Package ‘RMCDA’

February 19, 2025

Title Multi-Criteria Decision Analysis in R

Version 0.3

Author Annice Najafi

Maintainer [Annice Najafi] <annicenajafi27@gmail.com>

Description This package provides different methods of multi-criteria decision analysis.

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports dplyr,
stats,
igraph,
fmsb,
lpSolve,
MASS,
matlib,
nloptr,
matrixStats,
pracma

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

R topics documented:

apply.AHP *Apply AHP on the matrices*

Description

Apply AHP on the matrices

Usage

```
apply.AHP(A, comparing.competitors)
```

Arguments

`A`	the matrix containing information related to pairwise comparisons of criteria
`comparing.competitors`	the list of matrices related to pairwise comparisons of competitors for each criteria

Value

a list containing I. The weight of each criteria II. The criteria alternative unweighted matrix III. The weighted scores matrix IV. Competitor final scores

Examples

```
data <- read.csv(system.file("extdata", "AHP_input_file.csv", package = "RMCDA"), header=FALSE)
mat.lst <- read.csv.AHP.matrices(data)
mat.lst[[1]]->A
mat.lst[[2]]->comparing.competitors
results<- apply.AHP(A, comparing.competitors)
```

apply.ANP *Apply Analytical Network Process (ANP) on data*

Description

Apply Analytical Network Process (ANP) on data

Usage

```
apply.ANP(A, comparing.competitors, power)
```

Arguments

`A`	the matrix containing information related to pairwise comparisons of criteria
`comparing.competitors`	the list of matrices related to pairwise comparisons of competitors for each criteria
`power`	the power value of the supermatrix

Value

the limiting super matrix

Examples

```
data <- read.csv(system.file("extdata", "AHP_input_file.csv", package = "RMCDA"), header=FALSE)
mat.lst <- read.csv.AHP.matrices(data)
mat.lst[[1]]->A
mat.lst[[2]]->comparing.competitors
apply.ANP(A, comparing.competitors, 2)
```

---

apply.ARAS *Apply Additive Ratio Assessment (ARAS)*

Description

Apply Additive Ratio Assessment (ARAS)

Usage

```
apply.ARAS(mat, weights, beneficial.vector)
```

Arguments

mat	is a matrix and contains the values for different properties of different alternatives
weights	are the weights of each property in the decision making process
beneficial.vector	is a vector that contains the column number of beneficial properties.

Value

a vector containing the utility degree related to each alternative, higher utility indicates better ranking.

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)
colnames(mat)<-c("Toughness Index","Yield Strength","Young's Modulus",
                 "Density","Thermal Expansion","Thermal Conductivity","Specific Heat")
rownames(mat)<-c("AI 2024-T6", "AI 5052-O","SS 301 FH",
"SS 310-3AH",
"Ti-6AI-4V",
"Inconel 718",
"70Cu-30Zn")
weights <- c(0.28, 0.14, 0.05, 0.24, 0.19, 0.05, 0.05)
beneficial.vector<-c(1,2,3)
apply.ARAS(mat, weights, beneficial.vector)
```

apply.BORDA *Function to apply BORDA method to data*

Description

This function implements a simple Borda count approach for a decision matrix. It computes a rank for each criterion and then sums these ranks for each alternative. By specifying which columns are beneficial (i.e., higher values preferred), it automatically treats the remaining columns as non-beneficial (i.e., lower values preferred).

Usage

```
apply.BORDA(mat, beneficial.vector)
```

Arguments

`mat`	A numeric matrix or data frame. Rows represent alternatives, columns represent criteria.
`beneficial.vector`	An integer vector containing the column indices of criteria that are beneficial (profit). All other columns are treated as non-beneficial (cost).

Value

A numeric vector of total Borda scores for each alternative, in the original row order.

Examples

```
# Create a small decision matrix (4 alternatives x 3 criteria)
mat <- matrix(c(
  5, 9, 2,
  7, 3, 8,
  6, 5, 4,
  4, 7, 9
), nrow = 4, byrow = TRUE)

beneficial.vector <- c(1, 3)


borda_scores <- apply.BORDA(mat, beneficial.vector)
borda_scores
```

apply.BWM *Function for applying the Best-Worst Method*

Description

Function for applying the Best-Worst Method

Usage

```
apply.BWM(
  criteria.lst,
  worst.criteria,
  best.criteria,
  best.criteria.preference,
  worst.criteria.preference
)
```

Arguments

`criteria.lst` list of criteria

`worst.criteria` the worst criteria

`best.criteria` the best criteria

`best.criteria.preference` the comparison of the best criteria to others

`worst.criteria.preference` the comparison of the worst criteria to others

Value

the result of BWM

Examples

```
criteria.lst <- c("C1", "C2", "C3")
worst.criteria <- "C1"
best.criteria <- "C3"
best.criteria.preference <- c(8, 2, 1)
worst.criteria.preference <- c(1, 5, 8)
apply.BWM(criteria.lst, worst.criteria, best.criteria, best.criteria.preference, worst.criteria.preference)
```

`apply.CILOS` *Apply CILOS Weighting Method*

Description

Apply CILOS Weighting Method

Usage

```
apply.CILOS(mat, beneficial.vector)
```

Arguments

`mat` A numeric matrix representing decision criteria values.

`beneficial.vector` A numeric vector indicating the column indices of beneficial criteria.

Value

A numeric vector of calculated weights.

Examples

```
mat <- matrix(
  c(75.5, 95, 770, 187, 179, 239, 237,
    420, 91, 1365, 1120, 875, 1190, 200,
    74.2, 70, 189, 210, 112, 217, 112,
    2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
    21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
    0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
    0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06),
  nrow = 7, byrow = TRUE
)
beneficial.vector <- c(1, 2, 3, 6, 7)
apply.CILOS(mat, beneficial.vector)
```

`apply.COCOSO` *Apply COmbined COmpromise SOlution (COCOSO)*

Description

Apply COmbined COmpromise SOlution (COCOSO)

Usage

```
apply.COCOSO(mat, weights, beneficial.vector)
```

Arguments

`mat`	is a matrix and contains the values for different properties of different alternatives
`weights`	are the weights of each property in the decision making process
`beneficial.vector`	is a vector that contains the column number of beneficial properties.

Value

a vector containing the aggregated appraisal scores

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)
colnames(mat)<-c("Toughness Index","Yield Strength","Young's Modulus",
                 "Density","Thermal Expansion","Thermal Conductivity","Specific Heat")
rownames(mat)<-c("AI 2024-T6", "AI 5052-O","SS 301 FH",
"SS 310-3AH",
"Ti-6AI-4V",
"Inconel 718",
"70Cu-30Zn")
```

```
weights <- c(0.28, 0.14, 0.05, 0.24, 0.19, 0.05, 0.05)
beneficial.vector<-c(1,2,3)
apply.COCOSO(mat, weights, beneficial.vector)
```

---

apply.CODAS *Apply Combinative Distance-based Assessment (CODAS)*

Description

Apply Combinative Distance-based Assessment (CODAS)

Usage

```
apply.CODAS(mat, weights, beneficial.vector, psi)
```

Arguments

mat	is a matrix and contains the values for different properties of different alternatives
weights	are the weights of each property in the decision making process
beneficial.vector	is a vector that contains the column number of beneficial properties.
psi	threshold parameter

Value

a vector containing the calculated quantitative utility

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)
colnames(mat)<-c("Toughness Index","Yield Strength","Young's Modulus",
                 "Density","Thermal Expansion","Thermal Conductivity","Specific Heat")
rownames(mat)<-c("AI 2024-T6", "AI 5052-O","SS 301 FH",
"SS 310-3AH",
"Ti-6AI-4V",
"Inconel 718",
"70Cu-30Zn")
weights <- c(0.28, 0.14, 0.05, 0.24, 0.19, 0.05, 0.05)
beneficial.vector<-c(1,2,3)
psi <- 0.02
apply.CODAS(mat, weights, beneficial.vector, psi)
```

`apply.COPELAND` *Apply Copeland Method*

Description

Apply Copeland Method

Usage

```
apply.COPELAND(mat, beneficial.vector)
```

Arguments

`mat` A numeric matrix containing the values for different properties of different alternatives.

`beneficial.vector` A numeric vector containing the column indices of beneficial criteria. Non-beneficial criteria are assumed to be the remaining columns.

Value

A numeric vector containing the calculated Copeland scores for each alternative.

Examples

