Package 'mcdabench'

June 4, 2026

Type Package

Title Benchmarking for Multi-Criteria Decision Analysis

Version 1.1.2

Date 2026-06-05

Author Cagatay Cebeci [aut, cre]

Maintainer Cagatay Cebeci <cebecicagatay@gmail.com>

Description Performs and benchmarks various Multi-Criteria Decision Analysis (MCDA) methods. MCDA is a decision-making framework used to evaluate and rank alternatives based on multiple conflicting criteria using normalization, weighting, and aggregation techniques. The package implements a wide range of MCDA methods including ARAS (Additive Ratio Assessment), AROMAN (Alternative Ranking Order Method Accounting for two-step Normalization), COCOSO (Combined Compromise Solution), CODAS (Combinative Distance-based Assessment), COPRAS (Complex Proportional Assessment), EDAS (Evaluation based on Distance from Average Solution), ELECTRE (Elimination and Choice Expressing Reality) family (I-IV), FUCA (Faire Un Choix Adequat), GRA (Grey Relational Analysis), MABAC (Multi-Attributive Border Approximation Area Comparison), MAIRCA (Multi-Attributive Ideal-Real Comparative Analysis), MARCOS (Measurement of Alternatives and Ranking according to Compromise Solution), MAUT (Multi-Attribute Utility Theory), MAVT (Multi-Attribute Value Theory), MEGAN (Multi-criteria Evaluation with Gradual-weighting and Aggregation of Normalized distance matrices), MOORA (Multi-Objective Optimization on the basis of Ratio Analysis), OCRA (Operational Competitiveness Rating Analysis), ORESTE (Organisation, Rangement Et Synthese De Donnees Relationnelles), PROMETHEE (Preference Ranking Organization Method for Enrichment Evaluations I-VI), RAM (Root Assessment Method), ROV (Range of Value), SMART (Simple Multi-Attribute Rating Technique), TOPSIS (Technique for Order Preference by Similarity to Ideal Solution), VIKOR (VlseKriterijumska Optimizacija I Kompromisno Resenje), WASPAS (Weighted Aggregated Sum Product Assessment), WPM (Weighted Product Model), and WSM (Weighted Sum Model). The package computes comparative evaluation measures including Spearman rank correlation (Spearman, 1904) <doi:10.2307/1412107>, Salabun-Urbaniak's weight similarity index (Salabun and Urbaniak, 2020)<doi:10.1007/978-3-030-50417-5_47>, Wilcoxon signed-rank

test (Wilcoxon, 1945)<doi:10.2307/3001968>, and permutation- and bootstrap-based entropy difference tests for pairwise method comparisons using Jensen-Shannon divergence (Lin, 1991)<doi:10.1109/18.61115>. It also provides sensitivity and stability analysis of MCDA results. Weight sensitivity analysis is implemented through deterministic and stochastic perturbation of criterion weights, and is also integrated as a built-in step within the MEGAN method framework (Cebeci, 2026)<doi:10.7717/peerj-cs.3819>.

Depends R (>= 4.5.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports factoextra, ggplot2, gplots, igraph, monochromeR, networkD3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-06-04 11:50:14 UTC

Contents

mcdabench-package *Benchmarking for Multi-Criteria Decision Analysis Methods*

Description

Comparative benchmarking of Multi-Criteria Decision Analysis (MCDA) methods. These methods are designed to evaluate and rank alternatives based on multiple criteria, applying normalization, weighting, and aggregation techniques. The package includes popular decision-making methods such as MEGAN, TOPSIS, VIKOR, WASPAS, WSM, WPM, and others, facilitating flexible and efficient analyses for multi-criteria problems.

Details

The `mcdabench` package aims to simplify the application of MCDA methods for researchers, analysts, and decision-makers. The package includes functionality for:

- Data normalization (e.g., `"maxmin"`, `"sum"`, `"linear"`)
- Weight calculation using various techniques (`"equal"`, `"entropy"`, `"critic"`, etc.)
- Methods for ranking alternatives based on multiple criteria:
  - `edas` - Evaluation based on Distance from Average Solution
  - `mabac` - Multi-Attributive Border Approximation Area Comparison
  - `marcos` - Measurement of Alternatives and Ranking According to Compromise Solution
  - `ram` - Root Assessment Method
  - `topsis` - Technique for Order of Preference by Similarity to Ideal Solution
  - `vikor` - Compromise ranking based on utility and regret measures
  - `waspas` - Combines Weighted Sum Model (WSM) and Weighted Product Model (WPM)
  - `wsm` - Weighted Sum Method
  - `wpm` - Weighted Product Method
- Flexible support for benefit-cost criteria, customizable weight schemes, and normalization methods.

This package is suited for applications in various decision-making contexts, such as supply chain management, project evaluation, and resource allocation.

For detailed examples and method-specific information, refer to the help pages of individual functions.

Value

No return value. This documentation describes the `mcdabench` package and its functionality.

Author(s)

Cagatay Cebeci

Examples

```
data(egrids)
dm <- as.matrix(egrids$dmat) # Decision matrix
bc <- egrids$bcvec # Benefit and cost vector
wei <- egrids$weights # Weigths of the criteria

# Perform MEGAN analysis
resmegan <- megan(dmatrix=dm, bcvec=bc, weights=wei, thr=0)
print(resmegan$superiority)  # MEGAN superiority scores
print(resmegan$rank) # Ranks of alternatives by MEGAN

# Perform VIKOR analysis
resvikor <- vikor(dmatrix=dm, bcvec=bc, weights=wei, v=0.5)
print(resvikor$Q)  # VIKOR compromise scores
print(resvikor$rank)  # Ranks of alternatives by VIKOR

# Perform TOPSIS analysis
restopsis <- topsis(dmatrix=dm, bcvec=bc, weights=wei)
print(restopsis$p)  # TOPSIS closeness coefficient
print(restopsis$rank) # Ranks of alternatives by TOPSIS
```

---

ahp *AHP: Analytic Hierarchy Process*

Description

Computes criteria weights using the Analytic Hierarchy Process (AHP). Supports multiple weighting methods and consistency ratio calculation.

Usage

```
ahp(pmatrix, method = "maxeig", weights = NULL, tiesmethod = "average")
```

Arguments

`pmatrix`	A square reciprocal comparison matrix.
`method`	Weight calculation method. Options: `"maxeig"`, `"mean"`, `"geometric"`. Default is `"maxeig"`.
`weights`	Optional predefined weights. If provided, AHP calculations will be skipped.
`tiesmethod`	Ranking method for handling ties in weight values. Default is `"average"`.

Details

The AHP method calculates weights based on a comparison matrix. If `weights` are provided, direct ranking is performed without AHP computation. Otherwise, weights are computed using `method`, with `"maxeig"` using the principal eigenvector. The consistency ratio (CR) is also calculated to assess decision matrix reliability.

Value

A list with the following components:

weights	Computed weights for criteria.
rc	Consistency ratio (CR) for matrix reliability.
rank	Ranking of criteria based on weights.

Author(s)

Cagatay Cebeci

References

Saaty, T. L. (1990). How to make a decision: the analytic hierarchy process. *European Journal of Operational Research, 48*(1), 9-26. <doi:10.1016/0377-2217(90)90057-I>

Examples

```
dataset <- matrix(c(
  1, 6, 2, 5, 7,
  1/6, 1, 4, 8, 3,
  1/2, 1/4, 1, 9, 6,
  1/5, 1/8, 1/9, 1, 2,
  1/7, 1/3, 1/6, 1/2, 1
), nrow=5, byrow=TRUE)

resahp_1 <- ahp(dataset, method="maxeig")
print(resahp_1)

resahp_2 <- ahp(dataset, method="mean")
print(resahp_2)
```

---

aras *ARAS: Additive Ratio Assessment Method*

Description

The ARAS method, developed by Zavadskas and Turskis in 2010, aims to evaluate and rank alternatives based on given criteria using normalization, weighting, and utility degree computation. Due to its simple calculation procedures and clear logic, ARAS is widely used in multi-criteria decision-making (MCDA) problems.

Usage

```
aras(dmatrix, bcvec, weights, tiesmethod="average")
```

Arguments

dmatrix	Decision matrix, where rows represent alternatives and columns represent criteria.
bcvec	Benefit-cost vector, specifying for each criterion as a benefit (1) or a cost (-1).
weights	A vector of weights representing the relative importance of each criterion. See calcweights for details.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The ARAS method extends the decision matrix with ESP values, calcnormals the scores, applies weights, calculates utility values (S) and relative utilities (K) for ranking alternatives.

Value

A list containing the following components:

Si	The total utility values of the alternatives.
Ki	The relative utility values for ranking the alternatives.
rank	Ranking the alternatives according to the relative utility values.

Author(s)

Cagatay Cebeci

References

Zavadskas, E. K., & Turskis, Z. (2010). A new additive ratio assessment (ARAS) method in multicriteria decision-making. *Technological and Economic Development of Economy*, *16*(2), 159-172. <doi:10.3846/tede.2010.10>.

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
```

```
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply ARAS with user-defined weights
resaras_1 <- aras(dmatrix=dmat, bcvec=bc, weights=uw)
print(resaras_1)

# Apply ARAS with CRITIC weights
criwei <- calcweights(dmatrix=dmat, bcvec=bc, type="critic", normethod="ratio")

resaras_2 <- aras(dmatrix=dmat, bcvec=bc, weights=criwei)
print(resaras_2$Ki)    # Relative utilities
print(resaras_2$rank)  # Ranks of relative utilities
```

`aroman` *AROMAN: Alternative Ranking Order Method Accounting for Two Step Normalization*

Description

AROMAN applies a two-step normalization process to rank alternatives based on multiple criteria, combining max-min and vector normalization methods.

Usage

```
aroman(dmatrix, bcvec, weights, beta = 0.5, lambda = 0.5, tiesmethod="average")
```

Arguments

`dmatrix`	A numeric matrix (m alternatives x n criteria) representing the initial decision matrix.
`bcvec`	A numeric vector indicating the type of criteria: `1` for benefit criteria (higher is better), `-1` for cost criteria (lower is better).
`weights`	A numeric vector representing the importance of each criterion. If not provided, equal weights are assumed.
`beta`	A numeric value (default: 0.5) controlling the balance between max-min and vector normalization methods.
`lambda`	A numeric value (default: 0.5) adjusting the final ranking calculation by weighting cost and benefit criteria.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The decision matrix `dmatrix` is normalized using both max-min and vector normalization techniques. A weighted normalized matrix is created using the given criteria weights. The aggregated average normalization values are calculated.The sum of weighted normalized values is computed separately for benefit and cost criteria. A final ranking score (R_i) is calculated using `lambda` weighting. Alternatives are ranked based on decreasing R_i values.

Value

A list containing:

L	A numeric vector of weighted sum values for cost criteria.
A	A numeric vector of weighted sum values for benefit criteria.
R	A numeric vector of final ranking values.
rank	An integer vector indicating the ranked order of alternatives.

Author(s)

Cagatay Cebeci

References

Bošković, S., Švadlenka, L., Jovčić, S., Dobrodolac, M., Simić, V., & Bacanin, N. (2023). An alternative ranking order method accounting for two-step normalization (AROMAN) - A case study of the electric vehicle selection problem. *IEEE Access, 11*, 39496-39507. <doi:10.1109/ACCESS.2023.3265818>

Examples

```
# Define the decision matrix
dmat <- matrix(c(
  10, 5, 8,
  6, 9, 7,
  8, 7, 9,
  5, 8, 6
), nrow = 4, byrow = TRUE)

# Profit-cost vector
bc <- c(1, -1, 1)

# Weights vector
uw <- c(0.4, 0.3, 0.3)

# Apply AROMAN
resaroman <- aroman(dmatrix = dmat, bcvec = bc, weights = uw,
    beta = 0.6, lambda = 0.5)
print(resaroman)
```

---

boxplotmcda *MCDA Boxplot*

Description

A function to create boxplots for a Multi-criteria Decision Analysis (MCDA) data matrices with customizable labels, colors, and additional arguments.

Usage

```
boxplotmcda(datamatrix, labels = NULL, colors = NULL, mt = NULL, ...)
```

Arguments

`datamatrix`	A numeric data matrix to generate boxplots.
`labels`	Optional character vector for labeling the x-axis. Default is `NULL`, which assigns sequential labels such as `"A1"`, `"A2"`, etc.
`colors`	Optional vector specifying colors for each box. Default is `NULL`, which uses a rainbow color scheme.
`mt`	Optional main title for the plot.
`...`	Additional arguments to be passed to the underlying `boxplot` function.

Details

This function generates boxplots for the columns of the given data matrix. It allows customization through labels, colors, and additional boxplot arguments. The labels are displayed on the x-axis, and a grid is added for better visual clarity.

Value

No return value. This function is called for its side effect of producing a boxplot visualization of the MCDA data matrix.

Author(s)

Cagatay Cebeci

Examples

```
set.seed(123)
datamatrix <- matrix(runif(50, 0, 1), nrow = 10, ncol = 5)
bcvec <- c(-1, 1, 1, -1)

# Data boxplot with default labels and colors
boxplotmcda(datamatrix)

# Data boxplot with custom labels and colors
labels <- c("Criteria 1", "Criteria 2", "Criteria 3", "Criteria 4", "Criteria 5")
colors <- c("red", "blue", "green", "purple", "orange")
boxplotmcda(datamatrix, labels = labels, colors = colors)

# Normalized data boxplot with with default labels and colors
nmatrix <- calcnormal(X=datamatrix, bcvec=bcvec, type="maxmin")
boxplotmcda(nmatrix, mt="MaxMin Normalized Matrix")
```

`calcnormal` *Normalization of Decision Matrices*

Description

This function provides various normalization techniques to preprocess decision matrices for multi-criteria decision-making methods.

Usage

```
calcnormal(X, bcvec, type = "maxmin")
```

Arguments

`X`	A decision matrix with dimensions `n` x `m` (n rows and m columns).
`bcvec`	A benefit-cost vector, where `-1` indicates cost criteria and `1` indicates benefit criteria.
`type`	The normalization method to apply. The options include: `"maxmin"`, `"vector"`, `"ratio"`, `"sum"`, `"enhanced"`, `"linear"`, `"nonlinear"`, `"logarithmic"`, `"zavadskas"`, `"null"`.

Details

The supported normalization methods are:

`"null"` No normalization.

`"maxmin"` Max-Min normalization. Based on Ersoy (2021); Baydaş et al. (2024).

`"vector"` Vector normalization. Referenced in Nijkamp & van Delft (1977); Milani et al. (2005).

`"sum"` Sum normalization. Jee & Kang (2000); Milani et al. (2005); ÖWang & Luo (2010); Stanujkic et al. (2013); Ersoy (2021).

`"zavadskas"` Zavadskas-Turskis normalization. Zavadskas & Turskis (2011).

`"enhanced"` Enhanced accuracy normalization. Proposed by Zeng et al. (2013).

`"linear"` Linear normalization.

`"nonlinear"` Non-linear normalization.

`"logarithmic"` Logarithmic normalization.

`"ratio"` Linear ratio normalization.

Value

A normalized decision matrix of the same dimension as `X`, the original decision matrix.

Author(s)

Cagatay Cebeci

References

Baydas, M., Yilmaz, M., Jovic, Z., Stevic, Z., Ozuyar, S. E. G., & Ozcil, A. (2024). A comprehensive MCDM assessment for economic data: success analysis of maximum normalization, CODAS, and fuzzy approaches. *Financial Innovation, 10*(1), 105.

Ersoy, N. (2021). Selecting the best normalization technique for ROV method: Towards a real life application. *Gazi University Journal of Science, 34*(2), 592-609.

Jee, D. H., & Kang, K. J. (2000). A method for optimal material selection aided with decision making theory. *Materials & Design, 21*(3), 199-206.

Milani, A. S., Shanian, A., Madoliat, R., & Nemes, J. A. (2005). The effect of normalization norms in multiple attribute decision making models: a case study in gear material selection. *Structural and Multidisciplinary Optimization, 29*, 312-318.

Nijkamp, P., & van Delft, A. (1977). Multi-criteria analysis and regional decision-making (Vol. 8). Springer Science & Business Media.

Stanujkic, D., Djordjevic, B., & Djordjevic, M. (2013). Comparative analysis of some prominent MCDM methods: A case of ranking Serbian banks. *Serbian Journal of Management, 8*(2), 213-241.

Wang, Y. M., & Luo, Y. (2010). Integration of correlations with standard deviations for determining attribute weights in multiple attribute decision making. *Mathematical and Computer Modelling, 51*(1-2), 1-12.

Zavadskas, E. K., & Turskis, Z. (2011). Multiple criteria decision making (MCDM) methods in economics: an overview. *Technological and Economic Development of Economy, 17*(2), 397-427.

Zeng, Q. L., Li, D. D., & Yang, Y. B. (2013). VIKOR method with enhanced accuracy for multiple criteria decision making in healthcare management. *Journal of Medical Systems, 37*(2), 9908.

See Also

calcweights

Examples

```
data(egrids)
dmatrix <- egrids$dmat
bcvec <- egrids$bcvec

# Max-min normalization
normmaxmin <- calcnormal(X=dmatrix, bcvec=bcvec, type="maxmin")
print(normmaxmin)

# Vector normalization
normvector <- calcnormal(X=dmatrix, bcvec=bcvec, type="vector")
print(normvector)

# Zavadskas normalization
normzavad <- calcnormal(X=dmatrix, bcvec=bcvec, type="zavadskas")
print(normzavad)
```

`calcranks` *Calculate Consensus Ranks using Various Methods*

Description

This function calculates consensus ranks for a set of alternatives, given their rankings from multiple MCDM methods. It supports the Score method, Neustadt's method, and Buchholz's method for aggregating ranks.

Usage

```
calcranks(rankmatrix, rankmethod = "score")
```

Arguments

`rankmatrix`	A numeric matrix representing the rankings of alternatives by multiple decision-making methods. Rows correspond to different MCDM methods, and columns correspond to alternatives. Each cell 'rankmatrix[i, j]' should contain the rank of alternative 'j' given by method 'i' (smaller rank is better).
`rankmethod`	A character string specifying the consensus ranking method to use. Supported methods are: • `"score"`: Calculates the overall score for each alternative based on pairwise comparisons (a higher score indicates a better alternative). • `"neustadt"`: Computes Neustadt's rating vector, which is the difference between wins and losses in pairwise comparisons (a higher score indicates a better alternative). • `"buchholz"`: Computes Buchholz's rating vector, based on the sum of squared differences of pairwise comparison scores (a lower score indicates a more consistent/less controversial alternative, thus a better rank).

Value

A list containing the results of the ranking calculation:

- `rankmethod`: A character string indicating the ranking method used.
- `rating_scores`: A named numeric vector of the calculated rating scores for each alternative based on the chosen method.
- `ordered_alternatives`: A character vector of alternative names, ordered from the best-performing to the worst-performing according to the calculated `rating_scores`.
- `score_ranks`: A numeric vector of the final consensus ranks using the "minimum" ties method.
- `average_score_ranks`: A numeric vector of the final consensus ranks for each alternative the "average" tie method.

Author(s)

Cagatay Cebeci

References

Gogodze, J. (2019). Ranking-Theory Methods for Solving Multicriteria Decision-Making Problems. *Advances in Operations Research, 2019*(1), 3217949. <doi:10.1155/2019/3217949>

Examples

```
# Sample ranking matrix (rows are methods, columns are alternatives R1-R10)
rmat <- matrix(c(
  5, 10, 3, 9, 7, 8, 1, 2, 6, 4,
  10, 9, 6, 3, 7, 1, 5, 2, 4, 8,
  10, 9, 8, 4, 6, 1, 7, 5, 2, 3,
  8, 10, 4, 5, 9, 7, 1, 3, 2, 6,
  9, 7, 1, 8, 3, 4, 2, 6, 10, 5,
  8, 10, 5, 9, 6, 7, 1, 2, 3, 4,
  3, 1, 4, 6, 2, 7, 9, 10, 8, 5,
  2, 1, 5, 3, 4, 6, 8, 10, 9, 7,
  4, 1, 8, 2, 5, 3, 10, 9, 6, 7,
  2, 1, 2, 6, 4, 5, 3, 7, 9, 8,
  2, 1, 7, 4, 3, 5, 10, 9, 8, 6,
  7, 10, 3, 9, 5, 8, 1, 2, 6, 4,
  1, 2, 3, 7, 5, 10, 4, 6, 9, 8,
  4, 1, 6, 3, 2, 5, 10, 9, 8, 7
), nrow = 14, byrow = TRUE)

rownames(rmat) <- c("TOPSIS", "MAUT", "MACBETH", "ORESTE", "VIKOR", "SIR", "EVAMIX",
   "ARAS", "MOORA", "COPRAS", "WASPAS", "TODIM", "MABAC", "MARE")
colnames(rmat) <- paste0("R", 1:10)

# Calculate ranks using the Score method
results_score <- calcranks(rmat, rankmethod = "score")
print(results_score)

# Calculate ranks using Neustadt's method
results_neustadt <- calcranks(rmat, rankmethod = "neustadt")
print(results_neustadt)

# Calculate ranks using Buchholz's method
results_buchholz <- calcranks(rmat, rankmethod = "buchholz")
print(results_buchholz)
```

---

calcweights *Calculation of Criteria Weights*

Description

This function calculates criteria weights using different methods such as Equal weighting, and weighting based on Angle, CRITIC, CILOS, Entropy, Geometric mean, GINI coefficient, MEREC, MPSI, ROC, RS, Standard deviation, and Variance.

Usage

