzyr.accuracy(22.5361, 23.4793)

> mse(22.5361, 23.4793)

[1] 0.8896262

> mape(22.5361, 23.4793)

[1] 4.185285

> rmse(22.5361, 23.4793)

[1] 0.9432

> mae(22.5361, 23.4793)

[1] 0.9432

> FuzzyR::fuzzyr.accuracy(22.5361, 23.4793)

MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE

0.94320000 0.94320000 NA 0.04017156 0.04017156 4.01715554 4.09949712 0.04017156

**Case 2**

In the second case, calculate the map estimate of the first row in the original matrix that contains no zero entries (Here the 10th row) so, calculate map estimate of it as (map\_estimate(soil\_temperature[10,]), which is a non-zero row present in the original dataset

Apply the above operation upon the dataset.

Code:

library(Matrix)

library(GLDEX)

options(scipen=999)

#X\_temp <- matrix(rnorm(100), ncol = 2) # 50x2 matrix with random values

rankMatrix(X\_temp)

library(bayestestR)

#fun.zero.omit(c(0,1,2,3,4,0,2))

#x <- read.csv('D:/All\_Papers/Paper\_1\_O1/dataset details')

x<-read.csv('D:/R\_Experiments/Soil\_Temperature\_DS.csv')

print(x[1,1])

# Convert the data frame to a matrix

x\_matrix <- as.matrix(x, nrow=100000,ncol=1)

x\_matrix

# Convert the matrix to an array

x\_array <- array(x\_matrix, dim = dim(x\_matrix))

# Display the array

print(x\_array)

print(x\_matrix)

x\_matrix[100000,1]

# Convert to a 2D matrix (e.g., 150 rows and 20 columns)

matrix1 <- matrix (x\_array, nrow = 1000, ncol = 100)

matrix1

matrix2=matrix1

matrix2

# Fetch and display all the index positions containing zero

zero\_indices <- which(matrix1 == 0, arr.ind = TRUE)

print("Index positions containing zero:")

print(zero\_indices)

print(matrix2[162, 23])

# Calculate and display the total number of index positions containing zero

total\_zeros <- nrow(zero\_indices)

print(paste("Total number of index positions containing zero:", total\_zeros))

#fun.zero.omit(matrix1)

rankMatrix(matrix1)

rankMatrix(matrix2)

# Function to check which rows contain no zeros and show those rows

check\_rows\_without\_zeros <- function(matrix) {

rows\_without\_zeros <- apply(matrix, 1, function(row) all(row != 0))

no\_zero\_rows <- which(rows\_without\_zeros)

if (length(no\_zero\_rows) == 0) {

cat("All rows contain zero values.\n")

} else {

cat("Rows containing no zero values:\n")

print(no\_zero\_rows)

cat("Displaying rows with no zeros (showing first 10 rows for brevity):\n")

print(matrix[no\_zero\_rows[1:10], ])

}

}

# Check the rows and show the ones containing no zeros

check\_rows\_without\_zeros(matrix2)

matrix1[10,]

map\_estimate(matrix1[10,])

# fill the map\_estimate value of 10th row i.e. 16.86 into the places of the matrix where zero is present in the original matrix

# Define the constant value to replace zeros

constant\_value <- 16.85

matrix2[zero\_indices]<-constant\_value #map\_estimate() of [10,] th row

mean(matrix2)

mean(matrix1)

mse(22.5361, 22.88928)

mape(22.5361, 22.88928)

rmse(22.5361, 22.88928)

mae(22.5361, 22.88928)

FuzzyR::fuzzyr.accuracy(22.5361, 22.88928)