```
mat <- matrix(c(80, 60, 90,
                75, 85, 95,
                70, 65, 85,
                60, 75, 80),
              nrow = 4, byrow = TRUE)
colnames(mat) <- c("Criterion 1", "Criterion 2", "Criterion 3")
beneficial.vector <- c(1, 2, 3)
apply.COPELAND(mat, beneficial.vector)
```

`apply.COPRAS` *Apply COmplex PRoportional ASsessment (COPRAS) method*

Description

Apply COmplex PRoportional ASsessment (COPRAS) method

Usage

```
apply.COPRAS(mat, weights, beneficial.vector)
```

Arguments

`mat` is a matrix and contains the values for different properties of different alternatives

`weights` are the weights of each property in the decision making process

`beneficial.vector` is a vector that contains the column number of beneficial properties.

Value

a vector containing the calculated quantitative utility

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)
colnames(mat)<-c("Toughness Index","Yield Strength","Young's Modulus",
"Density","Thermal Expansion","Thermal Conductivity","Specific Heat")
rownames(mat)<-c("AI 2024-T6",
"AI 5052-O",
"SS 301 FH",
"SS 310-3AH",
"Ti-6AI-4V",
"Inconel 718",
"70Cu-30Zn")
weights <- c(0.28, 0.14, 0.05, 0.24, 0.19, 0.05, 0.05)
beneficial.vector<-c(1,2,3)
apply.COPRAS(mat, weights, beneficial.vector)
```

---

apply.CRADIS — *Function to apply CRiteria Aggregation for Decision Information Synthesis (CRADIS)*

Description

Function to apply CRiteria Aggregation for Decision Information Synthesis (CRADIS)

Usage

```
apply.CRADIS(mat, weights, beneficial.vector)
```

Arguments

`mat`	is a matrix containing the values for different properties of different alternatives
`weights`	are the weights of each property in the decision-making process
`beneficial.vector`	is a vector that contains the column numbers of beneficial criteria

Value

a vector containing the preference values for each alternative

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)
colnames(mat) <- c("Toughness Index", "Yield Strength", "Young's Modulus",
                "Density", "Thermal Expansion", "Thermal Conductivity", "Specific Heat")
rownames(mat) <- c("AI 2024-T6", "AI 5052-O", "SS 301 FH",
                   "SS 310-3AH", "Ti-6AI-4V", "Inconel 718", "70Cu-30Zn")
weights <- c(0.28, 0.14, 0.05, 0.24, 0.19, 0.05, 0.05)
beneficial.vector <- c(1, 2, 3)
apply.CRADIS(mat, weights, beneficial.vector)
```

`apply.CRITIC` *Apply CRITIC on comparison matrix*

Description

Apply CRITIC on comparison matrix

Usage

```
apply.CRITIC(A)
```

Arguments

`A`	the matrix A with row names corresponding to alternatives and column names corresponding to criteria

Value

the weight percentages related to matrix A obtained through the CRITIC method

Examples

```
A <- matrix(c(250, 200, 300, 275, 225, 16, 16, 32, 32, 16, 12, 8, 16, 8, 16, 5, 3, 4, 4, 2), nrow=5, ncol=4)
colnames(A)<-c("Price", "Storage space", "Camera", "Looks")
rownames(A)<-paste0("Mobile ", seq(1, 5, 1))
A[,"Price"] <- -A[,"Price"]
apply.CRITIC(A)
```

`apply.DEMATEL` *Apply DEMATEL method*

Description

Apply DEMATEL method

Usage

```
apply.DEMATEL(comparisons.mat)
```

Arguments

`comparisons.mat`
: the matrix containing information related to pairwise comparisons of criteria

Value

a list containing two vectors one holding D-R and the other D+R

Examples

```
comparisons.mat <- matrix(c(0, 3, 3, 4,
1, 0, 2, 1,
1, 2, 0, 2,
1, 2, 1, 0), nrow=4)
rownames(comparisons.mat)<-c("Price/cost", "Storage Space", "Camera", "Processor")
colnames(comparisons.mat)<-c("Price/cost", "Storage Space", "Camera", "Processor")
apply.DEMATEL(comparisons.mat)
```

`apply.EDAS` *Function to apply the Evaluation based on Distance from Average Solution (EDAS) method*

Description

Function to apply the Evaluation based on Distance from Average Solution (EDAS) method

Usage

```
apply.EDAS(mat, weights)
```

Arguments

`mat`	is a matrix and contains the values for different properties of different alternatives. Non-beneficial columns need to have negative values
`weights`	are the weights of each property in the decision making process

Value

the AS_i index from EDAS from which the final ranking can be found

Examples

```
mat <- matrix(c(250, 200, 300, 275, 225,
16, 16, 32, 32, 16,
12, 8, 16, 8, 16,
5, 3, 4, 4, 2), nrow=5)
colnames(mat)<-c("Price/cost", "Storage Space", "Camera", "Looks")
rownames(mat)<-paste0("Mobile", 1:5)
mat[,"Price/cost"]<--mat[,"Price/cost"]
weights <- c(0.35, 0.25, 0.25, 0.15)
apply.EDAS(mat, weights)
```

`apply.ELECTRE1` *Apply ELECTRE I method*

Description

Apply ELECTRE I method

Usage

```
apply.ELECTRE1(mat, weights)
```

Arguments

`mat`	A matrix or data frame where rows represent alternatives and columns represent criteria.
`weights`	A numeric vector of weights for each criterion.

Value

a list containing three matrices, the first one is the intersection of concordance and discordance matrices, the second one is the concordance matrix, and the third one is the discordance matrix.

Examples

```
mat <- matrix(c(25, 10, 30, 20, 30, 10, 15, 20, 30, 30, 30, 10), nrow=3)
colnames(mat)<-c("c1", "c2", "c3", "c4")
rownames(mat)<-c("a1", "a2", "a3")
weights <- c(0.2, 0.15, 0.4, 0.25)

# Apply ELECTRE I method
results <- apply.ELECTRE1(mat, weights)
```

apply.entropy *Find entropy of each criteria*

Description

Find entropy of each criteria

Usage

```
apply.entropy(A)
```

Arguments

A	the matrix A with row names corresponding to alternatives and column names corresponding to criteria

Value

the entropy value corresponding to each criteria

Examples

```
A <- matrix(c(250, 200, 300, 275, 225, 16, 16, 32, 32, 16, 12, 8, 16, 8, 16, 5, 3, 4, 4, 2), nrow=5, ncol=4)
colnames(A)<-c("Price", "Storage space", "Camera", "Looks")
rownames(A)<-paste0("Mobile ", seq(1, 5, 1))
A[,"Price"] <- -A[,"Price"]
apply.entropy(A)
```

apply.FAHP *Apply fuzzy AHP on criteria comparison matrix*

Description

Apply fuzzy AHP on criteria comparison matrix

Usage

```
apply.FAHP(A)
```

Arguments

A	the comparison matrix

Value

the fuzzy weights for each criteria

Examples

```
# example code
data <- read.csv(system.file("extdata", "AHP_input_file.csv", package = "RMCDA"), header=FALSE)
mat.lst <- read.csv.AHP.matrices(data)
mat.lst[[1]]->A
result <- apply.FAHP(A)
```

---

apply.GRA *Apply Grey Relational Analysis (GRA) method*

Description

Apply Grey Relational Analysis (GRA) method

Usage

```
apply.GRA(mat, weights, beneficial.vector, epsilon = 0.5)
```

Arguments

mat	is a matrix containing the values for different properties of different alternatives
weights	are the weights of each property in the decision-making process
beneficial.vector	is a vector containing the column numbers of beneficial properties. Non-beneficial properties are assumed to be the remaining columns.
epsilon	is a parameter for the GRA method, default is 0.5

Value

a vector containing the calculated GRA scores

Examples

```
mat <- matrix(c(80, 60, 90,
                75, 85, 95,
                70, 65, 85,
                60, 75, 80),
              nrow = 4, byrow = TRUE)
colnames(mat) <- c("Criterion 1", "Criterion 2", "Criterion 3")
weights <- c(0.4, 0.3, 0.3)
beneficial.vector <- c(1, 2, 3)
apply.GRA(mat, weights, beneficial.vector)
```

apply.IDOCRIW — *Apply Integrated Determination of Objective Criteria Weights (IDOCRIW) method*

Description

Apply Integrated Determination of Objective Criteria Weights (IDOCRIW) method

Usage

```
apply.IDOCRIW(mat, beneficial.vector)
```

Arguments

`mat`	is a matrix containing the values for different properties of different alternatives
`beneficial.vector`	is a vector containing the column numbers of beneficial criteria

Value

a vector containing the calculated weights for the criteria

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)
colnames(mat) <- c("Toughness Index", "Yield Strength", "Young's Modulus",
"Density", "Thermal Expansion", "Thermal Conductivity", "Specific Heat")
rownames(mat) <- c("AI 2024-T6", "AI 5052-O", "SS 301 FH",
"SS 310-3AH", "Ti-6AI-4V", "Inconel 718", "70Cu-30Zn")
beneficial.vector <- c(1, 2, 3, 6, 7)
apply.IDOCRIW(mat, beneficial.vector)
```

apply.MABAC — *Apply Multi-Attributive Border Approximation Area Comparison (MABAC)*

Description

R implementation of the MABAC method. The MABAC method computes the distance between each alternative and the Boundary Approximation Area (BAA), based on a weighted normalized decision matrix.