```
calcweights(dmatrix, bcvec = NULL, type = "equal", normethod = NULL)
```

Arguments

dmatrix	A normalized decision matrix.
bcvec	Benefit-Cost vector, used with certain methods such as MEREC.
type	The type of weighting method to be applied. The options include: "angle", "critic", "cilos", "entropy", "equal", "geom", "gini", "merec", "roc", "rs", "sdev", and "var". Default weighting method is "equal".
normethod	Optinal argument for normalization method to apply. The options include: "max", "maxmin", "vector", "sum", "enhanced", "linear", "nonlinear", "logarithmic", "zavadskas". Default is NULL.

Details

In multi-criteria decision-making methods, normalization is often necessary when calculating weights, but it depends on the specific approach used. Normalization ensures that criteria with different scales become comparable, which is crucial for accurate decision analysis. For instance, the entropy method applies normalization to make data scale-independent. Similarly, the CRITIC method typically requires normalization since it considers the variability and correlation between criteria. Additionally, various normalization techniques, such as Min-Max normalization, Z-score normalization, and vector normalization, can be employed depending on the method's requirements. Min-Max normalization scales data within a fixed range, making comparisons straightforward. Z-score normalization adjusts data based on mean and standard deviation, which helps in standardizing different distributions. Vector normalization converts values into unit vectors, ensuring proportional representation across criteria. Selecting the most appropriate normalization technique is key to ensuring accurate and meaningful results in MCDA. See link{normalize} for available methods of normalization.

The current version of function includes the following weighting methods:

angle Angular weighting method

critic CRITIC: CRiteria Importance Through Intercriteria Correlation method

cilos CLIOS: Criterion Impact LOSs method

entropy Entropy: Shannon entropy method

entropy2 Entropy method II

equal Equal weighting

geom Geometric mean based weighting

gini GINI coefficient based weighting

merec MEREC: MEthod based on the Removal Effects of Criteria)

mpsi MPSI: Modified Preference Selection Index

roc ROC: Rank Order Centroid weighting

rs RS: Rank Sum weighting

sdev Standard deviation-based weighting

var Variance-based weighting

Value

A vector of weights corresponding to each column of the decision matrix.

Author(s)

Cagatay Cebeci

References

Diakoulaki, D., Mavrotas, G., & Koumoutsos, K. (1995). Determining objective weights in multiple criteria problems using the CRITIC method. *Computers & Operations Research, 22*(7), 763-770. <doi:10.1016/0305-0548(94)00059-H>

Ghorabaee, M. K., Zavadskas, E. K., Olfat, L., & Turskis, Z. (2017). Multi-criteria inventory classification using a novel method of evaluation based on ratios and comparisons (MEREC). *Informatica, 28*(3), 599-616.

Gligoric, M., Gligoric, Z., Lutovac, S., Negovanovic, M., & Langovic, Z. (2022). Novel hybrid MPSI-MARA decision-making model for support system selection in an underground mine. *Systems, 10*(6), 248. <doi:10.3390/systems10060248>

Keshavarz-Ghorabaee, M., Amiri, M., Zavadskas, E. K., Turskis, Z., & Antucheviciene, J. (2021). Determination of objective weights using a new method based on the removal effects of criteria (MEREC). *Symmetry, 13*(4), 525. <doi:10.3390/sym13040525>

Shuai, D., Zongzhun, Z., Yongji, W., & Lei, L. (2012, May). A new angular method to determine the objective weights. In *2012 24th Chinese Control and Decision Conference (CCDC)* (pp. 3889-3892). IEEE.

Zavadskas, E. K., & Podvezko, V. (2016). Integrated determination of objective criteria weights in MCDM. *International Journal of Information Technology & Decision Making, 15*(2), 267-283. <doi:10.1142/S0219622016500036>

Zeleny, M. (1982). *Multiple criteria decision making*. McGraw-Hill.

Jia, J., Fischer, G. W., & Dyer, J. S. (1998). Attribute weighting methods and decision quality in the presence of response error: a simulation study. *Journal of Behavioral Decision Making, 11*(2), 85-105. <doi:10.1002/(SICI)1099-0771(199806)11:2<85::AID-BDM282>3.0.CO;2-K>

Examples

```
# Load the dataset
data(egrids)
dmat <- egrids$dmat  # Decision matrix
bc <- egrids$bcvec   # Benefit-cost vector

# Calculate weights using the equal weighting method
calcweights(dmatrix=dmat, type="equal")

# Calculate weights using the entropy method
calcweights(dmatrix=dmat, type="entropy")

# Calculate weights using the entropy method with MaxMin normalization
calcweights(dmatrix=dmat, bcvec=bc, type="entropy", normethod="maxmin")
```

```
# Calculate weights using the std. deviation method
calcweights(dmatrix=dmat, type="sdev")

# Calculate weights using the MEREC method
calcweights(dmatrix=dmat, bcvec=bc, type="merec")

# Calculate weights using the MPSI method
calcweights(dmatrix=dmat, bcvec=bc, type="mpsi")

# Calculate weights using the angular method
calcweights(dmatrix=dmat, bcvec=bc, type="angle")
```

`cocoso` *COCOSO: Combined Compromise Solution Method*

Description

COCOSO proposed by Yazdani et al (2019) is an MCDA method that normalizes the decision matrix and ranks alternatives based on specified criterion weights.

Usage

```
cocoso(dmatrix, bcvec, weights, lambda = 0.5, tiesmethod="average")
```

Arguments

`dmatrix`	A `matrix` containing the decision matrix. Alternatives are in rows, criteria are in columns.
`bcvec`	A vector indicating the preferred direction for each criterion (`c(1, -1, ...)`). Positive values (`1`) mean higher values are preferred, while negative values (`-1`) mean lower values are preferred.
`weights`	A vector of weights representing the relative importance of each criterion. See `calcweights` for details.
`lambda`	A weighting parameter (`0.5` by default).
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Value

Returns a list containing:

`S_i`	Total weighted normalized values of alternatives.
`P_i`	Multiplicative values of alternatives.
`k_ia`	First combined calculated value.
`k_ib`	Second combined calculated value.
`k_ic`	Third combined calculated value.
`k_i`	Final composite value.
`rank`	Ranking of the alternatives.

Author(s)

Cagatay Cebeci

References

Yazdani, M., Zarate, P., Kazimieras Zavadskas, E., & Turskis, Z. (2019). A combined compromise solution (CoCoSo) method for multi-criteria decision-making problems. *Management Decision, 57*(9), 2501-2519. <doi:10.1108/MD-05-2017-0458>

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply COCOSO with user-defined weights
rescocoso_1 <- cocoso(dmatrix=dmat, bcvec = bc, weights = uw, lambda=0.1)
print(rescocoso_1)

# Apply COCOSO with CRITIC weights
cw <- calcweights(dmatrix=dmat, bcvec=bc, type="critic", normethod="linear")
rescocoso_2 <- cocoso(dmatrix=dmat, bcvec = bc, weights = cw, lambda=0.8, tiesmethod="average")
print(rescocoso_2)
```

---

codas *CODAS: COmbinative Distance-based ASsessment*

Description

Implements the COmbinative Distance-based ASsessment (CODAS) method for multi-criteria decision analysis (Keshavarz et al, 2016). This method ranks alternatives based on a combination of Euclidean and Taxicab distances from the negative ideal solution.

Usage

```
codas(dmatrix, bcvec, weights, thr = 0.1, tiesmethod="average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix. Rows are alternatives and columns are criteria.
bcvec	A numeric vector indicating the benefit/cost nature of each criterion. Use 1 for benefit (to be maximized) and -1 for cost (to be minimized). The length of bcvec must equal the number of columns in dmatrix.
weights	A vector of weights representing the relative importance of each criterion. See calcweights for details.
thr	A numeric value between 0 and 1 representing the threshold parameter. It influences the balance between Euclidean and Taxicab distances in the appraisal score calculation. Default is 0.1.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The CODAS method evaluates alternatives by calculating their distances from the negative ideal solution using both Euclidean and Taxicab metrics. A combinative appraisal score is then computed, giving more preference to the Euclidean distance. The alternatives are ranked based on these appraisal scores in descending order.

Value

A list containing the following components:

niv	The negative ideal solution vector.
euc_dist	A numeric vector of Euclidean distances of each alternative from the negative ideal solution.
tax_dist	A numeric vector of Taxicab distances of each alternative from the negative ideal solution.
pi_i	A numeric vector of appraisal scores for each alternative.
rank	A numeric vector representing the rank of each alternative. Lower rank is better.

Author(s)

Cagatay Cebeci

References

Keshavarz Ghorabaee, M., Zavadskas, E. K., Turskis, Z., & Antuchevičienė, J. (2016). A new combinative distance-based assessment (CODAS) method for multi-criteria decision-making. <https://etalpykla.vilniustech.lt/han

See Also

calcweights, calcnormal

Examples

```
# Decision matrix (5 alternatives, 4 criteria)
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply CODAS with user-defined weights, maxmin normalization, and thr = 0.1
rescodas_1 <- codas(dmatrix=dmat, bcvec = bc, weights = uw)
print(rescodas_1)

# Apply CODAS with entropy weights
entwei <- calcweights(dmatrix=dmat, bcvec = bc, type="entropy", normethod="sum")
rescodas_2 <- codas(dmatrix=dmat, bcvec = bc, weights = entwei, thr = 0.5)
print(rescodas_2)
```

---

copras *COPRAS: Complex Proportional Assessment*

Description

Applies the COPRAS method for decision makings.

Usage

```
copras(dmatrix, bcvec, weights, normethod = "sum", tiesmethod = "average")
```

Arguments

dmatrix	A decision matrix where rows represent alternatives and columns represent criteria.
bcvec	A vector indicating the criteria type: 1 for benefit criteria, -1 for cost criteria.
weights	A numeric vector representing the relative importance of each criterion, summing to 1.

`normethod`	Normalization method applied to the decision matrix. The `"sum"` normalization is default to COPRAS.
`tiesmethod`	Method to resolve ties in ranking, default is `"average"`.

Value

A list with the following components:

`weights`	The weights applied to each criterion.
`splus`	The positive significance sum for each alternative.
`sminus`	The negative significance sum for each alternative.
`qi`	The relative significance (Q) values.
`ui`	Normalized utility values of each alternative.
`rank`	Ranking of alternatives based on utility values.

Author(s)

Cagatay Cebeci

References

Razavi Hajiagha, S. H., Hashemi, S. S., & Zavadskas, E. K. (2013). A complex proportional assessment method for group decision making in an interval-valued intuitionistic fuzzy environment. *Technological and Economic Development of Economy, 19*(1), 22-37. <doi:10.3846/20294913.2012.762953>

Examples

```
dmatrix <- matrix(c(
  10, 50, 30, 80,
  20, 60, 20, 70,
  30, 40, 40, 60,
  40, 30, 50, 50
), nrow=4, byrow=TRUE)

bcvec <- c(1, 1, -1, 1)
weights <- c(0.3, 0.4, 0.1, 0.2)

rescopras <- copras(dmatrix, bcvec, weights)
print(rescopras)
```

corplot *Correlation Matrix Visualization*

Description

Visualizes a correlation or similarity matrix using heatmap-style plots, allowing customization of colors, shapes, axis labels, and text annotations.

Usage

```
corplot(cormat, labsize = 10, fsize = 3, fcolor = "black",
   colpal = c("red", "white", "dodgerblue"), title = "Correlation Matrix",
   xlab = "Coef", ylab = "Coef", shape = "square", displabels = TRUE)
```

Arguments

cormat	A matrix representing correlations or similarities.
labsize	Numeric, base size of plot text and labels. Default is 10.
fsize	Numeric, font size for cell text annotations. Default is 3.
fcolor	Character, color of text annotations. Default is "black".
colpal	Character vector, specifying three colors for low, mid, and high values. Default is c("red", "white", "blue").
title	Character, title of the plot. Default is "Correlation Matrix".
xlab	Character, label for the X-axis. Default is "Coef".
ylab	Character, label for the Y-axis. Default is "Coef".
shape	Character, specifying either "circle" or "square" for visualization. Default is "square".
displabels	Logical, whether to display correlation values inside cells. Default is TRUE.

Value

No return value.

Examples

```
# Sample correlation matrix
set.seed(123)
df <- data.frame(
  X1 = rnorm(20), X2 = rnorm(20), X3 = rnorm(20), X4 = rnorm(20),
  X5 = rnorm(20), X6 = rnorm(20), X7 = rnorm(20), X8 = rnorm(20),
  X9 = rnorm(20), X10 = rnorm(20)
)
cormat <- cor(df, method = "spearman")

corplot(cormat, labsize = 12, fsize = 4, fcolor = "black",
        title = "Spearman Correlation Matrix", xlab = "Methods", ylab = "Methods",
        shape = "square", displabels = TRUE)
```

edas *EDAS: Evaluation based on Distance from Average Solution*

Description

This function implements the method Evaluation based on Distance from Average Solution (EDAS) (Keshavarz et al, 2015) for multi-criteria decision-making by calculating distances from the average solution for each criterion, and evaluating alternatives based on positive and negative distances.

Usage

```
edas(dmatrix, bcvec, weights, tiesmethod="average")
```

Arguments

`dmatrix`	Decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
`weights`	A numeric vector of criteria weights. See `calcweights` for details.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The EDAS method evaluates alternatives based on their positive and negative distances from the average solution for each criterion. Positive distances (`pda`) measure how much better an alternative is compared to the average, while negative distances (`nda`) measure how much worse it is. These values are combined into scores for ranking alternatives.

Value

A list containing the following components:

`amatrix`	The average solution for each criterion.
`pda`	The positive distance from the average for each alternative and criterion.
`nda`	The negative distance from the average for each alternative and criterion.
`sp`	The aggregated positive distances for each alternative.
`sn`	The aggregated negative distances for each alternative.
`nsp`	The normalized positive distances for each alternative.
`nsn`	The normalized negative distances for each alternative.
`score`	The final scores of alternatives, based on positive and negative distances.
`rank`	The ranks of final score of alternatives.

Author(s)

Cagatay Cebeci

References

Keshavarz Ghorabaee, M., Zavadskas, E. K., Olfat, L., & Turskis, Z. (2015). Multi-criteria inventory classification using a new method of evaluation based on distance from average solution (EDAS). *Informatica, 26(3)*, 435-451.<doi: 10.3233/INF-2015-1070>.

See Also

calcweights

Examples

```
# Decision matrix (5 alternatives, 4 criteria)
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
userwei <- c(0.1, 0.4, 0.3, 0.2)

# Apply EDAS with user-defined weights
resedas_1 <- edas(dmatrix=dmat, bcvec = bc, weights = userwei)
print(resedas_1)

# Apply EDAS with GINI weights
giniwei <- calcweights(dmatrix=dmat, bcvec = bc, type="gini")
resedas_2 <- edas(dmatrix=dmat, bcvec = bc, weights = giniwei)
print(resedas_2)
```

---

egrids *System Configuration for Optimizing Smart Grids*

Description

The egrids dataset contains a decision matrix (egrids), a benefit-cost vector (bcvec), and a weights vector (weights) designed for analyzing and ranking grid alternatives (Gs) using multi-criteria decision-making methods.

Usage

```
data(egrids)
```

Format

A list with the following components:

`dmat` A numeric matrix where rows represent grid alternatives (G) and columns represent criteria. Criteria include:

- `C1`: Energy Efficiency (
- `C2`: System Reliability (
- `C3`: Renewable Integration (
- `C4`: Grid Stability Score. Higher is better
- `C5`: Environmental Impact (Score). Lower is better
- `C6`: Operational Cost ($M). Lower is better
- `C7`: Emissions (CO_2 tons/year). Lower is better
- `C8`: Response Time (seconds). Lower is better
- `C9`: Infrastructure Investment ($M). Lower is better
- `C10`: Maintenance Cost ($M). Lower is better

`bcvec` A numeric benefit and cost vector specifying the type of each criterion:

- `1`: Benefit criterion meaning higher value is better.
- `-1`: Cost criterion meaning lower value is better.

`weights` A numeric vector containing the weights assigned to each criterion.

Details

This synthetic dataset provides basic criteria for evaluating and comparing smart grids using various multi-criteria decision analysis methods such as TOPSIS, VIKOR, WASPAS, MEGAN, and others. Each row in the decision matrix (`dmat`) corresponds to a different grid, while each column represents a specific criterion.

The `bcvec` vector identifies whether each criterion is a benefit (`1`) or a cost (`-1`), enabling proper handling in decision-making methods.

The `weights` vector contains the weights given to criteria.

Author(s)

Cagatay Cebeci

Examples

```
data(egrids)
dmat <- egrids$dmat # Decision matrix
bcvec <- egrids$bcvec # Benefit and cost vector
weights <- egrids$weights # Criteria weights

print(dmat)
print(bcvec)
print(weights)
```

`electre1` *ELECTRE I (ELimination Et Choix Traduisant la REalité)*

Description

Performs outranking analysis using the ELECTRE I method, identifying dominant alternatives based on concordance and discordance indices.

Usage

```
electre1(dmatrix, bcvec, weights, ct = 0.65, dt = 0.35, tiesmethod = "average")
```

Arguments

`dmatrix`	A numeric decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	A numeric vector indicating for each criterion whether higher values are preferable (`1`) or not (`-1`).
`weights`	A numeric vector of weights assigned to each criterion. Length must match the number of columns in `dmatrix`.
`ct`	Concordance threshold. Must be between 0 and 1. Defaults to `0.65`.
`dt`	Discordance threshold. Must be between 0 and 1. Defaults to `0.35`.
`tiesmethod`	Tie-breaking method for final ranking. Passed to `rank`. Defaults to `"average"`.

Details

ELECTRE I is the original outranking-based multi-criteria decision method. It constructs binary pairwise relations based on:

- Concordance index: weighted agreement over criteria supporting the preference of one alternative over another.
- Discordance index: the strongest opposition to that preference.

An alternative \(a_i\) outranks \(a_j\) only if:

- The concordance index is at least `ct`, and
- The discordance index is at most `dt`.

The final result includes the outranking relations and the kernel (alternatives that are not strongly dominated by others). Additionally, a basic ranking is computed based on how many alternatives each option outranks, using `tiesmethod`.

Value

A list with the following elements:

`Concordance`	The n x n matrix of concordance indices.
`Discordance`	The n x n matrix of discordance indices.
`Outrank`	Logical matrix indicating outranking relations between alternatives.
`Kernel`	Indices of non-dominated alternatives (the kernel).
`Outrank_Score`	Number of alternatives outranked by each alternative.
`rank`	Ranking based on outrank strength.

Author(s)

Cagatay Cebeci

See Also

`electre2`, `electre3`, `electre4`, `calcnormal`

Examples

```
dmatrix <- matrix(c(70, 3, 500,
                    90, 4, 400,
                    60, 2, 550),
                  nrow = 3, byrow = TRUE)
rownames(dmatrix) <- c("A1", "A2", "A3")
bcvec <- c(1, 1, -1)
weights <- c(0.4, 0.3, 0.3)
result <- electre1(dmatrix, bcvec, weights)
result$Kernel
result$rank
```

---

electre2 *ELECTRE II (ELimination Et Choix Traduisant la REalité)*

Description

Implements the ELECTRE II outranking method to rank a set of alternatives based on concordance, discordance, and credibility analysis.

Usage

```
electre2(dmatrix, bcvec, weights, p = NULL, q = NULL, v = NULL,
         st = 0.65, wt = 0.5, tiesmethod = "average")
```

Arguments

dmatrix	A numeric matrix or data frame representing the decision matrix, where rows are alternatives and columns are criteria.
bcvec	A numeric vector indicating the nature of each criterion: 1 for benefit and -1 for cost.
weights	A numeric vector containing weights for each criterion. Must match the number of columns in dmatrix.
p	A list of preference thresholds for each criterion. If NULL, default values are estimated from the data.
q	A list of indifference thresholds for each criterion. If NULL, defaults are estimated.
v	A list of veto thresholds for each criterion. If NULL, defaults are estimated.
st	Strong credibility threshold (default = 0.65). Values above this define a strong outranking relation.
wt	Weak credibility threshold (default = 0.5). Values between wt and st define a weak outranking relation.
tiesmethod	String specifying the tie-breaking method to be used in rank.

Details

ELECTRE II uses pairwise credibility analysis with two credibility thresholds to distinguish strong and weak outranking relations among alternatives.

The st and wt arguments define the strong and weak credibility cutoffs. While there is no fixed default specified in the original ELECTRE II methodology, values such as st = 0.65 and wt = 0.5 are commonly used in the literature due to their balance between discrimination and robustness.

However, users are encouraged to tune these thresholds based on the decision context, the number of alternatives, and how sensitive the comparison process should be. In highly uncertain decision environments, lower thresholds may be appropriate.

The final ranking is computed as the average rank between upward and downward outranking scores, providing a balanced assessment of dominance and vulnerability.

Value

A list with the following elements:

Concordance	Matrix of concordance indices between alternatives.
Discordance	Matrix of discordance indices between alternatives.
Credibility	Overall credibility matrix based on concordance and discordance.
Strong	Logical matrix indicating strong outranking relationships.
Weak	Logical matrix indicating weak outranking relationships.
Rank_Upward	Ranking of alternatives from strongest outranking profile.
Rank_Downward	Ranking of alternatives from most dominated profile.
rank	Final overall ranking based on average of upward and downward ranks.

Author(s)

Cagatay Cebeci

See Also

electre3, electre4, calcnormal, rank

Examples

```
dmatrix <- matrix(c(70, 3, 500,
                    90, 4, 400,
                    60, 2, 550),
                  nrow = 3, byrow = TRUE)
rownames(dmatrix) <- c("A1", "A2", "A3")
bcvec <- c(1, 1, -1)
weights <- c(0.4, 0.3, 0.3)
result <- electre2(dmatrix, bcvec, weights)
result$rank
```

---

electre3 *ELECTRE III (ELimination Et Choix Traduisant la REalité)*

Description

The ELECTRE III method applies fuzzy outranking by evaluating concordance and discordance between alternatives using user-defined or default pseudo-criteria. It generates a credibility matrix and provides net scores and a partial ranking.

Usage