**Output**

|  |
| --- |
| > map\_estimate(matrix1[10,])  MAP Estimate  Parameter | MAP\_Estimate  ------------------------  x | 16.85  > # fill the map\_estimate value of 10th row i.e. 16.86 into the places of the matrix where  zero is present in the original matrix  >  > # Define the constant value to replace zeros  > constant\_value <- 16.85  > matrix2[zero\_indices]<-constant\_value #map\_estimate() of [10,] th row  > mean(matrix2)  [1] 22.88928  > mean(matrix1)  [1] 22.5361  >  > mse(22.5361, 22.88928)  [1] 0.1247361  > mape(22.5361, 22.88928)  [1] 1.567174  > rmse(22.5361, 22.88928)  [1] 0.35318  > mae(22.5361, 22.88928)  [1] 0.35318  > FuzzyR::fuzzyr.accuracy(22.5361, 22.88928)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.35318000 0.35318000 NA 0.01542993 0.01542993 1.54299305 1.55498974 0.01542993 |
| After perform comparision between above case 1 and case 2, we found that case 2 provides best result. So we cosider the case 2 as proposed approach for filling the missing or zero valued original matrix |
| |  | | --- | |  | |

**Sensor 2 Case 3**

Let Complete column is missing in the original sensor dataset (Recovery solution)

**Code:**

filled\_matrix=matrix2

filled\_matrix

matrix2\_NA=matrix2

matrix2\_NA[,50]

# Step 3: Set the values at a particular column position to 0 say column 50 (Let column 50 is missing)

matrix2\_NA[,50] <- 0

# Print the modified matrix

print(matrix2\_NA)

#map\_estimate(original\_matrix[, 4])

map\_estimate(matrix2[,51])

matrix2\_NA[,50]<-17.90 #value of map\_estimate(matrix2[,51])

matrix2\_NA

mean(matrix2\_NA)

mean(matrix2)

mse(22.88928, 22.83544)

mape(22.88928, 22.83544)

rmse(22.88928, 22.83544)

mae(22.88928, 22.83544)

FuzzyR::fuzzyr.accuracy(22.88928, 22.83544)

**Output:**

|  |
| --- |
| mean(matrix2\_NA)  [1] 22.83544  > mean(matrix2)  [1] 22.88928  > mse(22.88928, 22.83544)  [1] 0.002898746  > mape(22.88928, 22.83544)  [1] 0.2352193  > rmse(22.88928, 22.83544)  [1] 0.05384  > mae(22.88928, 22.83544)  [1] 0.05384  > FuzzyR::fuzzyr.accuracy(22.88928, 22.83544)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE uMbRAE  0.053840000 0.053840000 NA 0.002357739 0.002357739 0.235773867 0.235496248 0.002357739 |
|  |
| |  | | --- | | > | |

**Step**

**Sensor 2 Case 3**

# Step 4:Set the values at a particular row position to 0 say row 50 (Let complete row 50 is missing)

filled\_matrix

rankMatrix(filled\_matrix)

matrix3=filled\_matrix

matrix3

matrix3\_NA=matrix3

matrix3\_NA[50,]<-0

matrix3\_NA[50,]

map\_estimate(filled\_matrix[51,])

matrix3\_NA[50,]<-13.26

mean(matrix3\_NA)

mean(filled\_matrix)

mse(22.88928, 22.88017)

mape(22.88928, 22.88017)

rmse(22.88928, 22.88017)

mae(22.88928, 22.88017)

FuzzyR::fuzzyr.accuracy(22.88928, 22.88017)

**OUTPUT**

|  |
| --- |
| mean(matrix3\_NA)  [1] 22.88017  > mean(matrix1)  [1] 22.5361  > mean(filled\_matrix)  [1] 22.88928  > mean(matrix3)  [1] 22.88928  > mean(filled\_matrix)  [1] 22.88928  > mse(22.88928, 22.88017)  [1] 0.0000829921  > mape(22.88928, 22.88017)  [1] 0.03980029  > rmse(22.88928, 22.88017)  [1] 0.00911  > mae(22.88928, 22.88017)  [1] 0.00911  > FuzzyR::fuzzyr.accuracy(22.88928, 22.88017)  MAE RMSE MASE MRAE GMRAE MAPE sMAPE  0.0091100000 0.0091100000 NA 0.0003981614 0.0003981614 0.0398161377 0.0398082127  uMbRAE  0.0003981614 |
|  |
| |  | | --- | | > | |