Usage

```
apply.MABAC(mat, weights, types)
```

Arguments

`mat`	A numeric matrix. Rows are alternatives; columns are criteria.
`weights`	A numeric vector of weights corresponding to criteria columns. Must sum to 1.
`types`	An integer vector of the same length as `weights`. Use 1 for a profit criterion and -1 for a cost criterion.

Value

A numeric vector with the MABAC preference values for each alternative. A higher value indicates a more preferred alternative.

Examples

```
# Example usage:
mat <- matrix(c(
  22600, 3800, 2,   5, 1.06, 3.00, 3.5,  2.8, 24.5, 6.5,
  19500, 4200, 3,   2, 0.95, 3.00, 3.4,  2.2, 24.0, 7.0,
  21700, 4000, 1,   3, 1.25, 3.20, 3.3,  2.5, 24.5, 7.3,
  20600, 3800, 2,   5, 1.05, 3.25, 3.2,  2.0, 22.5, 11.0,
  22500, 3800, 4,   3, 1.35, 3.20, 3.7,  2.1, 23.0, 6.3,
  23250, 4210, 3,   5, 1.45, 3.60, 3.5,  2.8, 23.5, 7.0,
  20300, 3850, 2,   5, 0.90, 3.25, 3.0,  2.6, 21.5, 6.0
), nrow = 7, byrow = TRUE)

weights <- c(0.146, 0.144, 0.119, 0.121, 0.115, 0.101, 0.088, 0.068, 0.050, 0.048)
types <- c(-1, 1, 1, 1, -1, -1, 1, 1, 1, 1)

apply.MABAC(mat, weights, types)
```

`apply.MACBETH` *Apply MACBETH (Measuring Attractiveness by a Categorical Based Evaluation TecHnique)*

Description

Apply MACBETH (Measuring Attractiveness by a Categorical Based Evaluation TecHnique)

Usage

```
apply.MACBETH(mat, beneficial.vector, weights)
```

Arguments

`mat`	A numeric matrix where rows represent alternatives and columns represent criteria.
`beneficial.vector`	An integer vector containing column indices for the beneficial (larger-is-better) criteria. Columns not in beneficial.vector are treated as non-beneficial (smaller-is-better).
`weights`	A numeric vector of the same length as the number of columns in mat, containing the relative importance weights for each criterion.

Value

A numeric vector V of length nrow(mat), the final attractiveness scores.

Examples

```
# Example matrix: 3 alternatives x 2 criteria
mat <- matrix(c(10, 5,
                12, 4,
                11, 6), nrow=3, byrow=TRUE)

# Suppose first column is beneficial, second is non-beneficial
benef.vec <- c(1)
wts <- c(0.6, 0.4)

# Get MACBETH scores
res <- apply.MACBETH(mat, benef.vec, wts)
print(res)
```

---

apply.MAIRCA — *Apply Multi-Attributive Real Ideal Comparative Analysis (MAIRCA)*

Description

R implementation of the MAIRCA method. The MAIRCA method computes the gap between ideal (theoretical) and empirical ratings to rank alternatives.

Usage

```
apply.MAIRCA(mat, weights, types)
```

Arguments

`mat`	A numeric matrix. Rows are alternatives; columns are criteria.
`weights`	A numeric vector of weights corresponding to criteria columns. Must sum to 1.
`types`	An integer vector of the same length as `weights`. Use 1 for a profit criterion and -1 for a cost criterion.

Value

A numeric vector with the MAIRCA preference values for each alternative. Higher values indicate more preferred alternatives.

Examples

```
# Example usage
mat <- matrix(c(70, 245, 16.4, 19,
                52, 246, 7.3, 22,
                53, 295, 10.3, 25,
                63, 256, 12.0, 8,
                64, 233, 5.3, 17),
              nrow = 5, byrow = TRUE)
```

```
weights <- c(0.04744, 0.02464, 0.51357, 0.41435)
types <- c(1, 1, 1, 1)
apply.MAIRCA(mat, weights, types)
```

`apply.MARA` *Apply the MARA (Magnitude of the Area for the Ranking of Alternatives) Method*

Description

MARA ranks alternatives based on multiple criteria, each weighted. Columns in beneficial.vector are treated as "max" (beneficial), and columns not in beneficial.vector are treated as "min" (cost).

Usage

```
apply.MARA(mat, weights, beneficial.vector)
```

Arguments

`mat`	A numeric matrix with each row an alternative and each column a criterion.
`weights`	A numeric vector of weights for each criterion (same length as number of columns).
`beneficial.vector`	An integer vector of column indices for the beneficial (max) criteria.

Details

The following function is the R implementation of the python function mara from the pyDecision package Source: https://github.com/Valdecy/pyDecision/blob/master/pyDecision/algorithm/mara.py

Value

A numeric vector of MARA scores for each alternative.

Examples

```
# Example
mat <- matrix(c(10, 2,
                20, 4,
                15, 5),
              nrow = 3, byrow = TRUE)
weights <- c(0.7, 0.3)
beneficial.vector <- c(1)  # First column is beneficial (max); second is cost (min)
apply.MARA(mat, weights, beneficial.vector)
```

`apply.MARCOS` *Apply Measurement of Alternatives and Ranking according to Compromise Solution (MARCOS)*

Description

Apply Measurement of Alternatives and Ranking according to Compromise Solution (MARCOS)

Usage

```
apply.MARCOS(mat, weights, beneficial.vector)
```

Arguments

`mat` is a matrix and contains the values for different properties of different alternatives.

`weights` are the weights of each property in the decision-making process.

`beneficial.vector` is a vector that contains the column number of beneficial properties.

Value

a vector containing the aggregated appraisal scores.

Examples

```
mat <- matrix(c(660, 1000, 1600, 18, 1200,
                800, 1000, 1600, 24, 900,
                980, 1000, 2500, 24, 900,
                920, 1500, 1600, 24, 900,
                1380, 1500, 1500, 24, 1150,
                1230, 1000, 1600, 24, 1150,
                680, 1500, 1600, 18, 1100,
                960, 2000, 1600, 12, 1150), nrow = 8, byrow = TRUE)
weights <- c(0.1061, 0.3476, 0.3330, 0.1185, 0.0949)
beneficial.vector <- c(2, 3, 4, 5)  # Columns 2, 3, 4, and 5 are beneficial
apply.MARCOS(mat, weights, beneficial.vector)
```

`apply.MAUT` *Apply Multi-Attribute Utility Theory (MAUT) Method*

Description

Apply Multi-Attribute Utility Theory (MAUT) Method

Usage

```
apply.MAUT(mat, weights, beneficial.vector, utility.functions, step.size = 1)
```

Arguments

`mat` is a matrix containing values for different properties of different alternatives

`weights` are the weights of each property in the decision-making process

`beneficial.vector`
is a vector containing the column numbers of beneficial properties

`utility.functions`
is a vector specifying the utility function for each criterion ('exp', 'step', 'quad', 'log', 'ln')

`step.size` is a numeric value used for the step utility function (default is 1)

Value

a matrix containing the calculated utility scores

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237, 420, 91), nrow = 3, byrow = TRUE)
weights <- c(0.3, 0.5, 0.2)
beneficial.vector <- c(1, 3)
utility.functions <- c("exp", "log", "quad")
step.size <- 1
result <- apply.MAUT(mat, weights, beneficial.vector, utility.functions, step.size)
```

---

apply.MOORA *Apply Multi-Objective Optimization on the basis of Ratio Analysis (MOORA)*

Description

Apply Multi-Objective Optimization on the basis of Ratio Analysis (MOORA)

Usage