```
electre3(dmatrix, bcvec, weights, p = NULL, q = NULL, v = NULL,
    thr = 0.5, tiesmethod = "average")
```

Arguments

dmatrix	A numeric matrix or data frame representing the decision matrix (alternatives as rows, criteria as columns).
bcvec	A numeric vector indicating whether each criterion is a benefit (1) or cost (-1).
weights	A numeric vector of criteria weights, summing to 1 or scaled accordingly.
p	A list of preference thresholds for each criterion. If NULL, default values are estimated.
q	A list of indifference thresholds for each criterion. If NULL, default values are estimated.
v	A list of veto thresholds for each criterion. If NULL, default values are estimated.
thr	The credibility threshold for dominance consideration (default = 0.5).
tiesmethod	A string indicating the method used to resolve ties in ranking. See rank for options.

Details

ELECTRE III uses the pseudo-criteria and fuzzy thresholds.

Value

A list with the following components:

`Credibility`	The n x n credibility matrix among alternatives.
`Net_Flow`	A numeric vector of average credibility scores per alternative.
`rank`	A named vector of the ranking of alternatives.

Author(s)

Cagatay Cebeci

See Also

`electre1`, `electre2`, `electre4`, `calcnormal`

Examples

```
dmatrix <- matrix(c(70, 3, 500,
                    90, 4, 400,
                    60, 2, 550),
                  nrow = 3, byrow = TRUE)
rownames(dmatrix) <- c("A1", "A2", "A3")
bcvec <- c(1, 1, -1)
weights <- c(0.4, 0.3, 0.3)
result <- electre3(dmatrix, bcvec, weights)
result$rank
```

---

electre4 *ELECTRE IV (ELimination Et Choix Traduisant la REalité)*

Description

The ELECTRE IV is a multi-criteria decision-making technique that ranks alternatives using concordance and discordance principles with progressive elimination.

Usage

```
electre4 (dmatrix, bcvec, weights, p = NULL, q = NULL, v = NULL,
    thr = 0.5, tiesmethod = "average")
```

Arguments

dmatrix	A decision matrix where rows represent alternatives and columns represent criteria.
bcvec	A vector specifying the benefit (1) or cost (-1) nature of each criterion.
weights	A vector of weights representing the relative importance of each criterion. See calcweights for details.
p	Pseudo-criterion preference threshold values. Default is calculated dynamically.
q	Pseudo-criterion indifference threshold values. Default is set to half of pt.
v	Pseudo-criterion veto threshold values. Default is set to pt multiplied by 1.5.
thr	Threshold for credibility-based elimination. A higher threshold (e.g., 0.7 or 0.8) will make selection more restrictive, favoring only alternatives with strong credibility. A lower threshold (e.g., 0.4 or 0.5) will allow more alternatives to remain in consideration. Default threshold is 0.5.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The ELECTRE IV method constructs concordance and discordance matrices based on pseudo-criteria thresholds. A credibility matrix is then derived, followed by a threshold-based filtering process to eliminate weaker alternatives.

Value

A list containing:

Concordance	Concordance matrix representing the degree of dominance of alternatives.
Discordance	Discordance matrix indicating veto situations.
Credibility	Credibility matrix integrating concordance and discordance effects.
Net_Flow	Computed net credibility flow values for ranking.
rank	Final ranking of alternatives after threshold-based elimination.

Author(s)

Cagatay Cebeci

See Also

calcweights, calcnormal

Examples

```
dmat <- matrix(c(5, 150, 3, 200, 4, 180), nrow = 3, byrow = TRUE)
bc <- c(-1, 1)
uw <- c(0.6, 0.4)

# Pseudo-criterion threshold values
pt <- list(list(p = 2), list(p = 30))
qt <- list(list(q = 1), list(q = 15))
vt <- list(list(v = 3), list(v = 45))
thr <- 0.6

reselectre4_1 <- electre4(dmatrix = dmat, bcvec = bc, weights = uw,
   p = pt, q = qt, v = vt, thr=thr)
print(reselectre4_1)

criwei <- calcweights(dmat, bcvec = bc, type = "critic")
reselectre4_2 <- electre4(dmatrix = dmat, bcvec = bc, weights = criwei)
print(reselectre4_2)
```

`flowplot` *Visualize Dominance Flow of Ranked Alternatives*

Description

Creates a directional plot illustrating the dominance flow among alternatives based on their ranks. Equal ranks are marked with "=" and dominance with ">".

Usage

```
flowplot(rankvec, colpal, txtcol = "black", orientation = "horizontal")
```

Arguments

`rankvec`	A numeric vector of rankings for each alternative. Names are used as labels; if missing, default labels `A1`, `A2`, ... are generated.
`colpal`	A vector of colors corresponding to unique rank values. Should have at least as many colors as unique entries in `rankvec`.
`txtcol`	Color for vertex labels. Defaults to black.
`orientation`	Layout direction of the flow. Use `"horizontal"` (default) or `"vertical"`.

Details

The function draws a graph in which each node represents an alternative. Arrows indicate flow of preference: a ">" means dominance, while "=" indicates tie. The layout is determined by the specified orientation. Edge labels are automatically positioned slightly above the edge direction based on flow geometry.

Value

No return value.

Author(s)

Cagatay Cebeci

Examples

```
rankvec <- c(1, 2, 3, 4, 5.5, 5.5, 7)
colpal <- rainbow(length(unique(rankvec)))

# Horizontal flow
flowplot(rankvec, colpal, orientation = "horizontal")

# Vertical flow
flowplot(rankvec, colpal, orientation = "vertical")
```

---

ftopsis *Fuzzy TOPSIS Method*

Description

Computes rankings for decision alternatives using the Fuzzy TOPSIS method, which evaluates alternatives based on triangular fuzzy numbers (l, m, u).

Usage

```
ftopsis(fuzzydmatrix, bcvec, fuzzyweights, tiesmethod = "average")
```

Arguments

`fuzzydmatrix`	A decision matrix where each criterion is represented by triangular fuzzy numbers (l, m, u), structured as three adjacent columns per criterion.
`bcvec`	A vector indicating the type of each criterion: 1 for benefit criteria and -1 for cost criteria. Must have the same length as the number of criteria.
`fuzzyweights`	A matrix where each row corresponds to a criterion and contains three columns representing fuzzy weights (l, m, u).
`tiesmethod`	Method for resolving ties in ranking, defaulting to ‘"average"‘.

Value

A list containing:

`Criteria`	Number of criteria evaluated.
`fuzzyweights`	Fuzzy weights applied to each criterion.
`FPIS`	Fuzzy Positive Ideal Solution.

`FNIS` Fuzzy Negative Ideal Solution.

`distance_to_FPIS`
: Euclidean distance of each alternative to FPIS.

`distance_to_FNIS`
: Euclidean distance of each alternative to FNIS.

`closeness_coefficient_fuzzy`
: Closeness coefficient values for ranking.

`rank` Final ranking of alternatives based on closeness coefficients.

Author(s)

Cagatay Cebeci

References

Nadaban, S., Dzitac, S., & Dzitac, I. (2016). Fuzzy TOPSIS: a general view. *Procedia Computer Science, 91*, 823-831. <doi:10.1016/j.procs.2016.07.088>

Examples

```
fuzzydmatrix <- matrix(c(
  8, 10, 12,    # A1-C1
  18, 20, 22,   # A2-C1
  28, 30, 32,   # A3-C1
  38, 40, 42,   # A4-C1
  45, 50, 55,   # A1-C2
  55, 60, 65,   # A2-C2
  35, 40, 45,   # A3-C2
  25, 30, 35,   # A4-C2
  28, 30, 32,   # A1-C3
  18, 20, 22,   # A2-C3
  38, 40, 42,   # A3-C3
  48, 50, 52,   # A4-C3
  75, 80, 85,   # A1-C4
  65, 70, 75,   # A2-C4
  55, 60, 65,   # A3-C4
  45, 50, 55    # A4-C4
), nrow = 4, byrow = TRUE)

bcvec <- c(1, 1, -1, 1)

fuzzyweights <- matrix(c(
  0.25, 0.30, 0.35,
  0.35, 0.40, 0.45,
  0.05, 0.10, 0.15,
  0.15, 0.20, 0.25
), nrow = 4, byrow = TRUE)

resftopsis <- ftopsis(fuzzydmatrix, bcvec, fuzzyweights)
print(resftopsis)
```

fuca *FUCA: Faire Un Choix Adéquat (Make an Adequate Choice)*

Description

Implements the Faire Un Choix Adéquat (FUCA) method that ranks alternatives based on the weighted sum of their ranks within each criterion for Multi-Criteria Decision Analysis (MCDA).

Usage

```
fuca(dmatrix, bcvec, weights, tiesmethod="average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix. Rows represent alternatives, and columns represent criteria. Row and column names are recommended for better output interpretation.
bcvec	A numeric vector indicating the benefit-cost orientation of each criterion. Use 1 for benefit criteria (higher values are preferred) and -1 for cost criteria (lower values are preferred). The length of this vector must match the number of columns in dmatrix.
weights	A numeric vector of criteria weights. See calcweights for details.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The Faire Un Choix Adéquat (FUCA), which translates from French to "Make an Adequate Choice," is a straightforward and intuitive MCDA method that avoids the need for data normalization (Fernando et al, 2010). The algorithm operates by ranking alternatives within each criterion and then aggregating these ranks using criterion weights.

Value

A list containing the following components:

Rank_Matrix	A numeric matrix where each row represents an alternative and each column represents a criterion. The entries are the ranks of the alternatives within each criterion.
Weighted_Rank_Scores	A numeric vector of the overall weighted rank scores for each alternative.
Ranking	A data frame containing the alternatives, their FUCA scores, and their ranks.
rank	A numeric vector representing the rank of each alternative (lower value indicates a better rank).

Author(s)

Cagatay Cebeci

References

Fernando, M. M. L., Escobedo, J. L. P., Azzaro-Pantel, C., Pibouleau, L., Domenech, S., & Aguilar-Lasserre, A. (2011). Selecting the best portfolio alternative from a hybrid multiobjective GA-MCDM approach for New Product Development in the pharmaceutical industry. In *2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM)* (pp. 159-166). IEEE.

See Also

calcweights

Examples

```
# Decision matrix (5 alternatives, 4 criteria)
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
userwei <- c(0.1, 0.4, 0.3, 0.2)

# Run FUCA with user-defined weights
resfuca_1 <- fuca(dmat, bcvec=bc, weights=userwei)
print(resfuca_1)

# Run FUCA with the calculated entropy weights
entwei <- calcweights(dmat, bcvec=bc, type="entropy")

resfuca_2 <- fuca(dmat, bcvec=bc, weights=entwei)
print(resfuca_2)
```

---

gengradwei *Gradual Modification of Initial Weights*

Description

Generates a matrix of weights by gradually modifying one criterion at a time while rescaling others accordingly.

Usage

```
gengradwei(initwei, rp = seq(0.1, 0.5, 0.1))
```

Arguments

`initwei`	A numeric vector of initial weights assigned to each criterion.
`rp`	A numeric sequence defining the percentage reduction applied to each criterion's weight, with default values ranging from 10% to 50%.

Details

This function iteratively modifies the initial weights, reducing one criterion at a time while proportionally adjusting others to maintain a valid weight distribution. Negative weights are prevented by setting a lower bound. The resulting matrix provides different weighting scenarios, which can be useful for sensitivity analysis in multi-criteria decision-making models.

Value

A numeric matrix where each row represents a different weight scenario. The first row contains the initial weights, while subsequent rows show modified weights based on the reduction percentages.

Author(s)

Cagatay Cebeci

References

Stevic, Z., Pamucar, D., Puška, A., & Chatterjee, P. (2020). Sustainable supplier selection in healthcare industries using a new MCDM method: Measurement of alternatives and ranking according to COmpromise solution (MARCOS). *Computers & Industrial Engineering, 140*, 106231. doi:10.1016/j.cie.2019.106231

Demir, G., Chatterjee, P., & Pamucar, D. (2024). Sensitivity analysis in multi-criteria decision making: A state-of-the-art research perspective using bibliometric analysis. *Expert Systems with Applications, 237*, 121660. doi:10.1016/j.eswa.2023.121660

Rachman, A. P., Ichwania, C., Mangkuto, R. A., Pradipta, J., Koerniawan, M. D., & Sarwono, J. (2024). Comparison of multi-criteria decision-making methods for selection of optimum passive design strategy. *Energy and Buildings, 314*, 114285. doi:10.1016/j.enbuild.2024.114285

See Also

`genrandwei`

Examples

```
# Define initial weights
init_weights <- c(C1 = 0.5, C2 = 0.3, C3 = 0.2)

# Generate gradual weight modifications
scenarios <- gengradwei(init_weights)
print(scenarios)
```

---

genrandwei *Random Modification of Initial Weights*

Description

Generates a set of weights by random modifying the initial weights over multiple iterations using a defined step size.

Usage

```
genrandwei(initwei, ss = NULL, niters = 5)
```

Arguments

initwei	A numeric vector representing the initial weights assigned to each criterion.
ss	The step size used for modifying weights. If NULL, it defaults to 5% of the smallest initial weight.
niters	Number of iterations to generate random weight modifications. Defaults to 5.

Details

This function iteratively adjusts the initial weight values by adding random perturbations within the defined step size. The modified weights are normalized to ensure their sum remains equal to 1. Additionally, negative weights are prevented.

Value

A numeric matrix where each row represents a different weight scenario. The first row contains the initial weights, while subsequent rows show random modifications.

Author(s)

Cagatay Cebeci

See Also

gengradwei

Examples

```
# Define initial weights
init_weights <- c(C1 = 0.4, C2 = 0.3, C3 = 0.3)

# Generate random weight modifications
scenarios <- genrandwei(init_weights, ss = 0.05, niters = 10)

# View resulting scenarios
print(scenarios)
```

---

gra *GRA: Grey Relational Analysis*

Description

Implements the Grey Relational Analysis (GRA) method for multi-criteria decision analysis by ranking alternatives based on their grey relation degrees with the ideal solution.

Usage

```
gra(dmatrix, bcvec, weights, idesol = NULL, grdmethod = "sum",
    rho = 0.5, tiesmethod="average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix. Rows are alternatives and columns are criteria.
bcvec	A numeric vector indicating the benefit/cost nature of each criterion. Use 1 for benefit (to be maximized) and -1 for cost (to be minimized). The length of bcvec must equal the number of columns in dmatrix.
weights	A vector of weights representing the relative importance of each criterion. See calcweights for details.
idesol	An optional numeric vector representing the ideal solution. If NULL (default), the ideal solution is determined from the normalized matrix (maximum for benefit criteria, minimum for cost criteria). The length must equal the number of columns in dmatrix.
grdmethod	The method used to calculate Grey Relational Degrees. Defaults to "sum", which applies a weighted sum. If set to "mean", a weighted mean is used instead. This choice affects the relative influence of criteria in the final ranking.
rho	A numeric value between 0 and 1 (default is 0.5) representing the distinguishing coefficient. It influences the grey relational coefficients.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The Grey Relational Analysis (GRA) method normalizes the decision matrix and then calculates the grey relational coefficients between each alternative and the ideal solution. The grey relation degree for each alternative is the average of its grey relational coefficients across all criteria. Alternatives are ranked based on their grey relation degrees in descending order (higher degree implies closer to the ideal solution). The grey relational coefficients for each criterion are multiplied by their respective weights before calculating the grey relation degree. To calculate Grey Relational Degrees, `"sum"` applies a weighted sum. If `grdmethod` is set to `"mean"`, a weighted mean is used instead. This choice affects the relative influence of criteria in the final ranking.

Value

A list containing the following components:

`Ideal_Solution` The ideal solution vector used for comparison.

`Grey_Relation_Coef`
: A numeric matrix of grey relational coefficients for each alternative and criterion.

`Grey_Relation_Deg`
: A numeric vector of grey relation degrees for each alternative.

`Ranking` A data frame with the ranked alternatives, their grey relation degrees, and their ranks.

`rank` A numeric vector representing the rank of each alternative (lower rank is better).

Author(s)

Cagatay Cebeci

References

Julong, D. (1989). Introduction to grey system theory. *The Journal of Grey System, 1*(1), 1-24. Deng, J.L. (1982), Control problems of grey systems, *Systems and Control Letters, 5*, 288-294.

See Also

`calcweights`, `calcnormal`

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")
```

```
# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply GRA with default settings
resgra_1 <- gra(dmatrix=dmat, bcvec=bc, weights=uw)
print(resgra_1)

# Apply GRA with a low rho value
resgra_2 <- gra(dmat, bc, weights = uw, rho = 0.3)
print(resgra_2)

# Apply GRA with a higher rho value
resgra_3 <- gra(dmatrix=dmat, bcvec=bc, weights=uw, grdmethod="sum", rho = 0.7)
print(resgra_3)

# Apply GRA with ideal solutions
ideal_solutions <- c(80, 15, 6, 16)
resgra_4 <- gra(dmatrix=dmat, bcvec=bc, weights=uw, idesol=ideal_solutions, rho=0.5)
print(resgra_4)
```

---

mabac *MABAC: Multi-Attributive Border Approximation Area Comparison*

Description

This function is an implementation of the Multi-Attributive Border Approximation Area Comparison (MABAC) method to evaluate and rank alternatives based on given criteria through normalization, weighting, and border approximation area calculations.

Usage

```
mabac(dmatrix, bcvec, weights, tiesmethod="average")
```

Arguments

`dmatrix`	Decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
`weights`	A vector of weights representing the relative importance of each criterion. Defaults to `"equal"`. See `calcweights` for details.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The MABAC method involves normalizing the decision matrix, applying weights to criteria, and calculating the border approximation area. The border approximation area represents the average value of the alternatives for each criterion. A distance matrix is computed by subtracting the border approximation area from the weighted normalized matrix, and the alternatives are ranked based on their scores.

Value

A list containing the following components:

G	The border approximation area for each criterion.
Q	The distance matrix for each alternative and criterion.
score	The final scores for each alternative.
rank	The ranks of final scores of alternatives.

Author(s)

Cagatay Cebeci

References

Pamučar, D., & Ćirović, G. (2015). The selection of transport and handling resources in logistics centers using Multi-Attributive Border Approximation area Comparison (MABAC). *Expert Systems with Applications, 42*(6), 3016-3028. <doi:10.1016/j.eswa.2014.11.057>

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply MABAC with user-defined weights
```

```
resmabac_1 <- mabac(dmatrix=dmat, bcvec=bc, weights=uw)
print(resmabac_1)

# Apply MABAC with MEREC weighting method
mw <- calcweights(dmat, bcvec=bc, type="merec", normethod="maxmin")
resmabac_2 <- mabac(dmatrix=dmat, bcvec=bc, weights=mw)
print(resmabac_2$score) # Relative utilities
print(resmabac_2$rank)  # Ranks of relative utilities
```

`macont6` *MACONT T6: Multi-Criteria Decision-Making with Comprehensive Normalization (6 steps)*

Description

MACONT 6 applies a multi-step normalization process to rank alternatives based on multiple criteria using a combination of arithmetic and geometric weighted averaging.

Usage

```
macont6(dmatrix, bcvec, weights, p=0.5, q=0.5, delta=0.5, theta=0.5, tiesmethod="average")
```

Arguments

`dmatrix`	A numeric matrix (m alternatives x n criteria) representing the initial decision matrix.
`bcvec`	A numeric vector indicating the type of criteria: `1` for benefit criteria (higher is better), `-1` for cost criteria (lower is better).
`weights`	A numeric vector representing the importance of each criterion. If not provided, equal weights are assumed.
`p`	A numeric value (between 0 and 1) representing the weight for the arithmetic averaging operator.
`q`	A numeric value (between 0 and 1) representing the weight for the geometric averaging operator.
`delta`	A numeric value (between 0 and 1) controlling the influence of rho and zeta in the ranking process.
`theta`	A numeric value (between 0 and 1) adjusting preference functions for ranking.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

1. The decision matrix `dmatrix` is normalized using three different techniques: ratio-based, max-min, and vector normalization.
2. A balanced normalized matrix is computed using weighted averages of the three normalization methods.

3. The S1 and S2 ranking values are calculated using rho and zeta components.
4. A final ranking score (S) is derived, adjusting for preference weighting.
5. Alternatives are ranked in descending order based on comprehensive scores.

Value

A list containing:

`balance_matrix`	A numeric matrix of balanced normalization values.
`S1_values`	A numeric vector of S1 ranking values.
`S2_values`	A numeric vector of S2 ranking values.
`comprehensive_scores`	A numeric vector of final ranking values (S).
`rank`	An integer vector indicating the ranked order of alternatives.

Author(s)

Cagatay Cebeci

References

Wen, Z., Liao, H., & Zavadskas, E. K. (2020). MACONT: Mixed aggregation by comprehensive normalization technique for multi-criteria analysis. *Informatica, 31*(4), 857-880. <doi:10.15388/20-INFOR417>

Nguyen, A. T. (2023). Expanding the data normalization strategy to the MACONT method for multi-criteria decision making. *Engineering, Technology & Applied Science Research, 13*(2), 10489-10495. <doi:10.48084/etasr.5672>

See Also

`topsis`, `vikor`, `electre1`

Examples

```
# Decision matrix
dmat <- matrix(c(
  5, 3, 4,
  4, 5, 2,
  3, 4, 5,
  5, 2, 3
), nrow = 4, byrow = TRUE)

bcvec <- c(1, -1, 1)
weights <- c(0.3, 0.4, 0.3)

# Run MACONT
res_macont <- macont6(dmatrix = dmat, bcvec = bcvec, weights = weights,
     p = 0.4, q = 0.3, delta = 0.6, theta = 0.7)

print(res_macont$rank)
```

`mairca` *MAIRCA: Multi-Attribute Ideal-Real Comparative Analysis*

Description

The MAIRCA is a multi-criteria decision-making method that normalizes the decision matrix and ranks alternatives based on specified criterion weights.

Usage

```
mairca(dmatrix, bcvec, weights, tiesmethod="average")
```

Arguments

`dmatrix`	Decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
`weights`	A vector of weights representing the relative importance of each criterion. Defaults to `"equal"`. See `calcweights` for details.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Value

Returns a list containing:

`Q`	Final values representing the ranking of alternatives.
`rank`	Ranking of the alternatives.

Author(s)

Cagatay Cebeci

References

Gigović, L., Pamučar, D., Bajić, Z., & Milićević, M. (2016). The combination of expert judgment and GIS-MAIRCA analysis for the selection of sites for ammunition depots. *Sustainability, 8*(4), 372. <doi:10.1080/1331677X.2018.1506706>

See Also

`calcweights`

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply MAIRCA with user-defined weights
resmairca_1 <- mairca(dmatrix=dmat, bcvec=bc, weights = uw)
print(resmairca_1)

# Apply MAIRCA with Critic weights
cw <- calcweights(dmat, bcvec=bc, type="critic")
resmairca_2 <- mairca(dmatrix=dmat, bcvec=bc, weights = cw)
print(resmairca_2)
```

`marcos` *MARCOS: Measurement of Alternatives and Ranking According to Compromise Solution*

Description

Implements the MARCOS method for MCDM by evaluating alternatives based on their relationships to ideal and anti-ideal solutions.