```
apply.MOORA(mat, weights, beneficial.vector)
```

Arguments

`mat` is a matrix and contains the values for different properties of different alternatives

`weights` are the weights of each property in the decision making process

`beneficial.vector`
is a vector that contains the column number of beneficial properties.

Value

a vector containing the calculated quantitative utility

Examples

```
mat <- matrix(c(60, 6.35, 6.8, 10, 2.5, 4.5, 3,
0.4, 0.15, 0.1, 0.2, 0.1, 0.08, 0.1,
2540, 1016, 1727.2, 1000, 560, 1016, 177,
500, 3000, 1500, 2000, 500, 350, 1000,
990, 1041, 1676, 965, 915, 508, 920), nrow=7)
colnames(mat)<-c("Load capacity", "Repeatability", "Maximum tip speed",
"Memory capacity", "Manipulator reach")
rownames(mat)<-paste0("A", 1:7)
weights <- c(0.1574, 0.1825, 0.2385, 0.2172, 0.2043)
beneficial.vector <- c(1, 3, 4, 5)
apply.MOORA(mat, weights, beneficial.vector)
```

---

apply.MOOSRA *Multi-objective Optimization on the Basis of Simple Ratio Analysis (MOOSRA)*

Description

Multi-objective Optimization on the Basis of Simple Ratio Analysis (MOOSRA)

Usage

```
apply.MOOSRA(mat, weights, beneficial.vector)
```

Arguments

`mat`	A matrix of decision-making criteria values for different alternatives.
`weight_vector`	A vector of weights for the criteria.
`beneficial.vectorA`	vector of column indices for beneficial criteria.
`show_graph`	A logical indicating whether to display the ranking graph (default = TRUE).
`verbose`	A logical indicating whether to print detailed output (default = TRUE).

Value

A matrix containing the alternatives and their calculated scores, sorted by rank.

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)
weights <- c(0.1, 0.2, 0.3, 0.1, 0.1, 0.1, 0.1)
beneficial.vector<- c(1, 2, 3, 6, 7)
apply.MOOSRA(mat, weights, beneficial.vector)
```

`apply.MULTIMOORA` *Apply MULTIMOORA method*

Description

Apply MULTIMOORA method

Usage

```
apply.MULTIMOORA(mat, beneficial.vector)
```

Arguments

`mat` A matrix of decision-making criteria values.

`beneficial.vector`
A vector containing the column indices of beneficial criteria (1-based indexing).

Value

A list of matrices containing rankings for MOORA, MOORA RP, and MULTIMOORA methods.

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
                420, 91, 1365, 1120, 875, 1190, 200,
                74.2, 70, 189, 210, 112, 217, 112,
                2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53), nrow = 4, byrow = TRUE)
beneficial.vector <- c(1, 3) # Columns 1 and 3 are beneficial
apply.MULTIMOORA(mat, beneficial.vector)
```

`apply.OCRA` *Apply Operational Competitiveness Rating (OCRA) method*

Description

The OCRA method independently evaluates alternatives with respect to beneficial (profit) and non-beneficial (cost) criteria, then combines these evaluations into an overall operational competitiveness rating.

Usage

```
apply.OCRA(mat, weights, beneficial.vector)
```

Arguments

`mat` A numeric matrix. Rows are alternatives; columns are criteria.

`weights` A numeric vector of weights corresponding to criteria columns. Must sum to 1.

`beneficial.vector`
A numeric vector containing the column indices of beneficial (profit) criteria. Non-beneficial criteria are assumed to be the remaining columns.

Value

A numeric vector with the OCRA preference values for each alternative. Higher values indicate a more preferred alternative.

Examples

```
mat <- matrix(c(
  7.7, 256, 7.2, 7.3, 7.3,
  8.1, 250, 7.9, 7.8, 7.7,
  8.7, 352, 8.6, 7.9, 8.0,
  8.1, 262, 7.0, 8.1, 7.2,
  6.5, 271, 6.3, 6.4, 6.1,
  6.8, 228, 7.1, 7.2, 6.5
), nrow = 6, byrow = TRUE)

weights <- c(0.239, 0.225, 0.197, 0.186, 0.153)
beneficial.vector <- c(1, 3, 4, 5)

apply.OCRA(mat, weights, beneficial.vector)
```

`apply.OPA` *Apply Ordinal Priority Approach (OPA)*

Description

This function applies the Ordinal Priority Approach (OPA) to determine the optimal weights for experts, criteria, and alternatives based on expert opinions, ranks, and criterion importance.

Usage

```
apply.OPA(expert.opinion.lst, expert.rank, criterion.rank.lst)
```

Arguments

`expert.opinion.lst`
: A list of matrices where each matrix represents the rankings of alternatives for each criterion as assessed by a particular expert. Each row corresponds to an alternative, and each column corresponds to a criterion.

`expert.rank`
: A numeric vector specifying the rank or weight of importance for each expert.

`criterion.rank.lst`
: A list of numeric vectors where each vector represents the rank or weight of importance for the criteria as assessed by each expert.

Value

A list of matrices where each matrix represents the optimal weights for the alternatives and criteria for a specific expert.

Examples

```
# Input Data
expert.x.alt <- matrix(c(1, 3, 2, 2, 1, 3), nrow = 3)
colnames(expert.x.alt) <- c("c", "q")
rownames(expert.x.alt) <- c("alt1", "alt2", "alt3")

expert.y.alt <- matrix(c(1, 2, 3, 3, 1, 2), nrow = 3)
colnames(expert.y.alt) <- c("c", "q")
rownames(expert.y.alt) <- c("alt1", "alt2", "alt3")

expert.opinion.lst <- list(expert.x.alt, expert.y.alt)
expert.rank <- c(1, 2)  # Ranks of experts

# Criterion ranks for each expert
criterion.x.rank <- c(1, 2)
criterion.y.rank <- c(2, 1)  # Adjusted criterion rank for expert y
criterion.rank.lst <- list(criterion.x.rank, criterion.y.rank)

# Apply OPA
weights <- apply.OPA(expert.opinion.lst, expert.rank, criterion.rank.lst)
print(weights)
```

`apply.ORESTE` *Apply the ORESTE (Organisation Rangement Et SynThèsE de données relationnelles) Method*

Description

Criteria with indexes in beneficial.vector are interpreted as beneficial (maximize), whereas others are cost-type (minimize). Rankings are performed for both the data matrix and the weights, then combined in the ORESTE manner.

Usage

```
apply.ORESTE(mat, weights, beneficial.vector, alpha = 0.4)
```

Arguments

`mat`	A numeric matrix with each row representing an alternative and each column a criterion.
`weights`	A numeric vector of weights for each criterion (same length as number of columns).
`beneficial.vector`	An integer vector of column indices specifying which criteria are "max" (beneficial).
`alpha`	A numeric parameter controlling the relative weight of data-based and weight-based ranks.

Value

A numeric vector of ORESTE scores (summed ranks) for each alternative.

Examples

```
mat <- matrix(c(10, 2,
                20, 4,
                15, 5),
              nrow = 3, byrow = TRUE)
weights <- c(0.7, 0.3)
beneficial.vector <- c(1)   # 1st column "max", 2nd column "min"

apply.ORESTE(mat, weights, beneficial.vector, alpha = 0.4)
```

`apply.PIV` *Apply Proximity Indexed Value (PIV) method*

Description

Apply Proximity Indexed Value (PIV) method

Usage

```
apply.PIV(mat, weights, beneficial.vector)
```

Arguments

`mat`	A numeric matrix containing the values for different properties of different alternatives.
`weights`	A numeric vector containing the weights of each property.
`beneficial.vector`	A numeric vector containing the column indices of beneficial criteria. Non-beneficial criteria are assumed to be the remaining columns.

Value

A numeric vector containing the calculated PIV scores for each alternative.

Examples

```
mat <- matrix(c(80, 60, 90,
                75, 85, 95,
                70, 65, 85,
                60, 75, 80),
              nrow = 4, byrow = TRUE)
colnames(mat) <- c("Criterion 1", "Criterion 2", "Criterion 3")
weights <- c(0.4, 0.3, 0.3)
beneficial.vector <- c(1, 2, 3)
apply.PIV(mat, weights, beneficial.vector)
```

`apply.po.ranking` *Apply Pre-Order Ranking (partial-order analysis)*

Description

This function is an R translation of the Python po.ranking() function It merges alternatives that are 'I' (indifferent), constructs a 0/1 partial-order matrix from 'P+' entries, sorts the alternatives by row sums, and then removes transitive edges.

Usage