Usage

```
marcos(dmatrix, bcvec, weights, normethod = "ratio", tiesmethod="average")
```

Arguments

`dmatrix`	Decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
`weights`	A vector of weights representing the relative importance of each criterion. Defaults to `"equal"`. See `calcweights` for details.

normethod	The normalization method to be applied. Defaults to "ratio" for MARCOS. See calcnormal for details.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The MARCOS method (Stević et al, 2020) constructs an extended decision matrix by adding ideal and anti-ideal solutions. The matrix is normalized, and weights are applied to each criterion. Scores are calculated based on the relative distance of each alternative to the ideal and anti-ideal solutions. Final rankings are determined using the compromise solution, which integrates the positive and negative distances.

Value

A list containing the following components:

ematrix	The extended decision matrix, including ideal and anti-ideal solutions.
nexmatrix	The normalized extended decision matrix.
wmatrix	The weighted normalized decision matrix.
S	The aggregated scores for each alternative.
k_neg	The relative closeness to the anti-ideal solution.
k_pos	The relative closeness to the ideal solution.
f_k_pos	The proportion of closeness to the positive ideal solution.
f_k_neg	The proportion of closeness to the negative ideal solution.
f_k	The final compromise score for each alternative.
rank	The ranks of final compromise scores of alternatives.

Author(s)

Cagatay Cebeci

References

Stević, Ž., Pamučar, D., Puška, A., & Chatterjee, P. (2020). Sustainable supplier selection in healthcare industries using a new MCDM method: Measurement of alternatives and ranking according to COmpromise solution (MARCOS). *Computers & Industrial Engineering, 140*, 106231. <doş:10.1016/j.cie.2019.106231>

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply MARCOS with user-defined weights
resmarcos_1 <- marcos(dmatrix=dmat, bcvec=bc, weights=uw)

print(resmarcos_1$f_k)   # Final compromise scores for alternatives
print(resmarcos_1$rank)  # Ranks of the final compromise scores for alternatives
```

`maut` *MAUT: Multi-Attribute Utility Theory*

Description

The MAUT evaluates alternatives based on weighted aggregation of their utilities across multiple criteria.

Usage

```
maut(dmatrix, bcvec, weights, utilfuncs = NULL, normutil = TRUE,
    ss = 1, tiesmethod = "average")
```

Arguments

`dmatrix`	A numeric matrix representing the decision matrix. Rows are alternatives and columns are criteria.
`bcvec`	A numeric vector indicating the benefit/cost nature of each criterion. Use 1 for benefit (to be maximized) and -1 for cost (to be minimized). The length of `bcvec` must equal the number of columns in `dmatrix`.
`weights`	A vector of weights representing the relative importance of each criterion. Defaults to `"equal"`. See `calcweights` for details.

utilfuncs	An optional character vector specifying the utility function for each criterion. If NULL (default), linear utility functions are used. If a single character string is provided, it is applied to all criteria. Currently, "linear", "exp", "log", "ln", "quad", and "step" are supported.
ss	Step size used by step utility function. Default is 1.
normutil	A logical value indicating whether to normalize the utility matrix to a 0-1 range after applying the initial utility functions (default is TRUE).
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The MAUT method involves defining a utility function for each criterion that reflects the decision maker's preferences. These utility functions transform the raw performance values into utility scores. The overall utility of each alternative is then calculated as the weighted sum of its utility scores across all criteria. Alternatives are ranked based on their overall utility scores in descending order.

In MAUT based on utility theory, the decision-maker calculates the utility of each criterion and strives to best reflect their preferences in the decision-making process. Utility is shaped by subjective perceptions and risk factors. On the other hand, MAVT based on value theory, the decision-maker determines the value of each criterion and optimizes these values to achieve specific objectives. MAVT generally focuses on the direct value of each criterion and adopts a more objective approach. Conceptual differences are related to the decision-maker's preferences and risk perceptions. While working with the same matrix, different decision-makers may use either MAUT or MAVT, but the chosen approach affects how criteria should be evaluated. For example, if a decision-maker is risk-averse, they may prefer to make decisions using MAUT, as it flexibly models utility functions by taking into account risk aversion or risk-taking. On the other hand, if a decision-maker prefers a more objective approach and wants to directly optimize the value of each criterion, MAVT may be more suitable. In this case, the obtained value for each criterion takes precedence.

Value

A list containing the following components:

Utility_Matrix	A numeric matrix containing the utility scores of each alternative for each criterion.
Weighted_Sum_Scores	A numeric vector of the overall utility scores for each alternative.
rank	A numeric vector representing the rank of each alternative (lower rank is better).

Author(s)

Cagatay Cebeci

References

Keeney, R. L., & Raiffa, H. (1993). Decisions with multiple objectives: preferences and value trade-offs. Cambridge University Press. Ishizaka, A., & Nemery, P. (2013). Multi-criteria decision analysis: methods and software. John Wiley & Sons.

See Also

calcweights

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Utility functions
uf <- c("linear", "linear","quad","step")

# Apply MAUT with default linear utility functions
resmaut_linear <- maut(dmatrix=dmat, bcvec=bc, weights=uw)
print(resmaut_linear)

# Apply MAUT without normalizing the utility matrix
resmaut_nonorm <- maut(dmatrix=dmat, bcvec=bc, weights=uw, normutil = FALSE)
print(resmaut_nonorm)

# Apply MAUT without normalizing the utility matrix with utility functions
resmaut_utils <- maut(dmatrix=dmat, bcvec=bc, weights=uw, utilfuncs=uf, normutil = FALSE)
print(resmaut_utils)
```

---

mavt *MAVT: Multi-Attribute Value Theory*

Description

Implements the Multi-Attribute Value Theory (MAVT) method for multi-criteria decision analysis. This method evaluates alternatives based on weighted aggregation of their values across multiple criteria.

Usage

```
mavt(dmatrix, bcvec, weights, valfuncs = NULL, normvals = TRUE,
    ss = 1, tiesmethod = "average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix. Rows are alternatives and columns are criteria.
bcvec	A numeric vector indicating the benefit/cost nature of each criterion. Use 1 for benefit (to be maximized) and -1 for cost (to be minimized). The length of bcvec must equal the number of columns in dmatrix.
weights	A numeric vector of weights for each criterion. The length must equal the number of columns in dmatrix.
valfuncs	An optional character vector specifying the utility function for each criterion. If NULL (default), linear utility functions are used. If a single character string is provided, it is applied to all criteria. Currently, "linear", "exp", "log", "ln", "quad", and "step" are supported.
ss	Step size used by step function. Default is 1.
normvals	A logical value indicating whether to normalize the value matrix to a 0-1 range after applying the initial value functions (default is TRUE).
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The MAVT method involves defining a value function for each criterion that reflects the decision maker's preferences or judgments about the desirability of different levels of performance. These value functions transform the raw performance values into value scores. The overall value of each alternative is then calculated as the weighted sum of its value scores across all criteria. Alternatives are ranked based on their overall value scores in descending order.

Value

A list containing the following components:

Value_Matrix	A numeric matrix containing the value scores of each alternative for each criterion.
Weighted_Sum_Scores	A numeric vector of the overall value scores for each alternative.
Ranking	A data frame with the ranked alternatives, their MAVT scores, and their ranks.
rank	A numeric vector representing the rank of each alternative (lower rank is better).

Author(s)

Cagatay Cebeci

References

Keeney, R. L., & Raiffa, H. (1993). Decisions with multiple objectives: preferences and value trade-offs. Cambridge University Press.

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply MAVT with default linear value functions
resmavt_linear <- mavt(dmatrix=dmat, bcvec=bc, weights=uw)
print(resmavt_linear)

# Apply MAVT without normalizing the value matrix
resmavt_nonorm <- mavt(dmatrix=dmat, bcvec=bc, weights=uw, normvals = FALSE)
print(resmavt_nonorm)
```

---

megan *MEGAN: Multi-criteria Evaluation via Gini-weighted Attributes in Numerical Decision Matrices*

Description

A novel method to make multi-criteria decision analysis (MCDA) using various criteria, weights, and thresholds on numerical decision matrices.

Usage

```
megan(dmatrix, bcvec, weights="gini", normethod = NULL, thr = NULL,
      tht="p25", tiesmethod="average")
```

Arguments

dmatrix	A numerical decision matrix (n x m), where n represents alternatives and m represents criteria.
bcvec	A boundary condition vector, where -1 represents cost criteria and 1 represents benefit criteria.
weights	A vector of weights representing the relative importance of each criterion. Defaults to "gini" because the algorithm original applies GINI weighting if no user-defined weights supplied. See calcweights for details.
normethod	The normalization method to be used in calculation of weights. Default is NULL because this method natively applies MaxMin normalization on distance matrix. See calcnormal for details.
thr	A numeric value between 0 and 100, representing the threshold for ranking alternatives based on their superiority scores. If NULL, the threshold is determined by tht. If threshold value is zero (thr=0, the rank of alternatives is returned using rank function with the ties method "average".
tht	A character string specifying the method for determining the threshold when thr is NULL. The options are "sdev" (standard deviation), "p5" (5th percentile), and "p25" (25th percentile) of differences between the superiority scores of alternatives. Default is "p25".
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The MEGAN method involves the following steps:

1. Normalize the decision matrix.
2. Calculate criteria weights (if not provided).
3. Calculate the weighted normalized decision matrix.
4. Convert cost criteria to benefit criteria.
5. Calculate distances to the minimum value for each criterion.
6. Normalize the distance matrix.
7. Calculate row sums of the normalized distances.
8. Calculate superiority scores.
9. Rank the alternatives based on their superiority scores and the given threshold. If threshold value is zero (thr=0, the rank of scores is returned using rank function with the ties method "average".)

Value

A list containing:

dmatrix	The original decision matrix.
wmatrix	The weighted normalized decision matrix.
minvec	The vector of minimum values of criteria in generalized normalized decision matrix.
distmatrix	The matrix of distances to the minimum values.
ndistmatrix	The normalized distance matrix.
distrowsums	The vector of row sums of the normalized distances.
superiority	The vector of superiority scores (as percentage over minimum) for each alternative.
rank	The vector of ranks for each alternative.

Author(s)

Cagatay Cebeci

See Also

calcweights, calcnormal

Examples

```
data(egrids)
dmat <- as.matrix(egrids$dmat) # Decision matrix
bc <- egrids$bcvec # Benefit and cost vector
wei <- egrids$weights # Criteria weights

# Apply MEGAN without threshold
resmegan_1 <- megan(dmatrix=dmat, bcvec=bc, weights=wei, thr=0)
print(resmegan_1)

# Apply MEGAN with a fixed threshold of 10%
resmegan_2 <- megan(dmatrix=dmat, bcvec=bc, weights=wei, thr=10)
print(resmegan_2$superiority)
print(resmegan_2$rank)

# Apply MEGAN with the 'p5' threshold
resmegan_3 <- megan(dmatrix=dmat, bcvec=bc, weights=wei, thr=NULL, tht="p5")
print(resmegan_3$superiority)
print(resmegan_3$rank)

# Apply MEGAN with the standard deviation threshold
resmegan_4 <- megan(dmatrix=dmat, bcvec=bc, weights=wei, thr=NULL, tht="sdev")
print(resmegan_4$superiority)
print(resmegan_4$rank)
```

megan2 *Multi-Criteria Evaluation with Gradually-weighted Aggregation*

Description

Performs multi-criteria evaluation with gradually modified weights, aggregating numerical decision matrices for ranking.

Usage

```
megan2 (dmatrix, bcvec, weights, weimethod="random", normethod=NULL, thr=0, tht="p25",
   tiesmethod="average", ss=0.05, niters=10, rp=c(0.1, 0.5, 0.1))
```

Arguments

dmatrix	A numeric matrix representing the decision matrix, where rows correspond to alternatives and columns correspond to criteria.
bcvec	A numeric vector indicating the benefit or cost criterion. Use 1 for benefit (higher is better) and -1 for cost (lower is better).
weights	A numeric vector of weights representing the relative importance of each criterion. If omitted, equal weights are assigned.
normethod	Normalization method applied to criteria before aggregation. If NULL, raw data is used.
weimethod	Weight modification method applied to chanage criteria weights. If gradual, gradual weighting method is used. Default is random for random change using a step size.
thr	Threshold value applied in preference aggregation. Default is 0.
tht	Threshold type used in preference evaluation. Defaults to "p25".
tiesmethod	Method for handling ties in ranking. Defaults to "average".
ss	Step size for modifying weights randomly. Defaults to 0.05.
niters	Number of iterations for generating random weight modifications. Defaults to 10.
rp	A numeric vector defining the percentage reductions applied to each criterion's weight for gradual modifications. Defaults to c(0.1, 0.5, 0.1).

Details

This function applies gradual modifications to the initial weights, redistributing the importance of criteria step by step. Rankings are aggregated based on preference evaluations and adjusted to handle ties.

Value

A list containing:

weimat	A matrix containing different weight scenarios generated through gradual modifications.
rankmat	A matrix of rankings corresponding to different weight scenarios.
rank	Final aggregated ranking of alternatives based on preference evaluations.

Author(s)

Cagatay Cebeci

See Also

gengradwei, rankaggregate, megan

Examples

```
# Decision matrix
dmat <- matrix(c(5, 150, 3, 200, 4, 180), nrow = 3, byrow = TRUE)
rownames(dmat) <- paste0("A", 1:3)
colnames(dmat) <- c("C1", "C2")

# Benefit-Cost vector
bc <- c(-1, 1)

# Initial weights
init_weights <- c(C1 = 0.6, C2 = 0.4)

# Run megan2 function with gradual weight modifications
result_grad <- megan2(dmatrix = dmat, bcvec = bc, weights = init_weights, rp = c(0.1, 0.3, 0.2))

# Generated weight scenarios
print(result_grad$weimat)

print(result_grad$rank) # Aggregated rankings

# Run megan2 function with random weight modifications
result_rand <- megan2(dmatrix = dmat, bcvec = bc, weights = init_weights, ss=0.02, niters=10)

# Generated weight scenarios
print(result_rand$weimat)

print(result_rand$rank) # Aggregated rankings
```

`methodbench` *Benchmark Multiple MCDM Methods*

Description

Runs and compares multiple Multi-Criteria Decision-Making (MCDM) methods on a given decision matrix, using consistent criteria weights and types. Supports both predefined and user-supplied methods with optional parameter customization.

Usage

```
methodbench(dmatrix,bcvec, weights, normethod = "maxmin",
  mcdm = "all", params = NULL, tiesmethod = "average")
```

Arguments

`dmatrix`	A numeric decision matrix, where rows represent alternatives and columns represent criteria.
`bcvec`	A vector indicating whether each criterion is a benefit (`1`) or cost (`0`) type.
`weights`	A numeric vector of weights corresponding to each criterion.
`normethod`	Normalization method used by applicable methods (e.g., `"maxmin"`, `"ratio"`).
`mcdm`	A character vector of method names to execute. Use `"all"` to include all built-in methods. User-defined functions are also supported.
`params`	An optional named list of method-specific parameter lists. Each element name should match a method name, and its value is a list of arguments passed to that method.
`tiesmethod`	How ties in ranking should be resolved. Passed to `rank()`, e.g., `"average"`, `"first"`, etc.

Details

This function is designed to support benchmarking of multiple MCDM methods under consistent problem setups. It supports built-in methods "aras","aroman","cocoso","codas","copras","edas","electre4", "fuca","gra","mabac", "macont","marcos","mairca","maut","mavt", "megan","megan2","moora","ocra","oretes", "promethee1","promethee2", "promethee3","promethee4","promethee5","promethee6", "ram","rov","smart","topsis","vikor" and also allows integration of external functions if they return a named list including a `rank` vector.

For each method, base arguments (`dmatrix`, `bcvec`, `weights`) are passed, and if any `params` are specified, they are merged in before calling the method via `do.call()`.

If a method fails or does not return a `rank` element, a warning is displayed and the result is skipped.

Value

A list containing:

`dmatrix`	The original decision matrix.
`weights`	The input weight vector.
`rankmat`	A matrix of rankings returned by each method. Rows represent methods, columns represent alternatives.

Author(s)

Cagatay Cebeci

See Also

`calcweights`, `rankaggregate`, `rankcompare`, `rankheatmap`

Examples

```
# Sample decision matrix
dm <- matrix(c(
  10, 20, 30, 1.5, 102, 55,
  15, 25, 35, 1.6, 90, 60,
  12, 22, 32, 1.7, 100, 58,
  13, 24, 33, 1.8, 95, 57,
  14, 26, 37, 1.9, 98, 59,
  11, 23, 31, 1.65, 101, 56,
  16, 27, 36, 1.55, 97, 61,
  17, 28, 38, 1.7, 99, 63,
  18, 29, 39, 1.8, 94, 62,
  19, 30, 40, 1.75, 96, 64
), nrow = 10, byrow = TRUE)
colnames(dm) <- paste0("C", 1:ncol(dm))
rownames(dm) <- paste0("ALT", 1:nrow(dm))

# Benefit-Cost vector
bc <- c(1, -1, 1, -1, 1, 1)

# User-defined weights
userwei <- c(0.3, 0.1, 0.2, 0.1, 0.2, 0.1)

prmlist <- list(
      aras      = list(),
      aroman    = list(lambda = 0.5, beta = 0.5),
      cocoso    = list(lambda = 0.5),
      codas     = list(thr = 0.1),
      copras    = list(),
      electre1  = list(),
      electre2  = list(),
      electre3  = list(),
      electre4  = list(),
      fuca      = list(),
```

```
      gra       = list(idesol = NULL, grdmethod = "sum", rho = 0.5),
      mabac     = list(),
      macont6   = list(p = 0.5, q = 0.5, delta = 0.5, theta = 0.5),
      marcos    = list(),
      mairca    = list(),
      maut      = list(utilfuncs = NULL, normutil = TRUE, ss = 1),
      mavt      = list(valfuncs = NULL, normvals = TRUE, ss = 1),
      megan     = list(normethod = "ratio", thr = 0, tht = "sdev"),
      megan2    = list(normethod = "ratio", thr = 0, tht = "sdev"),
      moora     = list(),
      ocra      = list(),
      oretes    = list(domplot = FALSE),
      promethee1 = list(),
      promethee2 = list(),
      promethee3 = list(strict = FALSE),
      promethee4 = list(alpha = 0.2),
      promethee5 = list(g = 0, l = 100),
      promethee6 = list(varmethod = "abs_sum"),
      ram       = list(normethod = "sum"),
      rov       = list(normethod = "maxmin"),
      smart     = list(),
      topsis    = list(normethod = "maxmin"),
      vikor     = list(normethod = "maxmin", v = 0.8),
      waspas    = list(normethod = "linear", v = 0.5),
      wpm       = list(normethod = "vector"),
      wsm       = list()
)

# prmlist$electre4 <- list(
#      p = c(0.6, 0.7, 0.5, 0.6, 0.6, 0.8),
#      q = c(0.3, 0.35, 0.25, 0.3, 0.3, 0.4),
#      v = c(0.9, 1.0, 0.8, 0.9, 0.9, 1.1))

# Compare all methods with default settings using user-defined weights
rescomp_1 <- methodbench(dmatrix=dm, bcvec=bc, weights=userwei, params=prmlist, mcdm="all")

print(rescomp_1$rankmat)

rankheatmap(rescomp_1$rankmat, colpal=1, cellnotes=TRUE, tcol="black")

# Overall ranks and flow of dominance
resoverall <- rankaggregate(rescomp_1$rankmat, tiesmethod="average",
   topk = 3, damp = 0.5, niters = 200, tol = 1e-4)
str(resoverall)

# Compare selected methods with 'equal' weights
methodlist <- c("aras", "cocoso", "copras", "edas", "electre4", "mairca",
   "mabac", "marcos", "megan", "ocra", "oretes", "promet1", "promet6",
   "topsis", "vikor", "waspas")

equwei <- calcweights(dm, bcvec=bc, type="equal")
rescomp_2 <- methodbench(dmatrix = dm, bcvec = bc, weights = equwei,
   mcdm = methodlist, params=prmlist)
```

```
print(rescomp_2$rankmat)
rankheatmap(rescomp_2$rankmat, colpal=4, cellnotes=TRUE, tcol="white")

# Overall ranks and outranking
resoverall <- rankaggregate(rescomp_2$rankmat, tiesmethod="average",
   topk = 3, damp = 0.5, niters = 200, tol = 1e-4)
print(resoverall$preference_ranking)
print(resoverall$preference_table)

# Compare selected methods with 'gini' weights
giniwei <- calcweights(dm, bcvec=bc, type="gini")
rescomp_3 <- methodbench(dmatrix = dm, bcvec = bc, weights = giniwei,
   mcdm = methodlist, params=prmlist)
print(rescomp_3$rankmat)
rankheatmap(rescomp_3$rankmat, colpal=1, cellnotes=TRUE, tcol="white")

# Overall ranks and outranking
resoverall <- rankaggregate(rescomp_3$rankmat, tiesmethod="average", topk = 3,
   damp = 0.5, niters = 200, tol = 1e-4)
print(resoverall)

# Custom function example
# A Dummy user-defined method
johndoe <- function(dmatrix, bcvec, weights, janefactor = 1) {
  score <- rowSums(scale(dmatrix) * weights) * janefactor
  ranking <- rank(-score, ties.method="average")
  results <- list(score = score, rank=ranking)
  return(results)
}

prmlist <- list(
   aras = list(),
   vikor = list(v = 0.5),
   waspas = list(v = 0.5),
   johndoe = list(jane = 0.25)
)

# Compare user-defined 'johndoe' with popular MCDM methods
rescomp_user <- methodbench(
  dmatrix = dm,
  bcvec = bc,
  weights = userwei,
  mcdm = c("edas", "topsis", "vikor", "aras", "waspas", "oreste", "johndoe"),
  params = prmlist
)

print(rescomp_user$rankmat)

rankheatmap(rescomp_user$rankmat, colpal = 3, cellnotes = TRUE, tcol = "black")

# Aggregate ranking
resoverall_user <- rankaggregate(rescomp_user$rankmat, tiesmethod = "average", topk = 3)
```

```
print(resoverall_user$preference_ranking)
print(resoverall_user$preference_table)
```

`moora` *MOORA: Multi-Objective Optimization by Ratio Analysis*

Description

The Multi-Objective Optimization by Ratio Analysis (MOORA) method for multi-criteria decision analysis ranks alternatives based on the balance between their performance on benefit and cost criteria.

Usage

```
moora(dmatrix, bcvec, weights, tiesmethod="average")
```

Arguments

`dmatrix`	A numeric matrix representing the decision matrix. Rows are alternatives and columns are criteria.
`bcvec`	A numeric vector indicating the benefit/cost nature of each criterion. Use 1 for benefit (to be maximized) and -1 for cost (to be minimized). The length of `bcvec` must equal the number of columns in `dmatrix`.
`weights`	A vector of weights representing the relative importance of each criterion. See `calcweights` for details.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The Multi-Objective Optimization by Ratio Analysis (MOORA) method first normalizes the decision matrix using vector normalization. Then, for each alternative, it calculates an optimization value by summing the weighted normalized values of the benefit criteria and subtracting the weighted normalized values of the cost criteria. Alternatives are ranked based on these optimization values in descending order (higher value implies better performance).

Value

A list containing the following components:

`Normalized_Matrix`	The normalized decision matrix.
`Weighted_Normalized_Matrix`	The weighted normalized decision matrix.
`Optimization_Values`	A numeric vector of the optimization values for each alternative.
`Ranking`	A data frame with the ranked alternatives, their MOORA scores, and their ranks.
`rank`	A numeric vector representing the rank of each alternative (lower rank is better).

Author(s)

Cagatay Cebeci

References

Brauers, W. K., & Zavadskas, E. K. (2006). The MOORA method and its application to privatization in a transition economy. *Control and Cybernetics, 35*(2), 445-469.

See Also

calcweights

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# MOORA with user-defined weights
resmoora_1 <- moora(dmatrix = dmat, bcvec = bc, weights = uw)
print(resmoora_1)

# MOORA with equal weights
equw <- calcweights(dmat, bcvec=bc, type="equal")
resmoora_2 <- moora(dmatrix =  dmat, bcvec = bc, weights = equw)
print(resmoora_2)

# MOORA with entropy weights
entw <- calcweights(dmat, bcvec=bc, type="entropy")
resmoora_3 <- moora(dmatrix = dmat, bcvec = bc, weights = entw)
print(resmoora_3)
```

---

ocra *OCRA: Operational Competitiveness Rating*

Description

Computes rankings for decision alternatives using the OCRA method, which evaluates benefit and cost criteria relative to reference values.

Usage

```
ocra(dmatrix, bcvec, weights, tiesmethod = "average")
```

Arguments

`dmatrix`	A decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	A vector indicating the criteria type: 1 for benefit criteria, -1 for cost criteria.
`weights`	A numeric vector representing the relative importance of each criterion, corresponding to the columns of the decision matrix.
`tiesmethod`	Method for resolving ties in ranking, defaulting to ‘"average"‘.

Value

A list with the following components:

`reference_values`	Reference values for each criterion (max for benefit, min for cost).
`cost_impact_I`	Impact scores for cost criteria (higher values indicate more inefficiency).
`benefit_impact_O`	Impact scores for benefit criteria (higher values indicate more deviation from ideal benefit levels).
`total_performance_P`	Final performance scores based on cost and benefit impact values. Lower values indicate better alternatives.
`rank`	Ranking of alternatives based on performance scores. Lower-ranked alternatives are considered better.

Author(s)

Cagatay Cebeci

References

Madic, M., Petkovic, D., & Radovanovic, M. (2015). Selection of non-conventional machining processes using the OCRA method. *Serbian Journal of Management, 10*(1), 61-73.

Examples

```
dmatrix <- matrix(c(
  10, 50, 30,
  20, 60, 20,
  30, 40, 40,
  40, 30, 50
), nrow = 4, byrow = TRUE)

bcvec <- c(1, -1, 1)  # Benefit (1), Cost (-1), Benefit (1)
weights <- c(0.3, 0.4, 0.3)  # Criterion weights

resocra <- ocra(dmatrix, bcvec, weights)
print(resocra)
```

`oretes` *ORETES: Organísation, Rangemént et Estímatíon En Synthèse*

Description

The "Organísation, Rangemént et Estímatíon En Synthèse (ORETES)" is an MCDM technique based on outranking relations and graph theory.The function calculates lambda values, psi ranks, a normalized dominance matrix, outranking flows, and provides a final ranking of alternatives. It also offers an optional visualization of the dominance graph.

Usage

```
oretes(dmatrix, bcvec, weights, tiesmethod = "average", domplot = FALSE)
```

Arguments

`dmatrix`	A numeric matrix or data frame representing the decision matrix. Rows are alternatives and columns are criteria.
`bcvec`	A numeric vector of the same length as the number of criteria, indicating the type of each criterion. Use `1` for benefit criteria (higher is better) and `-1` for cost criteria (lower is better).
`weights`	A numeric vector of the same length as the number of criteria, representing the weights (importance) of each criterion. These do not necessarily need to sum to 1.
`tiesmethod`	A character string specifying how ties are handled when ranking. Passed to `rank`. Common options include `"average"` (default), `"first"`, `"last"`, `"random"`, `"max"`, `"min"`.
`domplot`	A logical value. If `TRUE`, the dominance graph illustrating the outranking relationships between alternatives will be plotted. Requires the `igraph` package.

Details

The ORETES method proceeds through several steps:

1. **Lambda Values Calculation**: Normalizes criterion values for each alternative to a scale of 0 to 1, where 1 represents the ideal performance and 0 the worst.
2. **Psi Matrix Calculation**: Ranks the lambda values for each alternative across its criteria. This provides an internal ranking of criteria importance for each alternative.
3. **Dominance Matrix Calculation**: Computes the strength of preference (dominance) of one alternative over another. For each pair of alternatives (A, B), it sums the weights of the criteria where A performs better than B. The matrix is then normalized.
4. **Outranking Flow Calculation**: Determines the net outranking flow for each alternative by subtracting its incoming dominance strength from its outgoing dominance strength. A higher net flow indicates a stronger overall preference.
5. **Final Ranking**: Alternatives are ranked based on their net outranking flow, from highest to lowest.

The dominance graph, if plotted, visually represents the normalized dominance matrix, with arrows indicating the direction and strength of preference between alternatives. Thicker arrows indicate stronger dominance.

Value

A list containing the following components:

`lambda_matrix` A numeric matrix of normalized lambda values (0-1) for each alternative across each criterion.

`psi_matrix` A numeric matrix where each row represents an alternative and contains the ranks of its lambda values across criteria.

`dominance_matrix` A numeric matrix showing the normalized strength of dominance of row alternatives over column alternatives.

`outranking_flow` A numeric vector containing the net outranking flow for each alternative.

`rank` A numeric vector indicating the final rank of each alternative, where 1 is the best rank.

Note

This implementation of ORETES is an adaptation focused on the core principles of lambda values, dominance relations, and outranking flow. It simplifies some of the more intricate aspects of the original method, such as specific indifference/preference thresholds and detailed preference functions, for broader applicability in MCDM.

Author(s)

Cagatay Cebeci

References

Roubens, M. (1982). Preference relations on actions and criteria in multicriteria decision making. European Journal of Operational Research, 10(1), 51-55. <doi:10.1016/0377-2217(82)90131-X>

See Also

graph_from_data_frame, plot.igraph

Examples

```
# Sample decision matrix
dmatrix <- matrix(c(
  10, 50, 30,
  20, 60, 20,
  30, 40, 40,
  40, 30, 50),
  nrow = 4, byrow = TRUE,
  dimnames = list(paste0("A", 1:4), paste0("C", 1:3)
))

# Define benefit (1) or cost (-1) for each criterion
bcvec <- c(1, -1, 1)

# Define weights for each criterion
weights <- c(0.3, 0.4, 0.3) # Sum does not necessarily need to be 1

# ORETES without plotting the dominance graph
resoreste_1 <- oretes(dmatrix, bcvec, weights, domplot = FALSE)
print(resoreste_1)

# ORETES with the dominance graph
resoreste_2 <- oretes(dmatrix, bcvec, weights, domplot = TRUE)
print(resoreste_2)

# Zero weights (no dominance relationships)
  zerowei <- c(0, 0, 0)
  resoreste_3 <- oretes(dmatrix, bcvec, zerowei, domplot = TRUE)
  print(resoreste_3)
```

---

parcorplot *Parallel Coordinate Plot*

Description

This function creates a parallel coordinate plot using the provided data frame or matrix. Each row represents a different method, and each line in the plot corresponds to the values for that method.

Usage

```
parcorplot(x, xl = NULL, yl = NULL, lt = NULL, colpal = NULL)
```

Arguments

x	A data frame or matrix where the row names are the names of MCDM methods, and the columns contain the corresponding rank values for alternatives computed by the methods.
xl	Label for the x-axis. Default is "x".
yl	Label for the y-axis. Default is "y".
lt	Title for the legend. Default is "x".
colpal	A vector of colors for the lines. If NULL (default), a predefined color palette or `topo.colors` is used.

Value

No return value, but a plot displaying the parallel coordinates for each method.

Author(s)

Cagatay Cebeci

Examples

```
# Sample rank matrix
rankmat <- data.frame(
  ALT1 = c(3.0, 3.0, 2.5, 3.0, 3.0, 3.0, 2.0, 1.0),
  ALT2 = c(1, 1, 1, 1, 1, 1, 1, 2),
  ALT3 = c(2.0, 2.0, 2.5, 2.0, 2.0, 2.0, 3.0, 3.0)
)
rownames(rankmat) <- c(
  "ARAS", "EDAS", "ELECTRE4", "MABAC", "MARCOS", "MEGAN", "TOPSIS", "VIKOR"
)

parcorplot(rankmat, xl = "Alternatives", yl = "Ranks", lt = "MCDA Methods")
```

---

promethee1 — *PROMETHEE I Method for Partial Ranking in Multi-Criteria Decision Analysis*

Description

Implements the PROMETHEE I method to compute positive and negative preference flows, generate pairwise outranking comparisons, and determine a partial ranking of alternatives.

Usage

```
promethee1(dmatrix, bcvec, weights, normethod = NULL,
    prefuncs = NULL, thr = NULL, tiesmethod = "average")
```

Arguments

dmatrix	A numeric matrix or data frame representing the decision matrix. Rows are alternatives; columns are criteria.
bcvec	A numeric vector of -1 (cost) or 1 (benefit) values specifying the optimization direction of each criterion.
weights	A numeric vector of weights for criteria. Must sum to approximately 1.
normethod	Optional normalization method passed to calcnormal. If NULL, no normalization is applied.
prefuncs	A character vector or single value specifying preference function types for each criterion. Supported types include "linear", "usual", "quasi", "v-shape", "level", "linear-indif", and "gaussian".
thr	A list of threshold parameters for preference functions. Each element may include p, q, s, or sigma depending on the function type.
tiesmethod	Character string specifying tie-breaking method for final Phi+ ranking. Passed to rank.

Details

PROMETHEE I allows pairwise comparison of alternatives under multiple criteria using weighted preference functions. The method computes how much an alternative is preferred to others (Phi+) and how much it is outranked (Phi-), without producing a complete ordering in all cases. Indifference and incomparability are explicitly represented.

You can customize the behavior via different preference functions and thresholds for each criterion. When no thresholds are provided, default ones based on column differences are used.

Value

A list containing:

Phi_plus	A vector of positive preference flow values for each alternative.
Phi_minus	A vector of negative preference flow values for each alternative.
Partial_Ranking	A matrix indicating pairwise outranking relations ("P", "Q", "I", "R", "=").
rank	A numeric vector of ranks based on descending Phi+ values.

Author(s)

Cagatay Cebeci

Examples

```
# Sample decision matrix
dm <- matrix(c(10, 5, 8,
               6, 9, 7,
               8, 7, 9), nrow = 3, byrow = TRUE)
rownames(dm) <- paste0("A", 1:3)
colnames(dm) <- paste0("C", 1:3)
```

```
# All criteria are benefits
bc <- c(1, 1, 1)

# Weights sum to 1
userwei <- c(0.3, 0.4, 0.3)

# Preference functions and thresholds
prefuncs <- c("linear", "v-shape", "level")
thr <- list(
  list(p = 5),         # Linear
  list(p = 4),         # V-shape
  list(q = 1, p = 3)   # Level
)

# Apply PROMETHEE I
respromethee1 <- promethee1(dmatrix=dm, bcvec=bc, weights=userwei,
   prefuncs = prefuncs, thr = thr)
print(respromethee1)
```

---

promethee2 *PROMETHEE II: Preference Ranking Organization METHod for Enrichment Evaluations (Method II)*

Description

Performs the PROMETHEE II method for complete ranking of alternatives in a Multi-Criteria Decision Analysis (MCDA) problem.

Usage

```
promethee2(dmatrix, bcvec, weights, normethod = NULL, prefuncs = NULL,
    thr = NULL, tiesmethod="average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix, where rows correspond to alternatives and columns correspond to criteria.
bcvec	A numeric vector indicating the benefit or cost criterion. Use 1 for benefit (higher is better) and -1 for cost (lower is better) for each criterion. The length of bcvec must be equal to the number of columns in dmatrix.
weights	A vector of weights representing the relative importance of each criterion. See calcweights for details.
normethod	The normalization method if applied. Defaults to NULL. See calcnormal for details.

`prefuncs` A character vector specifying the preference function to be used for each criterion. The length of `prefuncs` must be equal to the number of columns in `dmatrix`, or it can be a single character string to apply the same preference function to all criteria. Available preference functions are: "linear", "usual", "quasi-criterion", "v-shape", and "level".

`thr` A list of threshold values for the preference functions. The structure depends on the chosen preference function for each criterion. It should be a list with the same length as the number of criteria. Each element of the list can be a named list or a numeric vector containing the required threshold parameters (e.g., `list(p = 50)` for "linear" and "v-shape", `list(s = 10)` for "quasi-criterion", `list(q = 5, p = 20)` for "level"). If a single list or numeric vector is provided, it will be applied to all criteria. If `NULL`, default thresholds based on the maximum absolute difference for each criterion will be used (with a warning).

`tiesmethod` The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The PROMETHEE II method calculates a complete preorder of alternatives. It extends PROMETHEE I by calculating the net outranking flow (Phi_net) for each alternative, which is the difference between positive (Phi_plus) and negative (Phi_minus) outranking flows. Then, alternatives can be ranked based on their Phi_net values.

Value

A list containing the following components:

`Phi_plus` A numeric vector of positive outranking flows for each alternative.

`Phi_minus` A numeric vector of negative outranking flows for each alternative.

`Phi_net` A numeric vector of net outranking flows for each alternative.

`rank` A numeric vector representing the complete ranking of alternatives based on the descending order of `Phi_net` values (higher is better). Ties are handled using the `"average"` method.

Author(s)

Cagatay Cebeci

References

Brans, J. P., & Mareschal, B. (1988). Geometrical representations for MCDA. *European Journal of Operational Research, 34*(1), 69-77. <doi:10.1016/0377-2217(88)90456-0>

Brans, J. P., Mareschal, B. (2005). Promethee Methods. In: *Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol 78.* Springer, New York, NY. <doi:10.1007/0-387-23081-5_5>

Behzadian, M., Kazemzadeh, R. B., Albadvi, A., & Aghdasi, M. (2010). PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research, 200*(1), 198-215. <doi:10.1016/j.ejor.2009.01.021>

Brans, J. P., De Smet, Y. (2016). PROMETHEE Methods. In: *Greco, S., Ehrgott, M., Figueira, J. (eds) Multiple Criteria Decision Analysis. International Series in Operations Research & Management Science, 233*. Springer, New York, NY. <doi:10.1007/978-1-4939-3094-4_6>

See Also

promethee1, promethee3, promethee4, promethee5, promethee6

Examples

```
# Decision matrix
dmat <- matrix(c(
  7, 12, 8, 6, 14,
  12, 18, 11, 8, 22,
  8, 14, 11, 7, 10,
  1.2, 1.9, 1.2, 0.8, 1.1,
  0, 0, 1, 0, 1,
  0, 1, 0, 1, 2
),
nrow = 5, byrow = FALSE)

colnames(dmat) <- c("C1", "C2", "C3", "C4", "C5", "C6")
rownames(dmat) <- c("A", "B", "C", "D", "E")

# Benefit-Cost vector
bc <- c(1, 1, 1, 1, -1, 1)

# Weights
uw <- c(0.13, 0.08, 0.20, 0.17, 0.12, 0.3)

# Threshold values
repeat_list <- list(list(p = 50), list(p = 30))
tvals <- rep(repeat_list, length.out = ncol(dmat))

# Preference function
pf <- rep("linear", ncol(dmat))

# Run PROMETHEE II function
respromethee2 <- promethee2(dmatrix=dmat, bcvec=bc, weights=uw,
    thr = tvals, prefuncs = pf)

# Phi+ values
print(respromethee2$Phi_plus)

# Phi- values
print(respromethee2$Phi_minus)

# Phi_net values
print(respromethee2$Phi_net)

# Complete ranking (PROMETHEE II)
print(respromethee2$rank)
```

```
# Run with a single threshold value applied to all criteria (linear)
tvals_single <- list(p = 40)
respromethee2_single <- promethee2(dmatrix=dmat, bcvec=bc, weights=uw,
    thr = tvals_single, prefuncs = "linear")

# Phi_net values for single threshold
print(respromethee2_single$Phi_net)

# Complete ranking for single threshold
print(respromethee2_single$rank)

# Run with different preference functions and mixed thresholds
tvals_mixed <- list(list(q = 10, p = 30), list(p = 20), list(q = 10, p = 30),
   list(q = 10, p = 30), list(q = 10, p = 30), list(q = 10, p = 30))
pf_mixed <- c("linear", "linear", "linear", "linear", "level", "level")
respromethee2_mixed <- promethee2(dmatrix=dmat, bcvec=bc, weights=uw,
   thr = tvals_mixed, prefuncs = pf_mixed)

# Phi_net values with mixed thresholds
print(respromethee2_mixed$Phi_net)

# Complete ranking with mixed thresholds
print(respromethee2_mixed$rank)
```

`promethee3` *PROMETHEE III: Preference Ranking Organization METHod for Enrichment Evaluations (Method III)*

Description

Performs the PROMETHEE III method for complete ranking of alternatives in a Multi-Criteria Decision Analysis (MCDM) problem, allowing for interval-based ranking.

Usage

```
promethee3(dmatrix, bcvec, weights, normethod = NULL,
   prefuncs = NULL, thr = NULL, strict = FALSE, tiesmethod="average")
```

Arguments

`dmatrix`	A numeric matrix representing the decision matrix, where rows correspond to alternatives and columns correspond to criteria.
`bcvec`	A numeric vector indicating the benefit or cost criterion. Use 1 for benefit (higher is better) and -1 for cost (lower is better) for each criterion. The length of `bcvec` must be equal to the number of columns in `dmatrix`.
`weights`	A vector of weights representing the relative importance of each criterion. See `calcweights` for details.

`normethod` An optional character string specifying the normalization method for the decision matrix because there is usually no need to normalize the decision matrix with PROMETHEE III. If not `NULL`, the `calcnormal` function will be used. Common options might include "linear", "vector", and "maxmin". Defaults to `NULL`, no normalization is performed. See `calcnormal` for details.

`prefuncs` A character vector specifying the preference function to be used for each criterion. The length of `prefuncs` must be equal to the number of columns in `dmatrix`, or it can be a single character string to apply the same preference function to all criteria. Available preference functions are: "linear", "usual", "quasi-criterion", "v-shape", and "level".

`thr` A list of threshold values for the preference functions. The structure depends on the chosen preference function for each criterion. It should be a list with the same length as the number of criteria. Each element of the list can be a named list or a numeric vector containing the required threshold parameters (e.g., `list(p = 50)` for "linear" and "v-shape", `list(s = 10)` for "quasi-criterion", `list(q = 5, p = 20)` for "level"). If a single list or numeric vector is provided, it will be applied to all criteria. If `NULL`, default thresholds based on the maximum absolute difference for each criterion will be used (with a warning).

`strict` A logical value indicating whether to perform strict ranking (`TRUE`) or interval-based ranking (`FALSE`). If `TRUE`, the result is identical to PROMETHEE II. Default is `FALSE`.