```
apply.po.ranking(partial.order.str)
```

Arguments

`partial.order.str`
: An n x n character matrix containing pairwise relations. The main relation codes are:
  - "P+": The row alternative strictly dominates the column alternative.
  - "I": The two alternatives are indifferent.
  - "R", "-", or other placeholders can appear but are less critical here.

Details

The function is an R implementation of the pre-order rank and regime function in the pyDecision package Source: https://github.com/Valdecy/pyDecision/blob/master/pyDecision/algorithm/regime.py

Value

A list with elements:

- partial.order.str: An updated partial.order.str after merges. Dimensions may be smaller than the input.
- partial.order.mat: An n' x n' numeric matrix of 0/1, where 1 indicates 'P+'.
- alts: A character vector of alternative labels, possibly merged (e.g., "a2; a1").
- alts_rank: The final ordering of alternatives from most dominating to least dominating.
- rank: A 0/1 matrix after removing transitive edges.

Examples

```
# Create a small 3x3 partial-order matrix
po_str <- matrix(c("P+", "P+", "R",
                   "R",   "-",   "I",
                   "R",   "I",   "-"), nrow=3, byrow=TRUE)

# Apply the pre-order ranking
res <- apply.po.ranking(po_str)

# View partial-order matrix and sorted alternatives
print(res$partial.order.mat)
print(res$alts_rank)
```

apply.PROMETHEE *Function for applying PROMOTHEE I or II*

Description

Function for applying PROMOTHEE I or II

Usage

```
apply.PROMETHEE(A, weights, type = "II")
```

Arguments

`A`	the comparison matrix with the row names indicating the alternatives and colnames indicating the criteria.
`weights`	the weights of criteria.

Value

the results of PROMOTHEE

Examples

```
A <- matrix(c(250, 200, 300, 275, 16, 16, 32, 32, 12, 8, 16, 8, 5, 3, 4, 2), nrow=4)
rownames(A)<-c("Mobile 1", "Mobile 2", "Mobile 3", "Mobile 4")
colnames(A)<-c("Price", "Memory", "Camera", "Looks")
weights <- c(0.35, 0.25, 0.25, 0.15)
apply.PROMETHEE(A, weights)
```

apply.PSI *Apply Preference Selection Index (PSI) method*

Description

Apply Preference Selection Index (PSI) method

Usage

```
apply.PSI(mat, beneficial.vector)
```

Arguments

`mat`	A numeric matrix containing the values for different properties of different alternatives.
`beneficial.vector`	A numeric vector containing the column indices of beneficial criteria. Non-beneficial criteria are assumed to be the remaining columns.

Value

A numeric vector containing the calculated PSI scores for each alternative.

Examples

```
mat <- matrix(c(80, 60, 90,
               75, 85, 95,
               70, 65, 85,
               60, 75, 80),
             nrow = 4, byrow = TRUE)
colnames(mat) <- c("Criterion 1", "Criterion 2", "Criterion 3")
beneficial.vector <- c(1, 2, 3)
apply.PSI(mat, beneficial.vector)
```

---

apply.RAFSI — *Ranking of Alternatives through Functional mapping of criterion sub-intervals into a Single Interval (RAFSI)*

Description

Ranking of Alternatives through Functional mapping of criterion sub-intervals into a Single Interval (RAFSI)

Usage

```
apply.RAFSI(
  mat,
  weights,
  beneficial.vector,
  ideal = NULL,
  anti_ideal = NULL,
  n_i = 1,
  n_k = 6
)
```

Arguments

`mat`	A numeric matrix or data frame with rows = alternatives, columns = criteria
`weights`	A numeric vector of weights (one per criterion)
`beneficial.vector`	A numeric vector that stores the column indices of all beneficial (i.e., "max") criteria. Columns not in `beneficial.vector` are treated as "min".
`ideal`	A numeric vector of ideal values for each criterion (optional)
`anti_ideal`	A numeric vector of anti-ideal values for each criterion (optional)
`n_i`	Lower bound in the functional mapping (default = 1)
`n_k`	Upper bound in the functional mapping (default = 6)

Value

A numeric vector of final RAFSI scores, one per row of `mat`.

Examples

```
mat <- matrix(c(3, 2, 5,
4, 3, 2,
1, 6, 4),
nrow = 3, byrow = TRUE)
weights <- c(0.3, 0.5, 0.2)
beneficial.vector <- c(1, 2)
apply.RAFSI(mat, weights, beneficial.vector,   n_i = 1, n_k = 6)
```

---

apply.REGIME — *Apply REGIME method (using a beneficial.vector)*

Description

This function implements the REGIME method of pairwise comparisons to produce a character matrix (cp.matrix) that marks each pair of alternatives as either "P+" (row dominates column), "I" (indifferent), or "-" (for diagonals).

Usage

```
apply.REGIME(mat, beneficial.vector, weights, doPreOrder = FALSE)
```

Arguments

`mat`	A numeric matrix of size n x m (n alternatives, m criteria).
`beneficial.vector`	An integer vector of columns that are beneficial ("max"). All other columns are assumed to be "min".
`weights`	A numeric vector of length m, containing weights for each criterion.
`doPreOrder`	A logical. If TRUE, the function also calls apply.po.ranking on the resulting cp.matrix and returns both the matrix and the partial-order results in a list.

Details

It uses a beneficial.vector of column indices for "max" criteria. Columns not in beneficial.vector are treated as "min". The function can optionally run apply.po.ranking on the resulting matrix for partial-order analysis.

1. Weights Normalization: We first normalize the weights so their sum equals 1.
2. Pairwise Comparison Matrix (g_ind):
   - For each pair of alternatives and each criterion:
     - If the criterion is beneficial (maximization) and the value for one alternative is greater than or equal to the value for another alternative, the weight for that criterion is added to the pair's comparison score (g_ind). Otherwise, the weight is subtracted from the score.
     - If the criterion is non-beneficial (minimization) and the value for one alternative is less than the value for another alternative, the weight is added to the score. Otherwise, the weight is subtracted.
3. cp.matrix:

  - "P+" indicates that one alternative dominates another if the comparison score (g_ind) is greater than 0.
  - "I" indicates that the alternatives are indifferent if the comparison score is 0 or if the scores for both directions are equal.
  - "-" is assigned to diagonal entries, where the alternatives are compared with themselves.
4. If doPreOrder = TRUE, the function calls apply.po.ranking on cp.matrix to merge indifferent alternatives ("I") and construct a partial order.

Value

- If doPreOrder = FALSE, returns an n x n character matrix cp.matrix.
- If doPreOrder = TRUE, returns a list with two elements:
  - cp.matrix: the character matrix
  - po.result: the output from apply.po.ranking

Examples

```
# Example data: 3 alternatives x 2 criteria
mat <- matrix(c(10, 5,
                12, 4,
                11, 6), nrow = 3, byrow = TRUE)

# Suppose first column is beneficial, second is non-beneficial
benef.vec <- c(1)  # means col1 is "max", col2 is "min"
wts <- c(0.6, 0.4)

# Call apply.REGIME without partial-order
regime.out <- apply.REGIME(mat, benef.vec, wts, doPreOrder = FALSE)
print(regime.out)

# Or with partial-order
regime.out2 <- apply.REGIME(mat, benef.vec, wts, doPreOrder = TRUE)
print(regime.out2$cp.matrix)
print(regime.out2$po.result)
```

`apply.RIM` *Function to apply Reference Ideal Method (RIM) Note: function is rewritten from the MCDM package to match the formatting of the R RMCDA package SOURCE: https://github.com/cran/MCDM/blob/master/R/RIM.R*

Description

The apply.RIM function implements the Reference Ideal Method (RIM) for multi-criteria decision making (MCDM) problems, allowing for degenerate intervals, i.e. cases where A == C or D == B.

Usage