`tiesmethod` The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The PROMETHEE III method extends PROMETHEE II to allow for interval-based ranking of alternatives, providing a more nuanced result that can account for uncertainty. It calculates positive (Phi_plus), negative (Phi_minus), and net (Phi_net) outranking flows. The final ranking depends on the `strict` parameter. If `strict = FALSE`, alternatives are ranked based on intervals (which, in this basic implementation, are point estimates equal to Phi_net). If `strict = TRUE`, the ranking is based on Phi_net, as in PROMETHEE II.

Value

A list containing the following components:

`Phi_plus` A numeric vector of positive outranking flows for each alternative.

`Phi_minus` A numeric vector of negative outranking flows for each alternative.

`Phi_net` A numeric vector of net outranking flows for each alternative.

`rank` A numeric vector representing the complete ranking of alternatives. If `strict = FALSE`, this represents an interval-based ranking (in this implementation, the intervals are point estimates). If `strict = TRUE`, this is a strict ranking based on `Phi_net` values (identical to PROMETHEE II). Ties are handled using the "average" method.

Author(s)

Cagatay Cebeci

References

Brans, J. P., & Mareschal, B. (1988). Geometrical representations for MCDA. *European Journal of Operational Research, 34*(1), 69-77. <doi:10.1016/0377-2217(88)90456-0>

Brans, J. P., Mareschal, B. (2005). Promethee Methods. In: *Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol 78.* Springer, New York, NY. <doi:10.1007/0-387-23081-5_5>

Behzadian, M., Kazemzadeh, R. B., Albadvi, A., & Aghdasi, M. (2010). PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research, 200*(1), 198-215. <doi:10.1016/j.ejor.2009.01.021>

Brans, J. P., De Smet, Y. (2016). PROMETHEE Methods. In: *Greco, S., Ehrgott, M., Figueira, J. (eds) Multiple Criteria Decision Analysis. International Series in Operations Research & Management Science, 233*. Springer, New York, NY. <doi:10.1007/978-1-4939-3094-4_6>

See Also

promethee1, promethee2, promethee4, promethee5, promethee6

Examples

```
# Decision matrix
dmat <- matrix(c(5, 150, 3, 200, 4, 180), nrow = 3, byrow = TRUE)
rownames(dmat) <- paste0("A", 1:3)
colnames(dmat) <- c("C1", "C2")

# Benefit-Cost vector
bc <- c(-1, 1)

# User-defined weights
uw <- c(0.6, 0.4)

# Threshold values
tvals <- list(list(p = 50), list(p = 30))

# Preference functions
pf <- c("linear", "linear")

# Run PROMETHEE III function (strict = FALSE for interval ranking)
respromethee3_interval <- promethee3(dmatrix=dmat, bcvec=bc, weights=uw,
   prefuncs = pf, thr = tvals, strict = FALSE)
print(respromethee3_interval)

# Run PROMETHEE III function (strict = TRUE for strict ranking)
respromethee3_strict <- promethee3(dmatrix=dmat, bcvec=bc, weights=uw,
   prefuncs = pf, thr = tvals, strict = TRUE)
print(respromethee3_strict)
```

promethee4 *PROMETHEE IV: Preference Ranking Organization METHod for Enrichment Evaluations (Method IV)*

Description

Performs the PROMETHEE IV method for complete ranking of alternatives in a Multi-Criteria Decision Analysis (MCDM) problem, considering the average outranking flow.

Usage

```
promethee4(dmatrix, bcvec, weights, normethod = NULL,
   prefuncs = NULL, thr = NULL, alpha = 0.2, tiesmethod="average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix, where rows correspond to alternatives and columns correspond to criteria.
bcvec	A numeric vector indicating the benefit or cost criterion. Use 1 for benefit (higher is better) and -1 for cost (lower is better) for each criterion. The length of bcvec must be equal to the number of columns in dmatrix.
weights	A vector of weights representing the relative importance of each criterion. See calcweights for details.
normethod	An optional character string specifying the normalization method for the decision matrix because there is usually no need to normalize the decision matrix with PROMETHEE IV. If not NULL, the calcnormal function will be used. Common options might include "linear", "vector", etc. Defaults to NULL, no normalization is performed. See calcnormal for details.
prefuncs	A character vector specifying the preference function to be used for each criterion. The length of pref_functions must be equal to the number of columns in dmatrix, or it can be a single character string to apply the same preference function to all criteria. Available preference functions are: "linear", "usual", "quasi-criterion", "v-shape", and "level".
thr	A list of threshold values for the preference functions. The structure depends on the chosen preference function for each criterion. It should be a list with the same length as the number of criteria. Each element of the list can be a named list or a numeric vector containing the required threshold parameters (e.g., list(p = 50) for "linear" and "v-shape", list(s = 10) for "quasi-criterion", list(q = 5, p = 20) for "level"). If a single list or numeric vector is provided, it will be applied to all criteria. If NULL, default thresholds based on the maximum absolute difference for each criterion will be used (with a warning).
alpha	A numeric value between 0 and 1 representing the influence of the average outranking flow in the final ranking. A higher value gives more importance to the average flow. Default value is 0.2.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

PROMETHEE IV ranks alternatives by considering their deviation from the average outranking flow. It calculates positive (`Phi_plus`), negative (`Phi_minus`), and net (`Phi_net`) outranking flows, and then computes the average net outranking flow (`Phi_avg`). The final ranking is based on the adjusted net outranking flow: `Phi_net - alpha * Phi_avg`.

Value

A list containing the following components:

`Phi_plus`	A numeric vector of positive outranking flows for each alternative.
`Phi_minus`	A numeric vector of negative outranking flows for each alternative.
`Phi_net`	A numeric vector of net outranking flows for each alternative.
`Phi_avg`	The average of the net outranking flows.
`rank`	A numeric vector representing the complete ranking of alternatives, considering the average outranking flow.

Author(s)

Cagatay Cebeci

References

Brans, J. P., & Mareschal, B. (1988). Geometrical representations for MCDA. *European Journal of Operational Research, 34*(1), 69-77. <doi:10.1016/0377-2217(88)90456-0>

Brans, J. P., Mareschal, B. (2005). Promethee Methods. In: *Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol 78.* Springer, New York, NY. <doi:10.1007/0-387-23081-5_5>

Behzadian, M., Kazemzadeh, R. B., Albadvi, A., & Aghdasi, M. (2010). PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research, 200*(1), 198-215. <doi:10.1016/j.ejor.2009.01.021>

Brans, J. P., De Smet, Y. (2016). PROMETHEE Methods. In: *Greco, S., Ehrgott, M., Figueira, J. (eds) Multiple Criteria Decision Analysis. International Series in Operations Research & Management Science, 233*. Springer, New York, NY. <doi:10.1007/978-1-4939-3094-4_6>

See Also

`promethee1`, `promethee2`, `promethee3`, `promethee5`, `promethee6`

Examples

```
# Decision matrix
dmat <- matrix(c(5, 150, 3, 200, 4, 180), nrow = 3, byrow = TRUE)
rownames(dmat) <- paste0("A", 1:3)
colnames(dmat) <- c("C1", "C2")

# Benefit-Cost vector
bc <- c(-1, 1)
```

```
# Weights
uw <- c(0.6, 0.4)

# Threshold values
tvals <- list(list(p = 50), list(p = 30))

# Preference function
pf <- c("linear", "linear")

# Run PROMETHEE IV function with alpha = 0.2
respromethee4_1 <- promethee4(dmatrix=dmat, bcvec=bc, weights=uw,
  thr= tvals, prefuncs = pf, alpha = 0.2)
print(respromethee4_1)

# Run PROMETHEE IV function with alpha = 0.8
respromethee4_2 <- promethee4(dmatrix=dmat, bcvec=bc, weights=uw,
  thr = tvals, prefuncs = pf, alpha = 0.8)
print(respromethee4_2)
```

`promethee5` *PROMETHEE V: Preference Ranking Organization METHod for Enrichment Evaluations (Method V)*

Description

PROMETHEE V method applies multi-criteria decision analysis, which ranks alternatives considering both their net outranking flow and feasibility constraints.

Usage

```
promethee5(dmatrix, bcvec, weights, normethod = NULL,
    prefuncs = NULL, thr = NULL, g = 0, l = 100, tiesmethod="average")
```

Arguments

`dmatrix`	A numeric matrix representing the decision matrix. Rows represent alternatives, and columns represent criteria.
`bcvec`	A numeric vector indicating the benefit/cost orientation of each criterion. Use 1 for benefit criteria (to be maximized) and -1 for cost criteria (to be minimized).
`weights`	A vector of weights representing the relative importance of each criterion. See `calcweights` for details.
`normethod`	An optional character string specifying the normalization method for the decision matrix because there is usually no need to normalize the decision matrix with PROMETHEE V. If not `NULL`, the `calcnormal` function will be used. Common options might include "linear", "vector", etc. Defaults to `NULL`, no normalization is performed. See `calcnormal` for details.

prefuncs	A character vector specifying the preference function to be used for each criterion. The length of prefuncs must be equal to the number of columns in dmatrix, or it can be a single character string to apply the same preference function to all criteria. Available preference functions are: "linear", "usual", "quasi-criterion", "v-shape", and "level".
thr	A list of threshold values for the preference functions. The structure depends on the chosen preference function for each criterion. It should be a list with the same length as the number of criteria. Each element of the list can be a named list or a numeric vector containing the required threshold parameters (e.g., list(p = 50) for "linear" and "v-shape", list(s = 10) for "quasi-criterion", list(q = 5, p = 20) for "level"). If a single list or numeric vector is provided, it will be applied to all criteria. If NULL, default thresholds based on the maximum absolute difference for each criterion will be used (with a warning).
g	A numeric value representing the lower limit for the net outranking flow (ϕ_{net}) for an alternative to be considered feasible. Must be non-negative. Default is 0.
l	A numeric value representing the upper limit for the net outranking flow (ϕ_{net}) for an alternative to be considered feasible. Must be non-negative. Default 100.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

PROMETHEE V method, an extension of the PROMETHEE II method is used for ranking alternatives in multi-criteria decision problems. PROMETHEE V incorporates feasibility constraints, defined by lower (g) and upper (l) limits on the net outranking flow.

The prefuncs argument allows for different preference functions to be used for each criterion. The preference function transforms the difference between the evaluations of two alternatives on a given criterion into a preference degree.

The thr argument specifies the threshold parameters for the preference functions. The required parameters vary depending on the chosen preference function.

Value

A list containing the following components:

Phi_plus	A numeric vector of positive outranking flows (ϕ^+) for each alternative.
Phi_minus	A numeric vector of negative outranking flows (ϕ^-) for each alternative.
Phi_net	A numeric vector of net outranking flows (ϕ_{net}) for each alternative.
Feasibility	A logical vector indicating whether each alternative is feasible (TRUE) or not (FALSE) based on the g and l limits.
rank	A numeric vector of the final ranks of the alternatives, with lower ranks indicating better performance. Infeasible alternatives are ranked lower than feasible ones.

Author(s)

Cagatay Cebeci

References

Brans, J. P., & Mareschal, B. (1988). Geometrical representations for MCDA. *European Journal of Operational Research, 34*(1), 69-77. <doi:10.1016/0377-2217(88)90456-0>

Brans, J. P., Mareschal, B. (2005). Promethee Methods. In: *Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol 78.* Springer, New York, NY. <doi:10.1007/0-387-23081-5_5>

Behzadian, M., Kazemzadeh, R. B., Albadvi, A., & Aghdasi, M. (2010). PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research, 200*(1), 198-215. <doi:10.1016/j.ejor.2009.01.021>

Brans, J. P., De Smet, Y. (2016). PROMETHEE Methods. In: *Greco, S., Ehrgott, M., Figueira, J. (eds) Multiple Criteria Decision Analysis. International Series in Operations Research & Management Science, 233*. Springer, New York, NY. <doi:10.1007/978-1-4939-3094-4_6>

See Also

promethee1, promethee2, promethee3, promethee4, promethee6

Examples

```
# Decision matrix
dmat <- matrix(c(5, 150, 3, 200, 4, 180, 6, 170), nrow = 4, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4")
colnames(dmat) <- c("C1", "C2")

# Benefit-Cost vector
bc <- c(-1, 1)

# Weights
uw <- c(0.6, 0.4)

# Threshold values
tvals <- list(list(p = 50), list(p = 30))

# Parameters for PROMETHEE V
gval <- 0
lval <- 100

# Preference function
pf <- c("linear", "linear")

# Run PROMETHEE V function
respromethee5 <- promethee5(dmatrix=dmat, bcvec=bc, weights=uw,
     prefunc = pf, thr = tvals, g = gval, l = lval)
print(respromethee5)
```

promethee6 *PROMETHEE VI: Preference Ranking Organization METHod for Enrichment Evaluations (Method VI)*

Description

PROMETHEE VI method for multi-criteria decision analysis, ranks alternatives based on a balance between their central tendency and variability.

Usage

```
promethee6 (dmatrix, bcvec, weights, normethod = NULL, prefuncs = NULL, thr = NULL,
   varmethod = "abs_sum", p = NULL, q = NULL, tiesmethod="average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix. Rows represent alternatives, and columns represent criteria.
bcvec	A numeric vector indicating the benefit/cost orientation of each criterion. Use 1 for benefit criteria (to be maximized) and -1 for cost criteria (to be minimized).
weights	A vector of weights representing the relative importance of each criterion. Defaults to "equal". See calcweights for details.
normethod	An optional character string specifying the normalization method for the decision matrix because there is usually no need to normalize the decision matrix with PROMETHEE V. If not NULL, the calcnormal function will be used. Common options might include "linear", "vector", etc. Defaults to NULL, no normalization is performed. See calcnormal for details.
prefuncs	Optional. A character vector specifying the preference function for each criterion. Can be a single string (applied to all criteria) or a vector with length equal to the number of criteria. Available preference functions are: "linear", "usual", "quasi-criterion", "v-shape", and "level". If NULL, "linear" is used for all criteria.
thr	Optional. A list of threshold parameters for the preference functions. If a single list, it's applied to all criteria. If a list of lists, each element corresponds to a criterion. The structure of each element depends on the preference function: • "linear", "v-shape": A list with a single numeric value named p (preference threshold). • "quasi-criterion": A list with a single numeric value named s (indifference threshold). • "level": A list with two numeric values named p (preference threshold) and q (indifference threshold). If NULL, default thresholds are calculated based on the maximum absolute difference for each criterion.
p	A numeric value representing the preference parameter used in the calculation of Lambda. Must be non-negative.

q	A numeric value representing the indifference parameter used in the calculation of Lambda. Must be non-negative.
varmethod	Variability method. The "abs_sum" for absolute sum. The "range" is also available as an option.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The promethee6 function implements the PROMETHEE VI method, an extension of the PROMETHEE II method. PROMETHEE VI considers both the central tendency and the variability of the net outranking flows to rank alternatives.

The prefuncs argument allows for different preference functions to be used for each criterion. The preference function transforms the difference between the evaluations of two alternatives on a given criterion into a preference degree.

The thr argument specifies the parameters for the preference functions. The required parameters vary depending on the chosen preference function.

The arguments p and q are specific to PROMETHEE VI. They represent the preference and indifference parameters, respectively, used to calculate the Lambda values, which adjust the final ranking based on the variability of the net outranking flows.

Value

A list containing the following components:

Phi_plus	A numeric vector of positive outranking flows (ϕ^+) for each alternative.
Phi_minus	A numeric vector of negative outranking flows (ϕ^-) for each alternative.
Phi_net	A numeric vector of net outranking flows (ϕ_{net}) for each alternative.
Central_tendency	A numeric vector representing the central tendency of the net outranking flows (equal to Phi_net).
Variability	A numeric vector representing the variability of the outranking flows, calculated as the absolute sum of positive and negative flows.
Lambda	A numeric vector of Lambda values for each alternative, based on its variability and the p and q parameters.
Xi	A numeric vector of final values for each alternative, calculated by adjusting the central tendency with Lambda and variability.
rank	A numeric vector of the final ranks of the alternatives, with lower ranks indicating better performance, based on the Xi values.

Author(s)

Cagatay Cebeci

References

Brans, J. P., & Mareschal, B. (1988). Geometrical representations for MCDA. *European Journal of Operational Research, 34*(1), 69-77. <doi:10.1016/0377-2217(88)90456-0> Brans, J. P., Mareschal, B. (2005). Promethee Methods. In: *Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol 78.* Springer, New York, NY. <doi:10.1007/0-387-23081-5_5> Behzadian, M., Kazemzadeh, R. B., Albadvi, A., & Aghdasi, M. (2010). PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research, 200*(1), 198-215. <doi:10.1016/j.ejor.2009.01.021> Brans, J. P., De Smet, Y. (2016). PROMETHEE Methods. In: *Greco, S., Ehrgott, M., Figueira, J. (eds) Multiple Criteria Decision Analysis. International Series in Operations Research & Management Science, 233*. Springer, New York, NY. <doi:10.1007/978-1-4939-3094-4_6>

See Also

promethee1, promethee2, promethee3, promethee4, promethee5

Examples

```
# Decision matrix
dmat <- matrix(c(5, 150, 3, 200, 4, 180), nrow = 3, byrow = TRUE)
rownames(dmat) <- paste0("A", 1:3)
colnames(dmat) <- paste0("C", 1:2)

# Benefit-Cost vector
bc <- c(-1, 1)

# Predefined weights
uw <- c(0.5, 0.5)

# Threshold values
tvals <- list(list(p = 50), list(p = 30))

# Preference function
pf <- c("linear", "linear")

# PROMETHEE VI p and q parameters
pval <- 120
qval <- 20

# Run PROMETHEE VI function
respromethee6_1 <- promethee6(dmatrix=dmat, bcvec=bc, weights=uw,
     prefuncs = pf, thr = tvals, p = pval, q = qval)
print(respromethee6_1)

# Using different preference functions
pf2 <- c("v-shape", "level")
thr2 <- list(list(p = 60), list(q = 15, p = 40))
respromethee6_2 <- promethee6(dmatrix=dmat, bcvec=bc, weights=uw,
      prefuncs = pf2, thr = thr2, p = pval, q = qval)
print(respromethee6_2)
```

```
# Different weights and normalization
uw <- c(0.7, 0.3)
respromethee6_3 <- promethee6(dmatrix=dmat, bcvec=bc, weights=uw,
  normethod = "ratio", p = pval, q = qval)
print(respromethee6_3)
```

`ram` *RAM: Root Assessment Method*

Description

Implements the Root Assessment Method (RAM) to evaluate and rank alternatives based on normalized scores, weighted contributions, and root-based calculations.

Usage

```
ram(dmatrix, bcvec, weights, normethod = "sum", tiesmethod="average")
```

Arguments

`dmatrix`	Decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
`weights`	A vector of weights representing the relative importance of each criterion. See `calcweights` for details.
`normethod`	The normalization method to be applied. Defaults to `"sum"`. See `calcnormal` for details.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The RAM method evaluates alternatives by normalizing the decision matrix, applying weights to criteria, and calculating scores based on the root formula that integrates positive and negative contributions. The final ranking is derived from the computed scores (`ri`), which reflect the balance between benefit and cost criteria.

Value

A list containing the following components:

`nmatrix`	The normalized decision matrix.
`weighted_matrix`	The decision matrix after applying weights.
`Spi`	The summed positive contributions for each alternative.
`Smi`	The summed negative contributions for each alternative.
`ri`	The final root-based scores for each alternative.
`rank`	The ranks of final root-based scores for alternatives.

Author(s)

Cagatay Cebeci

References

Sotoudeh-Anvari, A. (2023). Root Assessment Method (RAM): A novel multi-criteria decision making method and its applications in sustainability challenges. *Journal of Cleaner Production, 423*, 138695.

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply RAM with user-defined weights
resram_1 <- ram(dmatrix=dmat, bcvec=bc, weights=uw)
print(resram_1)

# Apply RAM with equal weights
equw <- calcweights(dmat, bcvec=bc, type="equal")
resram_2 <- ram(dmatrix=dmat, bcvec=bc, weights=equw)
print(resram_2)
```

---

rankaggregate *Aggregat Ranks of Alternatives for Decision Making*

Description

Aggregates and summarizes the ranking information of alternatives obtained from multiple methods. Given a rank matrix where rows represent evaluation methods (e.g., MCDA methods) and columns represent alternatives, the function computes aggregated overall ranks, produces hierarchical outranking flow, and counts the frequency of each alternative appearing in the top `tk` ranks.

Usage

```
rankaggregate(rankmat, topk=3, damp=0.5, niters=100, tol=1e-4, tiesmethod="average")
```

Arguments

rankmat	A numeric matrix in which rows correspond to different ranking methods and columns correspond to alternatives. Each element rankmat[i, j] indicates the rank of alternative j based on method i, with lower values indicating better performance.
topk	A positive integer indicating the "top tk" positions. For each method, the function counts how many times each alternative appears within the top tk ranks. The default is 3.
damp	A numeric value in $(0, 1)$ used as the damping factor in the Markov chain aggregation method. The default is 0.5.
niters	A positive integer specifying the maximum number of iterations allowed in the Markov chain aggregation method. The default is 200.
tol	A numeric value specifying the convergence tolerance for the iterative procedure used in the Markov chain aggregation method. The default is 1e-4.
tiesmethod	A character string specifying the method used to handle ties when computing overall ranks. See rank for available options (e.g., "average", "first", "random", "max", "min"). The default is "average".

Details

The prefsummary function aggregates ranking information for a set of alternatives evaluated by multiple methods. In particular, it performs the following tasks:

1. Computes alternative rankings based on the Top-tk counts method.
2. Computes the overall ranks based on the sum of the ranks across methods using the Rank Sum approach.
3. Produces aggregated ranks using the Median of ranks method.
4. Computes pairwise comparisons between alternatives based on the Copeland method (assigning 1 point for a win and 0.5 points for a tie in pairwise comparisons).
5. Calculates overall scores using the Kemeny & Young method based on Kendall Tau distances.
6. Computes a stationary score and ranking using a Markov chain (PageRank-like) approach.
7. Combines the rankings from the above methods into a matrix and constructs a hierarchical (outranking) string (preference hierarchy) for each method.

Value

A list with the following components:

restopk	A list from the Top-tk counts method containing the top-tk threshold, counts for each alternative, and the corresponding ranking.
resranksum	A list containing the rank sums for each alternative and the ranking based on these sums.

`resmedian` A list containing the median rank for each alternative and its ranking.

`rescopeland` A list containing the Copeland scores for each alternative and the corresponding ranking.

`reskemeny` A list containing the Kemeny & Young scores for each alternative and the corresponding ranking.

`resmarkov` A list containing the Markov chain results: number of iterations, the wins matrix, the transition matrix, the Markov scores, and the resulting ranking.

`preference_ranking`
A matrix that aggregates the rankings from all methods. Row names correspond to methods while column names correspond to alternatives.

`preference_table`
A data frame presenting, for each method, a hierarchical outranking string (i.e., a preference hierarchy among alternatives).

Author(s)

Cagatay Cebeci

References

Saari, D. G., & Merlin, V. R. (1996). The Copeland method: I.: Relationships and the dictionary. *Economic Theory, 8*, 51-76. <doi:10.1007/BF01212012>

Examples

```
# Example rank matrix (rows: methods, columns: alternatives)
ranks <- c(
  5.0, 7.0, 6.0, 10.0, 8.0, 9.0, 3.0, 1.0, 4.0, 2.0,
  7.0, 8.0, 3.5, 3.5, 9.0, 3.5, 3.5, 3.5, 3.5, 10.0,
  5.0, 7.0, 6.0, 10.0, 9.0, 8.0, 3.0, 2.0, 4.0, 1.0,
  6.0, 5.0, 3.0, 9.0, 8.0, 10.0, 1.0, 2.0, 7.0, 4.0,
  5.5, 7.0, 4.0, 10.0, 9.0, 8.0, 3.0, 1.0, 5.5, 2.0,
  9.0, 7.0, 6.0, 8.0, 2.0, 10.0, 4.0, 5.0, 1.0, 3.0,
  10.0, 9.0, 6.0, 7.0, 4.0, 8.0, 5.0, 3.0, 2.0, 1.0,
  5.0, 7.0, 6.0, 10.0, 8.0, 9.0, 3.0, 1.0, 4.0, 2.0,
  5.0, 7.0, 6.0, 10.0, 8.0, 9.0, 3.0, 1.0, 4.0, 2.0,
  5.0, 7.0, 6.0, 10.0, 8.0, 9.0, 3.0, 1.0, 4.0, 2.0,
  2.0, 8.0, 3.0, 9.0, 5.0, 10.0, 1.0, 6.0, 4.0, 7.0,
  2.0, 8.0, 3.0, 9.0, 5.0, 10.0, 1.0, 6.0, 4.0, 7.0,
  9.0, 6.0, 7.0, 5.0, 3.0, 8.0, 4.0, 2.0, 2.0, 1.0,
  10.0, 6.0, 8.0, 7.0, 5.0, 9.0, 4.0, 3.0, 2.0, 1.0,
  10.0, 6.0, 7.0, 9.0, 5.0, 8.0, 4.0, 3.0, 2.0, 1.0,
  7.0, 10.0, 5.0, 9.0, 6.0, 8.0, 3.0, 1.0, 4.0, 2.0,
  6.0, 9.0, 5.0, 10.0, 7.0, 8.0, 3.0, 1.0, 4.0, 2.0,
  7.0, 10.0, 5.0, 9.0, 6.0, 8.0, 3.0, 1.0, 4.0, 2.0,
  3.0, 3.0, 10.0, 3.0, 3.0, 3.0, 8.0, 6.0, 9.0, 7.0,
  3.0, 9.0, 5.0, 10.0, 8.0, 7.0, 6.0, 2.0, 4.0, 1.0,
  10.0, 6.0, 8.0, 7.0, 5.0, 9.0, 4.0, 3.0, 2.0, 1.0,
  2.0, 8.0, 3.0, 9.0, 5.0, 10.0, 1.0, 6.0, 4.0, 7.0,
  2.0, 8.0, 3.0, 9.0, 5.0, 10.0, 1.0, 6.0, 4.0, 7.0,
```

```
  8.0, 6.0, 5.0, 10.0, 7.0, 9.0, 3.0, 1.0, 4.0, 2.0,
  9.0, 3.0, 7.0, 5.0, 10.0, 1.0, 8.0, 2.0, 6.0, 4.0,
  8.0, 9.0, 3.0, 4.0, 10.0, 6.0, 2.0, 1.0, 5.0, 7.0,
  8.5, 8.5, 3.0, 5.0, 8.5, 6.0, 2.0, 1.0, 4.0, 8.5,
  5.0, 7.0, 6.0, 10.0, 8.0, 9.0, 3.0, 1.0, 4.0, 2.0
)

rankmat <- matrix(ranks, nrow = 28, byrow = TRUE)
rownames(rankmat) <- c(
  "ARAS", "COCOSO", "CODAS", "EDAS", "ELECTRE4", "FUCA", "GRA", "MABAC",
  "MAIRCA", "MARCOS", "MAUT", "MAVT", "MEGAN", "MOORA", "PROMET1",
  "PROMET2", "PROMET3", "PROMET4", "PROMET5", "PROMET6", "RAM", "ROV",
  "SMART", "TOPSIS", "VIKOR", "WASPAS", "WPM", "WSM"
)
colnames(rankmat) <- paste0("ALT_", 1:ncol(rankmat))

# Aggregate the ranks
res_summary <- rankaggregate(rankmat, tiesmethod = "average",
   topk = 1, tol = 1e-4, niters = 10, damp = 0.4)
print(res_summary)
```

`rankcompare` *Comparison of Ranking Results*

Description

Computes and returns multiple ranking comparison measures, including Spearman correlation, Bootstrap tests based-on Jensen-Shannon divergences, Permutation test based-on Shannon-entropy differences, Wojciech Salabun similarity coefficient, and Wilcoxon rank sum test statistics.

Usage

```
rankcompare(rankmat, nperms = 1000, nboot = 1000,
    entropyopt = "jsd", j = 1, k = 3,
    alpha = 0.05, padjmethod = "fdr",
    biplot = TRUE)
```

Arguments

`rankmat`	A matrix containing ranking data where columns represent different ranking scenarios.
`nperms`	The number of permutations for entropy-based comparison (default: 1000).
`nboot`	The number of resampling for entropy-based comparison using bootstrap (default: 1000).
`entropyopt`	Entropy method: "shannon" for Shannon entropy or "jsd" for Jensen-Shannon divergence (default: "jsd").
`j`	For modified index of agreement, an integer specifying the exponent for the absolute difference, typically set to 1.

k	For rank similarity for a ranking range from k1 to k2. Usually set to 3 for top 3 rank similarity.
alpha	Significance threshold for statistical test such as Kruskal-Wallis test (default: 0.05).
padjmethod	P-value adjustment method for Kruskal-Wallis test (default: "fdr").
biplot	Logical whether Geladi's PCA-bilot is displayed or not.

Value

A list with five elements:

src	A matrix of Spearman correlation comparisons
entperm	A matrix of entropy-based comparisons with permutation tests
entboot	A matrix of entropy-based comparisons with bootstrap
wsrs	A matrix of Wojciech Salabun similarity comparisons
mia	A matrix of modified index of agreement comparisons
wilcox	A matrix of Wilcoxon Rank Sum test comparisons
rangesim	A matrix of rank range similarities
geladipca	Geladi's PCA analysis results and PCA biplot.

Author(s)

Cagatay Cebeci

See Also

sensana, rankentboot, rankentperm, rankmia, rankwilcox, rankwssim, rankspearman, rankrangesim

Examples

```
rankmat <- matrix(sample(1:10, 50, replace=TRUE), ncol=5)
rownames(rankmat) <- paste0("A", 1:10)
colnames(rankmat) <- paste0("M", 1:5)

result <- rankcompare(rankmat, entropyopt="jsd", nboot=100, nperms=100,
   padjmethod="bonferroni", k=3, j=1)
print(result)
```

rankentboot *Bootstrap-Based Rank Entropy Comparison*

Description

Computes pairwise entropy differences for ranking matrices using Shannon entropy or Jensen-Shannon divergence. The function applies a bootstrap-based statistical test and adjusts p-values using various correction methods.

Usage

```
rankentboot(rankmat, nboot = 1000, entropyopt = "shannon", padjmethod = "none")
```

Arguments

rankmat	A ranking matrix where rows represent methods and columns represent ranking positions.
nboot	The number of bootstrap resampling iterations. Defaults to 1000.
entropyopt	The entropy calculation method. Options: "shannon" Uses Shannon entropy. "jsd" Uses Jensen-Shannon divergence.
padjmethod	The p-value adjustment method. Options include: "holm" Holm correction (step-down method). "hochberg" Hochberg correction (step-up method). "hommel" Hommel correction (enhanced Holm method). "bonferroni" Bonferroni correction (dividing p-values by the number of tests). "BH" Benjamini-Hochberg procedure (controls False Discovery Rate). "BY" Benjamini-Yekutieli procedure (more conservative FDR control). "fdr" False Discovery Rate correction (alias for BH method). "none" No adjustment applied (raw bootstrap p-values).

Details

This function computes entropy-based differences between ranking methods and uses bootstrap sampling to generate p-values. P-values can be adjusted using multiple correction methods. The output matrix contains:

- Lower triangle: Similarity scores (inverse entropy differences).
- Upper triangle: Bootstrap p-values (adjusted if selected).
- Diagonal: Empty values.

Value

A list containing:

`entropy`	Selected entropy method ("shannon" or "jsd").
`p_adjustment`	Selected p-value adjustment method.
`nboot`	Number of bootstrap iterations performed.
`displaymat`	Final comparison matrix with similarity scores and adjusted p-values.

Author(s)

Cagatay Cebeci

See Also

`p.adjust`, `rankcompare`, `ranksrd`, `rankentperm`, `rankmia`, `rankwilcox`, `rankwssim`, `rankspearman`, `rankrangesim`

Examples

```
# Rank matrix
rankmat <- as.matrix(read.table(text = "
                     G1   G2 G3 G4   G5 G6 G7   G8   G9 G10  G11  G12
ARAS              3 12.0  1  8  4.0  7  6 10.0  2.0   5 11.0  9.0
EDAS              3 12.0  2  8  4.0  9  6 10.0  1.0   5 11.0  7.0
ELECTRE4          3 11.5  4  8  1.5  7  5  9.5  1.5   6 11.5  9.5
FUCA              4  3.0 10  7  2.0  6  1  9.0  5.0  11  8.0 12.0
GRA               6  3.0 10  5  2.0  7  1  9.0  4.0  11  8.0 12.0
MABAC             3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
MAIRCA           10  1.0  9  5 12.0  6  7  4.0 11.0   8  3.0  2.0
MARCOS            3 12.0  1  8  4.0  7  6 10.0  2.0   5 11.0  9.0
MEGAN             5  3.0 11  7  1.0  6  2  9.0  4.0  10  8.0 12.0
MOORA             7  1.0 11  9  2.0  4  3  8.0  6.0  10  5.0 12.0
PROMET2           3  9.0  6  8  1.0  5  4 10.0  2.0   7 11.0 12.0
RAM               7  1.0 11  9  2.0  4  3  8.0  6.0  10  5.0 12.0
ROV               3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
SMART             5  3.0 11  7  2.0  6  1  9.0  4.0  10  8.0 12.0
TOPSIS            2 12.0  4  8  1.0  7  6  9.0  3.0   5 11.0 10.0
VIKOR             1 12.0  4  8  2.0  5  7 10.0  3.0   6  9.0 11.0
WASPAS            2 12.0  1  8  4.0  7  6 10.0  3.0   5 11.0  9.0
", header = TRUE, row.names = 1))

print(rankmat)

res_shannon_adj_none <- rankentboot(rankmat, nboot = 100,
   entropyopt = "shannon", padjmethod="none")
print(res_shannon_adj_none)

res_shannon_adj_bonf <- rankentboot(rankmat, nboot = 100,
   entropyopt = "shannon", padjmethod="bonferroni")
print(res_shannon_adj_bonf)
```

```
res_jsd_adj_fdr <- rankentboot(rankmat, nboot = 100,
   entropyopt = "jsd", padjmethod="fdr")
print(res_jsd_adj_fdr)

res_jsd_adj_holm <- rankentboot(rankmat, nboot = 100,
   entropyopt = "jsd", padjmethod="holm")
print(res_jsd_adj_holm)
```

`rankentperm` *Entropy-Based Pairwise Rank Comparison*

Description

Compares pairwise rankings using either Shannon Entropy or Jensen-Shannon Divergence (JSD), with permutation testing for statistical significance.

Usage

```
rankentperm(rankmat, nperms = 1000, entropyopt = "shannon", padjmethod="none")
```

Arguments

`rankmat`	A matrix representing ranked alternatives for multiple scenarios. Rows correspond to alternatives, and columns to scenarios.
`nperms`	An integer specifying the number of permutations for statistical testing. Default is `1000`.
`entropyopt`	A character string specifying the entropy method: `"shannon"` for Shannon Entropy or `"jsd"` for Jensen-Shannon Divergence. Default is `"shannon"`.
`padjmethod`	The p-value adjustment method. Options include: `"holm"` Holm correction (step-down method). `"hochberg"` Hochberg correction (step-up method). `"hommel"` Hommel correction (enhanced Holm method). `"bonferroni"` Bonferroni correction (dividing p-values by the number of tests). `"BH"` Benjamini-Hochberg procedure (controls False Discovery Rate). `"BY"` Benjamini-Yekutieli procedure (more conservative FDR control). `"fdr"` False Discovery Rate correction (alias for BH method). `"none"` No adjustment applied (raw bootstrap p-values).

Details

This function analyzes ranking stability by computing entropy-based differences between rank distributions across weighting scenarios or MCDA methods. The function allows the choice of Shannon Entropy or Jensen-Shannon Divergence, offering different sensitivity levels. Shannon Entropy was introduced the concept of entropy in Shannon's paper, 'A Mathematical Theory of Communication' in 1949, laying the foundation for information theory. The Jensen-Shannon Divergence (JSD) is based on the work of Johan Jensen and Claude Shannon. It is a symmetric version of the

Kullback-Leibler Divergence: The KL Divergence is an asymmetric metric $D(P||Q) \neq D(Q||P)$ The JSD was designed to address this asymmetry issue, making it a symmetric measure. A permutation test evaluates the statistical significance of obtained entropy based ranking differences.

Value

A list containing:

`nperms`	The number of permutations used in the analysis.
`entropyopt`	The entropy method used (`"shannon"` or `"jsd"`).
`entropy_matrix`	A matrix storing entropy-based differences between scenarios.
`pval_matrix`	A matrix containing p-values from permutation tests, indicating whether ranking distributions are significantly different.

Note

Please note that JSD can provide a slightly more sensitive difference assessment compared to Shannon Entropy.

Author(s)

Cagatay Cebeci

References

Shannon, C.E., Weaver, W. (1949). The Mathematical Theory of Communication, Univ of Illinois Press. <ISBN:0-252-72548-4>

Maasoumi, E., & Racine, J. (2002). Entropy and predictability of stock market returns. *Journal of Econometrics, 107*(1-2), 291-312.

See Also

rankcompare, rankentboot, rankmia, rankwilcox, rankwssim, rankspearman, rankrangesim

Examples

```
# Example rank matrix
rankmat <- matrix(c(
  1, 1, 1, 1, 2, 2, 1, 1, 2, 2,
  2, 2, 2, 3, 3, 1, 2, 2, 3, 3,
  3, 3, 3, 2, 1, 3, 3, 3, 1, 1,
  4, 4, 4, 3, 4, 4, 4, 4, 4, 4,
  5, 5, 5, 5, 5, 5, 5, 5, 5, 5
), nrow = 5, byrow = TRUE)

rownames(rankmat) <- c("OptA", "OptB", "OptC", "OptD", "OptE")
colnames(rankmat) <- paste0("SW_", 1:10)

# Shannon Entropy method
res_shannon <- rankentperm(rankmat, nperms=100, entropyopt = "shannon", padjmethod="fdr")
```

```
print(res_shannon$displaymat)
print(res_shannon$pval_matrix)

# Jensen-Shannon Divergence method
res_jsd <- rankentperm(rankmat, nperms=100, entropyopt = "jsd", padjmethod="bonferroni")
print(res_jsd$displaymat)
print(res_jsd$pval_matrix)
```

---

rankgamma *Compute Goodman & Kruskal's Gamma Statistic*

Description

Computes the Goodman & Kruskal's Gamma statistic between all pairs of rankings in a given matrix.

Usage

```
rankgamma(rankmat)
```

Arguments

rankmat	A numeric matrix where rows represent different ranking entities and columns represent ranking criteria.

Value

A list with the following components:

rankmat	The input ranking matrix.
gammamat	A matrix of Goodman & Kruskal's Gamma statistics computed between each pair of rows in the ranking matrix.

Author(s)

Cagatay Cebeci

Examples

```
# Rank matrix based on MCDA methods
rankmat <- matrix(c(
    3, 12.0, 1, 8, 4.0, 7, 6, 10.0, 2.0, 5, 11.0, 9.0,
    3, 12.0, 2, 8, 4.0, 9, 6, 10.0, 1.0, 5, 11.0, 7.0,
    3, 11.5, 4, 8, 1.5, 7, 5, 9.5, 1.5, 6, 11.5, 9.5,
    4, 3.0, 10, 7, 2.0, 6, 1, 9.0, 5.0, 11, 8.0, 12.0,
    6, 3.0, 10, 5, 2.0, 7, 1, 9.0, 4.0, 11, 8.0, 12.0,
    3, 12.0, 4, 8, 1.0, 7, 6, 9.0, 2.0, 5, 10.0, 11.0,
    10, 1.0, 9, 5, 12.0, 6, 7, 4.0, 11.0, 8, 3.0, 2.0,
    3, 12.0, 1, 8, 4.0, 7, 6, 10.0, 2.0, 5, 11.0, 9.0,
```

```
    3, 12.0, 4, 8, 1.0, 7, 6, 9.0, 2.0, 5, 10.0, 11.0,
    5, 3.0, 11, 7, 1.0, 6, 2, 9.0, 4.0, 10, 8.0, 12.0,
    7, 1.0, 11, 9, 2.0, 4, 3, 8.0, 6.0, 10, 5.0, 12.0,
    4, 10.0, 6, 8, 1.0, 5, 3, 9.0, 2.0, 7, 11.0, 12.0,
    3, 9.0, 6, 8, 1.0, 5, 4, 10.0, 2.0, 7, 11.0, 12.0,
    4, 10.0, 6, 8, 1.0, 5, 3, 9.0, 2.0, 7, 11.0, 12.0,
    7, 1.0, 11, 9, 2.0, 4, 3, 8.0, 6.0, 10, 5.0, 12.0,
    3, 12.0, 4, 8, 1.0, 7, 6, 9.0, 2.0, 5, 10.0, 11.0,
    5, 3.0, 11, 7, 2.0, 6, 1, 9.0, 4.0, 10, 8.0, 12.0,
    2, 12.0, 4, 8, 1.0, 7, 6, 9.0, 3.0, 5, 11.0, 10.0,
    1, 12.0, 4, 8, 2.0, 5, 7, 10.0, 3.0, 6, 9.0, 11.0,
    2, 12.0, 1, 8, 4.0, 7, 6, 10.0, 3.0, 5, 11.0, 9.0,
    2, 12.0, 1, 7, 4.0, 8, 6, 10.0, 3.0, 5, 11.0, 9.0,
    4, 7.0, 9, 5, 1.0, 6, 2, 8.0, 3.0, 11, 10.0, 12.0
), nrow=22, byrow=TRUE) # Changed nrow to 22 as there are 22 rows of data

rownames(rankmat) <- c("ARAS", "EDAS", "ELECTRE4", "FUCA", "GRA", "MABAC", "MAIRCA",
    "MARCOS", "MAUT", "MEGAN", "MOORA", "PROMET1", "PROMET2", "PROMET6", "RAM", "ROV",
    "SMART", "TOPSIS", "VIKOR", "WASPAS", "WPM", "WSM")
colnames(rankmat) <- c("G1", "G2", "G3", "G4", "G5",
    "G6", "G7", "G8", "G9", "G10", "G11", "G12")

print(rankmat)

result <- rankgamma(rankmat)
print(round(result$gammamat, 2))
gammamat <- round(result$gammamat, 2)

corplot(gammamat, labsize = 10, fsize = 3, fcolor = "black",
  colpal = c("red", "white", "dodgerblue"), title = "Goodman- Kruskall's Gamma Statistics",
   xlab = "Methods", ylab = "Methods", shape = "square", displabels = TRUE)
```

`rankheatmap` *Rank Heatmap Plot*

Description

Draws a heatmap of the overall ranks of n number of alternatives across p metrics.

Usage