```
apply.RIM(mat, weights, AB, CD)
```

Arguments

`mat` A matrix m x n containing the values of the m alternatives for the n criteria.

`weights` A numeric vector of length n, containing the weights for the criteria. The sum of the weights must be equal to 1.

`AB` A matrix (2 x n), where the first row of AB corresponds to the A extreme, and the second row of AB corresponds to the B extreme of the domain (universe of discourse) for each criterion.

`CD` A matrix (2 x n), where the first row of CD corresponds to the C extreme, and the second row of CD corresponds to the D extreme of the ideal reference for each criterion.

Degenerate intervals:

1. If the first element of AB matches the first element of CD, then the interval between A and C collapses to a point.
    - Any value x within this range is treated under a fallback rule:
        - If x equals both A and C, the normalized value is set to 1.
        - Otherwise, the normalized value is set to 0.
2. If the second element of CD matches the second element of AB, then the interval between D and B collapses to a point.
    - A similar fallback applies:
        - If x equals both D and B, the normalized value is set to 1.
        - Otherwise, the normalized value is set to 0.

These fallback rules ensure the function does not stop but, instead, issues a warning and assigns a default. Adjust these defaults if your MCDM context requires different handling.

Value

A data frame containing:

- Alternatives: The index of each alternative.
- R: The R index (score) for each alternative.
- Ranking: The ranking of the alternatives based on the R score.

Reference: Cables, E.; Lamata, M.T.; Verdegay, J.L. (2016). RIM-reference ideal method in multicriteria decision making. Information Science, 337-338, 1-10.

Examples

```
# Example decision matrix
mat <- matrix(
  c(30,40,25,27,45,0,
    9,0,0,15,2,1,
    3,5,2,3,3,1,
    3,2,3,3,3,2,
    2,2,1,4,1,2),
  nrow = 5, ncol = 6, byrow = TRUE
)

#Example weights vector (must sum to 1)
weights <- c(0.2262,0.2143,0.1786,0.1429,0.119,0.119)
```

```
#Example AB matrix
AB <- matrix(
  c(23,60,0,15,0,10,
    1,3,1,3,1,5),
  nrow = 2, ncol = 6, byrow = TRUE
)

#Example CD matrix
CD <- matrix(
  c(30,35,10,15,0,0,
    3,3,3,3,4,5),
  nrow = 2, ncol = 6, byrow = TRUE
)


apply.RIM(mat, weights, AB, CD)
```

`apply.ROV` *Apply Range of Value (ROV) method*

Description

Apply Range of Value (ROV) method

Usage

```
apply.ROV(mat, weights, beneficial.vector)
```

Arguments

`mat`	A numeric matrix containing the values for different properties of different alternatives.
`weights`	A numeric vector containing the weights of each property.
`beneficial.vector`	A numeric vector containing the column indices of beneficial criteria. Non-beneficial criteria are assumed to be the remaining columns.

Value

A numeric vector containing the calculated ROV scores for each alternative.

Examples

```
mat <- matrix(c(80, 60, 90,
                75, 85, 95,
                70, 65, 85,
                60, 75, 80),
              nrow = 4, byrow = TRUE)
colnames(mat) <- c("Criterion 1", "Criterion 2", "Criterion 3")
weights <- c(0.4, 0.3, 0.3)
beneficial.vector <- c(1, 2, 3)
apply.ROV(mat, weights, beneficial.vector)
```

apply.SAW *Apply Simple Additive Weighting Method (SAW)*

Description

Apply Simple Additive Weighting Method (SAW)

Usage

```
apply.SAW(mat, weights, beneficial.vector)
```

Arguments

mat	is a matrix and contains the values for different properties of different alternatives
weights	are the weights of each property in the decision making process
beneficial.vector	is a vector that contains the column number of beneficial properties.

Value

a vector containing the score and corresponding ranking for the SAW function

Examples

```
mat <- matrix(c(60, 6.35, 6.8, 10, 2.5, 4.5, 3,
0.4, 0.15, 0.1, 0.2, 0.1, 0.08, 0.1,
2540, 1016, 1727.2, 1000, 560, 1016, 177,
500, 3000, 1500, 2000, 500, 350, 1000,
990, 1041, 1676, 965, 915, 508, 920), nrow=7)
colnames(mat)<-c("Load capacity", "Repeatability", "Maximum tip speed",
"Memory capacity", "Manipulator reach")
rownames(mat)<-paste0("A", 1:7)
weights <- c(0.1574, 0.1825, 0.2385, 0.2172, 0.2043)
beneficial.vector <- c(1, 3, 4, 5)
apply.SAW(mat, weights, beneficial.vector)
```

apply.SBWM *Function for applying the Stratified Best-Worst Method (SBWM)*

Description

Function for applying the Stratified Best-Worst Method (SBWM)

Usage

```
apply.SBWM(
  comparison.mat,
  others.to.worst,
  others.to.best,
  state.worst.lst,
  state.best.lst,
  likelihood.vector
)
```

Arguments

`comparison.mat` the comparison matrix containing the alternatives as column names and the criteria as row names.

`others.to.worst` the comparison of the criteria to the worst criteria for each state, column names should be states and the row names are criteria

`others.to.best` the comparison of the criteria to the best criteria for each state, column names should be states and the row names are criteria

`state.worst.lst` the vector containing the name of the worst criteria in each state

`state.best.lst` the vector containing the name of the best criteria in each state

`likelihood.vector` the vector containing the likelihood of being in each state.

Value

the result of SBWM

Examples

```
data <- read.csv(system.file("extdata", "stratified_BWM_case_study_I_example.csv", package = "RMCDA"), header
mat.lst <- read.csv.SBWM.matrices(data)
comparison.mat <- mat.lst[[1]]
others.to.worst <- mat.lst[[2]]
others.to.best <- mat.lst[[3]]
state.worst.lst <- mat.lst[[4]]
state.best.lst <- mat.lst[[5]]
likelihood.vector <- mat.lst[[6]]
apply.SBWM(comparison.mat, others.to.worst, others.to.best, state.worst.lst, state.best.lst, likelihood.vect
```

---

apply.SECA — *Apply Simultaneous Evaluation of Criteria and Alternatives (SECA) method*

Description

Apply Simultaneous Evaluation of Criteria and Alternatives (SECA) method

Usage

```
apply.SECA(mat, beneficial.vector, beta = 3)
```

Arguments

`mat`	A numeric matrix containing the values for different properties of different alternatives.
`beneficial.vector`	A numeric vector containing the column indices of beneficial properties. Non-beneficial properties are assumed to be the remaining columns.
`beta`	A numeric value controlling the balance between criteria variability and similarity. Default is 3.

Value

A numeric vector containing the calculated weights for each criterion.

Examples

```
mat <- matrix(c(80, 60, 90,
                75, 85, 95,
                70, 65, 85,
                60, 75, 80),
              nrow = 4, byrow = TRUE)
colnames(mat) <- c("Criterion 1", "Criterion 2", "Criterion 3")
beneficial.vector <- c(1, 2, 3)
apply.SECA(mat, beneficial.vector)
```

`apply.SMART` *Apply the SMART Method*

Description

This function implements the SMART (Simple Multi-Attribute Rating Technique) method in R.

Usage

```
apply.SMART(
  dataset,
  grades,
  lower,
  upper,
  beneficial.vector,
  graph = TRUE,
  verbose = TRUE
)
```

Arguments

`dataset`	A numeric matrix or data frame of size (n x m), rows = alternatives, columns = criteria.
`grades`	A numeric vector of length `m` (one grade per criterion). They get transformed into weights via $(2^{1/2})^{grades}$ and normalized.
`lower`	A numeric vector of length `m` with lower bounds for each criterion.
`upper`	A numeric vector of length `m` with upper bounds for each criterion.
`beneficial.vector`	A numeric vector containing column indices that are beneficial ("max").

Value

A matrix (or data frame) named result with two columns: The row index (alternative) and the final SMART score for that alternative.

The rows of result are sorted by score in descending order.

Examples

```
# Example usage
data_mat <- matrix(c(10, 20, 15,  7,
                     30,  5,  8, 25),
                   nrow = 2, byrow = TRUE)
# Suppose we have 4 criteria (2 rows, 4 columns)
# We'll treat columns 1, 2, 3 as beneficial, and column 4 as non-beneficial
benef_vec <- c(1, 2, 3)

# Grades for each of 4 criteria
grades <- c(2, 2, 1, 3)
lower  <- c(0, 0, 0,  0)
upper  <- c(40, 40, 40, 40)

# Run SMART
result <- apply.SMART(dataset = data_mat,
                    grades = grades,
                    lower  = lower,
                    upper  = upper,
                    beneficial.vector = benef_vec)

result
```

---

apply.SMCDM — *Apply Stratified Multi-Criteria Decision Making (SMCDM) method*

Description

Apply Stratified Multi-Criteria Decision Making (SMCDM) method

Usage

```
apply.SMCDM(
  comparison.mat,
  state.criteria.probs,
  likelihood.vector,
  independent.events = TRUE
)
```

Arguments

`comparison.mat` the matrix containing alternatives as row names and criteria as column names and corresponding scores as cell values.

`state.criteria.probs` the matrix containing the states as column names and criteria as row names and the corresponding scores as matrix values.

`likelihood.vector`
: the vector containing the likelihood of being in each state.

`independent.event`
: this parameter is set to TRUE by default which indicates only the probability of the occurence of each event is required (strati I and II). If set to FALSE then the user should provide the probabilities of occurrence of all states.

Value

the SMCDM results

Examples

```
data <- read.csv(system.file("extdata", "SMCDM_input.csv", package = "RMCDA"), header=FALSE)
mat.lst <- read.csv.SMCDM.matrices(data)
comparison.mat <- mat.lst[[1]]
state.criteria.probs <- mat.lst[[2]]
likelihood.vector <- mat.lst[[3]]
apply.SMCDM(comparison.mat, state.criteria.probs, likelihood.vector)
```

---

apply.SPOTIS — *Apply the Stable Preference Ordering Towards Ideal Solution (SPOTIS) method*

Description

Apply the Stable Preference Ordering Towards Ideal Solution (SPOTIS) method

Usage

```
apply.SPOTIS(matrix, weights, types, bounds)
```

Arguments

`matrix`	A numeric matrix or data frame where rows represent alternatives and columns represent criteria.
`weights`	A numeric vector of weights for each criterion. The sum of weights must equal 1.
`types`	A numeric vector indicating the type of each criterion: 1 for profit and -1 for cost.
`bounds`	A numeric matrix where each row contains the minimum and maximum bounds for each criterion.

Value

A numeric vector of preference scores for alternatives. Lower scores indicate better alternatives.

Examples

```
# Decision matrix
matrix <- matrix(c(10.5, -3.1, 1.7,
                   -4.7, 0, 3.4,
                   8.1, 0.3, 1.3,
                   3.2, 7.3, -5.3), nrow = 4, byrow = TRUE)

# Criteria bounds
bounds <- matrix(c(-5, 12,
                   -6, 10,
                   -8, 5), nrow = 3, byrow = TRUE)

# Criteria weights
weights <- c(0.2, 0.3, 0.5)

# Criteria types
types <- c(1, -1, 1)

# Apply SPOTIS
preferences <- apply.SPOTIS(matrix, weights, types, bounds)
print(round(preferences, 4))
```

`apply.SRMP` *Apply SRMP (Simple Ranking Method using Reference Profiles) on data*

Description

Apply SRMP (Simple Ranking Method using Reference Profiles) on data

Usage

```
apply.SRMP(evaluations.mat, reference.profiles, weights)
```

Arguments

`evaluations.mat`
: the matrix comparing alternatives based on criteria

`reference.profiles`
: matrix containing reference profile information

`weights`
: of different criteria

Value

alternatives ranked using SRMP

Examples

```
evaluations.mat <- matrix(c(41, 46, 43, -2, -4, -5.5, 4, 2, 3), nrow=3)
colnames(evaluations.mat) <- c("S", "L", "J")
rownames(evaluations.mat) <- c("x", "y", "z")
reference.profiles <- matrix(c(42, 45, -5, -3, 2, 4), nrow=2)
colnames(reference.profiles) <- c("S", "L", "J")
```

```
rownames(reference.profiles) <- c("p1", "p2")
weights <- c(1/3, 1/3, 1/3)
apply.SRMP(evaluations.mat, reference.profiles, weights)
```

`apply.TODIM` *Apply TODIM (TOmada de Decisao Interativa e Multicriterio)*

Description

Implements the core TODIM logic in R

Usage

```
apply.TODIM(mat, weights, beneficial.vector, teta = 1)
```

Arguments

`mat`	A numeric matrix where each row is an alternative and each column is a criterion.
`weights`	A numeric vector of weights for each criterion (same length as number of columns of mat).
`beneficial.vector`	A vector of column indices corresponding to beneficial criteria (i.e., the larger the value, the better). Columns not listed here will be treated as non-beneficial.
`teta`	A numeric scalar in TODIM). Default is 1.

Details

In the TODIM formula, theta acts as an “attenuation factor” or penalty for negative dominance differences. This parameter allows you to adjust how severely negative differences weigh in the final scoring. A common default is 1, but you could experiment with other values if you want to amplify or reduce the penalty effect.

If you set teta = 1, it uses the standard TODIM approach. If you do not want to vary this parameter, you can leave it at its default value of 1.

Value

A numeric vector of rescaled scores, one per alternative (row).

Examples

```
# Small synthetic example
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)

colnames(mat)<-c("Toughness Index","Yield Strength","Young's Modulus",
"Density","Thermal Expansion","Thermal Conductivity","Specific Heat")
```

```
rownames(mat)<-c("AI 2024-T6", "AI 5052-O","SS 301 FH",
"SS 310-3AH","Ti-6AI-4V","Inconel 718","70Cu-30Zn")
weights <- c(0.28, 0.14, 0.05, 0.24, 0.19, 0.05, 0.05)
beneficial.vector<-c(1,2,3)

apply.TODIM(mat, weights, beneficial.vector, teta=1)
```

`apply.TOPSIS` *Apply TOPSIS on matrix A with weight of criteria stored in vector w*

Description

Apply TOPSIS on matrix A with weight of criteria stored in vector w

Usage

```
apply.TOPSIS(A, w)
```

Arguments

A	the matrix A with row names corresponding to alternatives and column names corresponding to criteria
w	the weight vector corresponding to the weight of each criteria

Value

performance scores obtained through TOPSIS

Examples

```
A <- matrix(c(250, 200, 300, 275, 225, 16, 16, 32, 32, 16, 12, 8, 16, 8, 16, 5, 3, 4, 4, 2), nrow=5, ncol=4)
colnames(A)<-c("Price", "Storage space", "Camera", "Looks")
rownames(A)<-paste0("Mobile ", seq(1, 5, 1))
A[,"Price"] <- -A[,"Price"]
apply.TOPSIS(A, c(1/4, 1/4, 1/4, 1/4))
```

`apply.VIKOR` *Function for applying VIKOR to data*

Description

Function for applying VIKOR to data

Usage

```
apply.VIKOR(A, weights, nu = 0.5)
```

Arguments

`A`	the comparison matrix
`weights`	the weights of criteria
`nu`	weight of the maximum utility strategy - set by default to 0.5

Value

a list containing the names of Qi followed by values of Qi, Si, Ri, condition 1, and condition 2.

Examples

```
A <- matrix(c(250, 200, 300, 275, 225, 16, 16, 32, 32, 16, 12, 8, 16, 8, 16, 5, 3, 4, 4, 2), nrow=5, ncol=4)
colnames(A)<-c("Price", "Memory", "Camera", "Looks")
rownames(A)<-paste0("Mobile ", seq(1, 5, 1))
A[,"Price"] <- -A[,"Price"]
apply.VIKOR(A, c(0.35, 0.3, 0.2, 0.15))
```

`apply.WASPAS` *Weighted Aggregated Sum Product Assessment (WASPAS)*

Description

Weighted Aggregated Sum Product Assessment (WASPAS)

Usage

```
apply.WASPAS(mat, weights, beneficial.vector, lambda)
```

Arguments

`mat`	is a matrix and contains the values for different properties of different alternatives
`weights`	are the weights of each property in the decision making process
`beneficial.vector`	is a vector that contains the column number of beneficial properties
`lambda`	a value between 0 and 1, used in the calculation of the W index

Value

the Q index from WASPAS

Examples

```
mat <- matrix(c(0.04, 0.11, 0.05, 0.02, 0.08, 0.05, 0.03, 0.1, 0.03,
1.137, 0.854, 1.07, 0.524, 0.596, 0.722, 0.521, 0.418, 0.62,
960, 1920, 3200, 1280, 2400, 1920, 1600, 1440, 2560), nrow=9)
colnames(mat)<-c("Dimensional Deviation (DD)", "Surface Roughness (SR)",
"Material Removal Rate (MRR)")

rownames(mat)<-paste0("A", 1:9)
beneficial.vector <- c(3)
weights <- c(0.1047, 0.2583, 0.6369)
apply.WASPAS(mat, weights, beneficial.vector, 0.5)
```

apply.WINGS *Apply WINGS (Weighted Influence Non-linear Gauge System)*

Description

This function implements the core calculations of the WINGS method, ignoring any plotting or quadrant labeling. It returns three vectors:

- r_plus_c: (R + C) for each row/column
- r_minus_c: (R - C) for each row/column
- weights: normalized weights derived from (R + C).

Usage

```
apply.WINGS(mat)
```

Arguments

`mat`	A square numeric matrix. The WINGS method is typically applied on an n x n cross-impact or adjacency matrix.

Value

A list with three elements: r_plus_c, r_minus_c, and weights.

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)

colnames(mat)<-c("Toughness Index","Yield Strength","Young's Modulus",
"Density","Thermal Expansion","Thermal Conductivity","Specific Heat")
rownames(mat)<-c("AI 2024-T6", "AI 5052-O","SS 301 FH",
"SS 310-3AH","Ti-6AI-4V","Inconel 718","70Cu-30Zn")

result <- apply.WINGS(mat)
result$r_plus_c    # (R + C)
result$r_minus_c   # (R - C)
result$weights     # Weights
```

`apply.WISP` *Apply WISP (Integrated Simple Weighted Sum Product) method,*

Description

Performs the WISP method calculations, returning a utility score for each alternative. Columns whose indices appear in beneficial.vector are treated as beneficial (max); all other columns are treated as non-beneficial (min).

Usage

```
apply.WISP(mat, beneficial.vector, weights, simplified = FALSE)
```

Arguments

`mat`	A numeric matrix with alternatives in rows and criteria in columns.
`beneficial.vector`	An integer vector of column indices that are beneficial ("max") criteria. All columns not in beneficial.vector are assumed to be "min".
`weights`	A numeric vector of weights, one for each criterion (same length as the number of columns of mat).
`simplified`	A logical. If FALSE, uses all four partial utilities; if TRUE it uses only n_wsd and n_wpr in the final aggregation.

Value

A numeric vector of length nrow(mat) with the final WISP utility scores.

Examples

```
mat <- matrix(c(75.5, 95, 770, 187, 179, 239, 237,
420, 91, 1365, 1120, 875, 1190, 200,
74.2, 70, 189, 210, 112, 217, 112,
2.8, 2.68, 7.9, 7.9, 4.43, 8.51, 8.53,
21.4, 22.1, 16.9, 14.4, 9.4, 11.5, 19.9,
0.37, 0.33, 0.04, 0.03, 0.016, 0.31, 0.29,
0.16, 0.16, 0.08, 0.08, 0.09, 0.07, 0.06), nrow=7)

colnames(mat)<-c("Toughness Index","Yield Strength","Young's Modulus",
"Density","Thermal Expansion","Thermal Conductivity","Specific Heat")
rownames(mat)<-c("AI 2024-T6", "AI 5052-O","SS 301 FH",
"SS 310-3AH","Ti-6AI-4V","Inconel 718","70Cu-30Zn")

# Suppose the first two columns are beneficial, and the 3rd is non-beneficial
beneficial.vector <- c(1,2, 4)
weights <- c(0.28, 0.14, 0.05, 0.24, 0.19, 0.05, 0.05)

# Get the WISP scores
apply.WISP(mat, beneficial.vector, weights, simplified=FALSE)
```

`apply.WSM_WPM` *Apply Weighted Sum Model (WSM) or Weighted Product Model (WPM) on data*

Description

Apply Weighted Sum Model (WSM) or Weighted Product Model (WPM) on data

Usage

```
apply.WSM_WPM(mat, beneficial.vector, weights, method = "WSM")
```

Arguments

`mat`	is a matrix and contains the values for different properties of different alternatives
`beneficial.vector`	is a vector that contains the column number of beneficial properties.
`weights`	are the weights of each property in the decision making process

Value

a vector containing the calculated preference score, run rank(-apply.WSM(mat, beneficial.vector, weights)) to get the ranks

Examples

```
mat <- matrix(c(250, 200, 300, 275, 225, 16, 16, 32, 32, 16, 12, 8, 16, 8, 16, 5, 3, 4, 4, 2), nrow=5, ncol=4)
colnames(mat)<-c("Price", "Storage space", "Camera", "Looks")
rownames(mat)<-paste0("Mobile ", seq(1, 5, 1))
beneficial.vector <- c(2, 3, 4)
weights <- c(0.25, 0.25, 0.25, 0.25)
apply.WSM_WPM(mat, beneficial.vector, weights, "WSM")
```

`find.weight` *Finding the weights for each criteria given a pairwise comparison matrix A in the AHP method*

Description

Finding the weights for each criteria given a pairwise comparison matrix A in the AHP method

Usage

```
find.weight(A)
```

Arguments

`A`	the matrix containing information related to pairwise comparisons of criteria

Value

a list containing the value of CI/RI and a vector containing the weights of each criteria

---

generate.SPOTIS.bounds   *Generate bounds for criteria from a decision matrix*

Description

Generate bounds for criteria from a decision matrix

Usage

```
generate.SPOTIS.bounds(matrix)
```

Arguments

matrix	A numeric matrix or data frame where rows represent alternatives and columns represent criteria.

Value

A numeric matrix with two columns: minimum and maximum bounds for each criterion.

Examples

```
# Decision matrix
matrix <- matrix(c(96, 145, 200,
                   100, 145, 200,
                   120, 170, 80,
                   140, 180, 140,
                   100, 110, 30), nrow = 5, byrow = TRUE)

# Generate bounds
bounds <- generate.SPOTIS.bounds(matrix)
print(bounds)
```

---

plot.AHP.decision.tree   *Plot decision tree*

Description

Plot decision tree

Usage

```
## S3 method for class 'AHP.decision.tree'
plot(
  A,
  comparing.competitors,
  results,
  vertex_font = 1.2,
  edge_font = 1,
  asp = 0.8,
  max_width = 5,
  vertex_size = 50
)
```

Arguments

A	the comparison matrix
comparing.competitors	the list of matrices related to pairwise comparisons of competitors for each criteria
results	results of running AHP on data
vertex_font	font of text on vertex
edge_font	size of the arrows
asp	aspect ratio of the graph
max_width	maximum width
vertex_size	vertex size

Value

the decision tree plot

---

plot.spider — *Plot spider plot*

Description

Plot spider plot

Usage

```
## S3 method for class 'spider'
plot(data, colors = palette("default"))
```

Arguments

data	the result of MCDA scores
colors	the color scheme of choice

Value

the spider plot

read.csv.AHP.matrices *Read csv file containing pairwise comparison matrices for applying AHP or ANP*

Description

Read csv file containing pairwise comparison matrices for applying AHP or ANP

Usage

```
read.csv.AHP.matrices(data)
```

Arguments

data	the matrix containing information related to pairwise comparisons of criteria

Value

a list containing a matrix A related to pairwise comparison of criteria and a list containing multiple matrices related to pairwise comparisons of different competitor products

Examples

```
data <- read.csv(system.file("extdata", "AHP_input_file.csv", package = "RMCDA"), header=FALSE)
mat.lst <- read.csv.AHP.matrices(data)
```

read.csv.SBWM.matrices *Read csv file containing input to the stratified BWM method*

Description

Read csv file containing input to the stratified BWM method

Usage

```
read.csv.SBWM.matrices(data)
```

Arguments

data	input of the csv file

Value

the inputs to the SBWM method

Examples

```
data <- read.csv(system.file("extdata", "stratified_BWM_case_study_I_example.csv", package = "RMCDA"), header
mat.lst <- read.csv.SBWM.matrices(data)
```

`read.csv.SMCDM.matrices`

Read csv file containing pairwise comparison matrices for applying SMCDM

Description

Read csv file containing pairwise comparison matrices for applying SMCDM

Usage

```
read.csv.SMCDM.matrices(data)
```

Arguments

`data`	the matrix containing information related to pairwise comparisons of criteria

Value

a list containing a matrix A related to pairwise comparison of criteria and a list containing multiple matrices related to pairwise comparisons of different competitor products

Examples

```
data <- read.csv(system.file("extdata", "SMCDM_input.csv", package = "RMCDA"), header = FALSE)
mat.lst <- read.csv.SMCDM.matrices(data)
```

Index