```
rankheatmap(rankmat, colpal = NULL, yl = NULL, xl = NULL, htitle = NULL,
  tcol = "white", dendro = "row", cellnotes = FALSE)
```

Arguments

`rankmat`	A data frame or numeric matrix containing performance ranks of the alternatives (models, methods, etc.). The alternatives are located in the columns of the data frame.

colpal	A color palette to be used in drawing the heatmap, or an integer between 1 and 6 for built-in palettes.
xl	A character string describing the x-axis label.
yl	A character string describing the y-axis label.
htitle	A character string describing the main title of the heatmap.
tcol	A character string describing the color of text in the cells of the heatmap.
dendro	A character string that stands for row and column dendrograms. Use 'row', 'column', or 'both'.
cellnotes	A logical flag indicating whether the cell notes are displayed or not. Default value is FALSE. Avoid using it for large matrices.

Details

A heatmap is a type of graphic where the values of elements in a matrix are represented by colors. It visualizes a data matrix with two dimensions, applying gradual colors to indicate different values of the items. Typically, darker colors signal higher values while lighter colors denote lower values. Heatmaps are popular in exploratory data analysis and data science applications. It's essential to decide on proper color encoding and meaningful reordering of rows and columns to yield informative heatmaps (Gehlenborg & Wong, 2012).

Heatmaps are often used to identify patterns, correlations, and outliers in datasets at a glance. They find applications in fields like finance, geography, data analytics, and genomics. Additionally, they can be extremely useful for model evaluation, letting users quickly gauge the performance of alternative models based on multiple criteria.

Value

No value returned.

Author(s)

Cagatay Cebeci

References

Gehlenborg, N., & Wong, B. (2012). Heat maps. *Nature Methods, 9*(3), 213.

See Also

boxplotmcda, rankcompare

Examples

```
# Example rank matrix
alternatives <- paste0("Alt", LETTERS[1:8])
criteria <- paste0("C", 1:15)

set.seed(24)
rankmat <- data.frame(Method = alternatives)
```

```
for (criterion in criteria) {
  rankmat[[criterion]] <- sample(1:8, 8, replace = FALSE)
}
rownames(rankmat) <- rankmat[,1]
rankmat <- as.matrix(rankmat[,-1])
print(rankmat)
rankheatmap(rankmat)

# Heatmap with color palette number 1 and row dendrogram only
rankheatmap(rankmat, colpal = 1, dendro = "row", cellnotes = TRUE)

# Heatmap with user-defined color palette, column dendrogram, and cell notes
colpalcustom <- colorRampPalette(c("orange", "green", "blue"))
rankheatmap(rankmat, colpal = colpalcustom, dendro = "row", cellnotes = TRUE)
```

`rankmia` *Modified Index of Agreement*

Description

Calculates the Modified Index of Agreement (MIA) between multiple rankings by MCDA methods.

Usage

```
rankmia(rankmat, j = 1, na.rm = TRUE)
```

Arguments

`rankmat`	A numeric matrix where each row represents a different ranking and each column represents an item being ranked. The values in the matrix are the ranks assigned to each item.
`j`	An integer specifying the exponent for the absolute difference, typically set to 1.
`na.rm`	A logical value indicating whether missing values (`NA`) should be removed before calculation. Defaults to `TRUE`.

Details

The Modified Index of Agreement (MOIA) is an improved version of the Index of Agreement (IA), originally developed by Willmott (1981). It was designed as a standardized measure to evaluate the accuracy of model predictions compared to observed data. However, limitations led to modifications that enhance its sensitivity and reliability.

This function extends the application of MOIA to the comparison of rank orderings, useful in Multi-Criteria Decision Analysis (MCDA) problems. The Index of Agreement and Modified Index of Agreement values range between 0 and 1, where 1 indicates perfect agreement and 0 indicates complete disagreement.

MIA is calculated using the following formula:

$$MOIA = 1 - \frac{\sum_{i=1}^{N}(|S_i - \bar{O}|^j + |O_i - \bar{O}|^j)}{\sum_{i=1}^{N}|O_i - S_i|^j}$$

where:

MIA The Modified Index of Agreement value (ranging from 0 to 1).

N The total number of items being ranked.

S_i The rank assigned by one MCDA method.

O_i The rank assigned by another MCDA method.

$\bar{O}$ The mean rank value, calculated as $(N + 1)/2$.

j Exponent used for absolute differences (typically $j = 1$).

$|\cdot|$ The absolute value function.

$\sum$ Summation over all ranked items.

Higher values indicate greater agreement between rankings. This formula allows quantification of ranking similarity across MCDA methods.

Value

A data frame where each row and column represents a ranking from the input `rankmat`. The values in the data frame are the Modified Index of Agreement scores between the corresponding pairs of rankings, formatted to two decimal places.

Author(s)

Cagatay Cebeci

See Also

`rankcompare`, `rankentboot`,`rankentperm`, `rankwilcox`, `rankwssim`, `rankspearman`, `rankrangesim`

Examples

```
# Rank matrix based on MCDA methods
rankmat <- matrix(c(
    3, 12.0, 1, 8, 4.0, 7, 6, 10.0, 2.0, 5, 11.0, 9.0,
    3, 12.0, 2, 8, 4.0, 9, 6, 10.0, 1.0, 5, 11.0, 7.0,
    3, 11.5, 4, 8, 1.5, 7, 5, 9.5, 1.5, 6, 11.5, 9.5,
    4, 3.0, 10, 7, 2.0, 6, 1, 9.0, 5.0, 11, 8.0, 12.0,
    6, 3.0, 10, 5, 2.0, 7, 1, 9.0, 4.0, 11, 8.0, 12.0,
    3, 12.0, 4, 8, 1.0, 7, 6, 9.0, 2.0, 5, 10.0, 11.0,
    10, 1.0, 9, 5, 12.0, 6, 7, 4.0, 11.0, 8, 3.0, 2.0,
    3, 12.0, 1, 8, 4.0, 7, 6, 10.0, 2.0, 5, 11.0, 9.0,
    3, 12.0, 4, 8, 1.0, 7, 6, 9.0, 2.0, 5, 10.0, 11.0,
    5, 3.0, 11, 7, 1.0, 6, 2, 9.0, 4.0, 10, 8.0, 12.0,
    7, 1.0, 11, 9, 2.0, 4, 3, 8.0, 6.0, 10, 5.0, 12.0,
    4, 10.0, 6, 8, 1.0, 5, 3, 9.0, 2.0, 7, 11.0, 12.0,
    3, 9.0, 6, 8, 1.0, 5, 4, 10.0, 2.0, 7, 11.0, 12.0,
```

```
    4, 10.0, 6, 8, 1.0, 5, 3, 9.0, 2.0, 7, 11.0, 12.0,
    7, 1.0, 11, 9, 2.0, 4, 3, 8.0, 6.0, 10, 5.0, 12.0,
    3, 12.0, 4, 8, 1.0, 7, 6, 9.0, 2.0, 5, 10.0, 11.0,
    5, 3.0, 11, 7, 2.0, 6, 1, 9.0, 4.0, 10, 8.0, 12.0,
    2, 12.0, 4, 8, 1.0, 7, 6, 9.0, 3.0, 5, 11.0, 10.0,
    1, 12.0, 4, 8, 2.0, 5, 7, 10.0, 3.0, 6, 9.0, 11.0,
    2, 12.0, 1, 8, 4.0, 7, 6, 10.0, 3.0, 5, 11.0, 9.0,
    2, 12.0, 1, 7, 4.0, 8, 6, 10.0, 3.0, 5, 11.0, 9.0,
    4, 7.0, 9, 5, 1.0, 6, 2, 8.0, 3.0, 11, 10.0, 12.0
), nrow=22, byrow=TRUE) # Changed nrow to 22 as there are 22 rows of data

rownames(rankmat) <- c("ARAS", "EDAS", "ELECTRE4", "FUCA", "GRA", "MABAC", "MAIRCA",
    "MARCOS", "MAUT", "MEGAN", "MOORA", "PROMET1", "PROMET2", "PROMET6", "RAM", "ROV",
    "SMART", "TOPSIS", "VIKOR", "WASPAS", "WPM", "WSM")
colnames(rankmat) <- c("G1", "G2", "G3", "G4", "G5",
    "G6", "G7", "G8", "G9", "G10", "G11", "G12")

print(rankmat)

# Calculate the Modified Index of Agreement matrix
resmia <- rankmia(rankmat)
print(resmia)
```

---

rankpca *Principal Component Analysis for Ranking Data with Geladi's Approach*

Description

Performs Principal Component Analysis (PCA) on a matrix of ranks, where rows represent alternatives and columns represent criteria. This function adapts Geladi's approach to interpreting PCA results in the context of multi-criteria performance, focusing on overall performance (PC1) and uniformity of behavior (PC2) of alternatives.

Usage

```
rankpca(rankmat, biplot = TRUE, reverse = FALSE)
```

Arguments

rankmat	A matrix or data.frame where rows represent alternatives and columns represent criteria. Each cell contains the rank of an alternative for a specific criterion (e.g., 1 for best, 8 for worst). **Row names must be provided** to identify alternatives.
biplot	If TRUE (default), a biplot showing the PCA scores for alternatives and loadings for criteria will be generated using the factoextra package.
reverse	If FALSE (default), the matrix is processed as original, but if TRUE the values in rows are reversed by sorting in opposite direction

Details

This function applies PCA to ranking data, offering insights into the underlying patterns and relationships among alternatives based on their performance across multiple criteria. It extends Geladi's concept of using PCA to analyze "regression vectors" to ranking vectors.

The interpretation of the first two principal components (PC1 and PC2) follows Geladi's framework:

- **PC1 (Overall Performance):** This component typically captures the largest amount of variance and differentiates alternatives based on their overall performance. Alternatives clustering at one end of PC1 generally perform better or worse across most criteria. The exact interpretation (e.g., positive PC1 score = better/worse) depends on the signs of the criteria loadings.
- **PC2 (Uniformity of Behavior):** This component represents the consistency of an alternative's performance across different criteria. Alternatives with PC2 scores close to zero exhibit uniform behavior, meaning their ranking profile is consistent. Alternatives with PC2 scores significantly deviating from zero show non-uniform behavior, indicating their performance varies considerably depending on the criterion.

Note: PCA on ranking data is primarily a **discovery and understanding tool**, not a direct ranking method. It helps identify clusters, outliers, and key driving criteria, complementing explicit ranking methods like SRD.

Value

A `list` containing the following components:

`pca_results` The raw `prcomp` object returned by the PCA analysis.

`alternative_scores`
: A `data.frame` of principal component scores for each alternative (rows are alternatives, columns are PCs).

`criteria_loadings`
: A `data.frame` of loadings for each criterion on the principal components (rows are criteria, columns are PCs).

`explained_variance`
: A numeric vector indicating the proportion of variance explained by each principal component.

`cumulative_variance`
: A numeric vector indicating the cumulative proportion of variance explained by the principal components.

`pc1_interpretation`
: A character string providing an interpretation of PC1's meaning based on the data's characteristics.

`pc2_interpretation`
: A character string providing an interpretation of PC2's meaning.

References

Geladi, P. (1995). The regression model comparison plot (REMOCOP), Proc. Spectroscopy Across the Spectrum IV, Norwich, UK, 11–14 July 1994, in: D. Andrews, A. Davies (Editors), Frontiers inAnalytical Spectroscopy, The Royal Society of Chemistry, Cambridge, UK, 1995, pp. 225–236.

Héberger, K. (2010). Sum of Ranking Differences (SRD) for method comparison and its relationship to the principle of parsimony. *Chemometrics and Intelligent Laboratory Systems*, 101(1), 1-8.

See Also

prcomp, fviz_pca_biplot, ranksrd

Examples

```
# Sample rank matrix (8 alternatives, 15 criteria)
rankmat <- data.frame(
  C1 = c(7, 3, 2, 8, 1, 6, 5, 4), C2 = c(2, 1, 4, 7, 6, 3, 8, 5),
  C3 = c(5, 8, 1, 7, 4, 6, 2, 3), C4 = c(8, 2, 1, 4, 3, 5, 6, 7),
  C5 = c(1, 5, 4, 6, 2, 3, 7, 8), C6 = c(6, 8, 2, 3, 1, 4, 5, 7),
  C7 = c(6, 7, 1, 8, 2, 3, 5, 4), C8 = c(6, 3, 4, 5, 2, 1, 7, 8),
  C9 = c(1, 8, 3, 4, 6, 5, 7, 2), C10 = c(7, 8, 5, 2, 4, 6, 3, 1),
  C11 = c(6, 2, 3, 8, 5, 7, 4, 1), C12 = c(6, 8, 3, 7, 4, 1, 5, 2),
  C13 = c(2, 3, 7, 5, 6, 1, 8, 4), C14 = c(2, 1, 4, 5, 3, 8, 6, 7),
  C15 = c(1, 6, 4, 5, 8, 7, 3, 2),
  row.names = c("AltA", "AltB", "AltC", "AltD", "AltE", "AltF", "AltG", "AltH")
)

# Perform Geladi PCA and show the biplot
respca <- rankpca(rankmat, biplot = TRUE, reverse=TRUE)

# Geladi PCA Analysis Results
print(paste("Proportion of Explained Variance:", paste(round(respca$expvar, 3), collapse = ", ")))

# Alternative Scores (PC1 and PC2)
print(round(respca$ascores[, 1:2], 2))

# Criteria Loadings (PC1 and PC2)
print(round(respca$critloads[, 1:2], 2))

# PC1 Interpretation
print(respca$intprtpc1)

# PC2 Interpretation
print(respca$intprtpc2)
```

---

rankrangesim *Calculate Similarity of Ranked Alternatives in a Given Range*

Description

Calculates the similarity between ranking methods by selecting alternatives from a given range and comparing them across different methods. Additionally, it provides an option to visualize the similarity matrix as a heatmap.

Usage

```
rankrangesim(mat, k = 3, plotsim = FALSE,
   low_color = "blue", high_color = "red", text_color = "white")
```

Arguments

mat	A matrix containing ranked alternatives with methods as row names and alternatives as column names.
k	A numeric value or a vector of length 2 specifying the range of rankings to compare. If a single number is provided, it selects from the beginning or end of the rankings.
plotsim	Logical. If TRUE, generates a heatmap showing the similarity percentages across methods.
low_color	Color for low similarity values in the heatmap. Default is "blue".
high_color	Color for high similarity values in the heatmap. Default is "red".
text_color	Color for the text labels inside the heatmap cells. Default is "white".

Details

This function takes a ranking matrix and extracts the top or bottom alternatives for each ranking method within the specified range. It then calculates how often methods select the same alternatives, forming a similarity matrix.

If `plotsim = TRUE`, the function generates a heatmap using ggplot2 to visually display the similarity percentages.

Value

A list containing:

selected_k	A list of selected alternatives for each ranking method.
similarity	A matrix showing the count of common selected alternatives between methods.
percentage	A matrix representing similarity percentages of selected alternatives between methods.

See Also

rankcompare, rankentboot,rankentperm, rankmia, rankwilcox, rankwssim, rankspearman

Examples

```
# Example ranking matrix
rankmat <- matrix(c(
   3, 12, 1, 8, 4, 7, 6, 10, 2, 5, 11, 9,
   3, 12, 2, 8, 4, 9, 6, 10, 1, 5, 11, 7,
   3, 11.5, 4, 8, 1.5, 7, 5, 9.5, 1.5, 6, 11.5, 9.5,
   4, 3, 10, 7, 2, 6, 1, 9, 5, 11, 8, 12,
   6, 3, 10, 5, 2, 7, 1, 9, 4, 11, 8, 12,
   3, 12, 4, 8, 1, 7, 6, 9, 2, 5, 10, 11,
```

```
   3, 12, 1, 8, 4, 7, 6, 10, 2, 5, 11, 9,
   5, 3, 11, 7, 1, 6, 2, 9, 4, 10, 8, 12,
   7, 1, 11, 9, 2, 4, 3, 8, 6, 10, 5, 12,
   3, 9, 6, 8, 1, 5, 4, 10, 2, 7, 11, 12
), nrow = 10, byrow = TRUE)

rownames(rankmat) <- c("ARAS", "EDAS", "ELECT4", "FUCA", "GRA",
   "MABAC", "MARCOS", "MEGAN", "MOORA", "PROMT2")
colnames(rankmat) <- paste0("G", 1:12)

# Similarity for the best and worst k
k1 <- 3 # Best 3 alternatives
k2 <- -3 # Worst 3 alternatives

res_k1 <- rankrangesim(rankmat, k=k1, plotsim = TRUE)  # Best 3
res_k2 <- rankrangesim(rankmat, k=k2, plotsim = FALSE) # Worst 3

print(res_k1$selected_k)
print(res_k1$similarity)
print(res_k1$percentage)

print(res_k2$selected_k)
print(res_k2$similarity)
print(res_k2$percentage)

# Similarity for a specific range

kr <- c(5, 8)
res_kr <- rankrangesim(rankmat, k=kr, plotsim = TRUE,
   low_color = "dodgerblue", high_color = "orange", text_color = "gray")

print(res_kr$selected_k)
print(res_kr$similarity)
print(res_kr$percentage)
```

`rankspearman` *Spearman's Rank Correlation Matrix*

Description

Calculates Spearman's rank correlation matrix for a ranking matrix.

Usage

```
rankspearman(rankmat)
```

Arguments

`rankmat`	A numerical matrix containing ranking values. The function automatically transposes this matrix before calculating correlations, meaning each column should represent a criterion/variable.

Details

In the resulting matrix, statistical significance stars appear above the diagonal, while correlation coefficients appear below the diagonal. Elements on the diagonal show the correlation coefficient (usually 1.00). The function calculates Spearman's rank correlation and associated p-values for each pair of variables using `cor.test(method = "spearman")`. Significance stars are assigned based on p-values:

*** $p < 0.001$

** $p < 0.01$

* $p < 0.05$

Blank $p >= 0.05$ (not significant)

The stars appear in cells above the diagonal show the signifiance of correlations. Cells below the diagonal contain rounded correlation coefficients. Diagonal cells show a variable's correlation with itself (1.00).

Value

Returns a list:

`cormat`	A formatted character matrix displaying correlation coefficients (below the diagonal) and statistical significance stars (above the diagonal). Diagonal elements are displayed as 1.00.
`p_values`	A numerical matrix containing calculated p-values for each variable pair. Diagonal elements will be `NA` (Not Available).

Note

If ties exist in the ranking matrix, the `cor.test` function may be unable to compute exact p-values and will instead use approximate values. This usually does not affect the validity of the analysis but is important to note.

Author(s)

Cagatay Cebeci

See Also

`rankcompare`, `rankentboot`,`rankentperm`, `rankmia`, `rankwilcox`, `rankwssim`, `rankrangesim`

Examples

```
# Sample ranking matrix
rankmat <- as.matrix(read.table(text = "
         G1   G2 G3 G4   G5 G6 G7   G8   G9 G10  G11  G12
ARAS      3 12.0  1  8  4.0  7  6 10.0  2.0   5 11.0  9.0
EDAS      3 12.0  2  8  4.0  9  6 10.0  1.0   5 11.0  7.0
ELECTRE4  3 11.5  4  8  1.5  7  5  9.5  1.5   6 11.5  9.5
FUCA      4  3.0 10  7  2.0  6  1  9.0  5.0  11  8.0 12.0
```

```
GRA        6  3.0 10  5  2.0  7  1  9.0  4.0  11  8.0 12.0
MABAC      3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
MAIRCA    10  1.0  9  5 12.0  6  7  4.0 11.0   8  3.0  2.0
MARCOS     3 12.0  1  8  4.0  7  6 10.0  2.0   5 11.0  9.0
MAUT       3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
MEGAN      5  3.0 11  7  1.0  6  2  9.0  4.0  10  8.0 12.0
MOORA      7  1.0 11  9  2.0  4  3  8.0  6.0  10  5.0 12.0
PROMET1    4 10.0  6  8  1.0  5  3  9.0  2.0   7 11.0 12.0
PROMET2    3  9.0  6  8  1.0  5  4 10.0  2.0   7 11.0 12.0
PROMET6    4 10.0  6  8  1.0  5  3  9.0  2.0   7 11.0 12.0
RAM        7  1.0 11  9  2.0  4  3  8.0  6.0  10  5.0 12.0
ROV        3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
SMART      5  3.0 11  7  2.0  6  1  9.0  4.0  10  8.0 12.0
TOPSIS     2 12.0  4  8  1.0  7  6  9.0  3.0   5 11.0 10.0
VIKOR      1 12.0  4  8  2.0  5  7 10.0  3.0   6  9.0 11.0
WASPAS     2 12.0  1  8  4.0  7  6 10.0  3.0   5 11.0  9.0
WPM        2 12.0  1  7  4.0  8  6 10.0  3.0   5 11.0  9.0
WSM        4  7.0  9  5  1.0  6  2  8.0  3.0  11 10.0 12.0",
header = TRUE, row.names = 1))

result <- rankspearman(rankmat)

print(result$displaymat)
print(result$p_values)
```

`ranksrd` *SRD: Sum of Ranking Differences*

Description

Calculates the Sum of Ranking Differences (SRD) for a set of alternatives across multiple criteria. The SRD method assesses the performance of alternatives by comparing their actual rankings to a defined reference ranking. A lower SRD value indicates better agreement with the reference and thus, better performance.

Usage

```
ranksrd(rankmat, refopt = "mean", defrank = NULL, tiesmethod="average")
```

Arguments

`rankmat`	A `matrix` or `data.frame` where rows represent alternatives and columns represent criteria. Each cell contains the rank of an alternative for a specific criterion (e.g., 1 for best, 8 for worst).
`refopt`	A character string specifying the type of reference ranking to be used. Options are: • `"mean"`: The reference rank for each alternative is its mean rank across all criteria. This is the default.

- `"best"`: The reference rank for each alternative is 1 (ideal best performance in every criterion).
- `"worst"`: The reference rank for each alternative is the maximum possible rank found within the `rankmat` (ideal worst performance in every criterion).
- `"custom"`: The reference rank is provided by the user via the `defrank` argument. This allows for a specific, externally defined ideal ranking.

`defrank` A numeric vector providing custom reference ranks for each alternative. This argument is required if `refopt` is `"custom"`. If provided, its names must match the row names of `rankmat`. Defaults to `NULL` when not used.

`tiesmethod` The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The SRD method, as proposed by Károly Héberger (Héberger, 2010), offers a fair and unambiguous way to compare methods, models, or alternatives. It quantifies the "goodness" of an alternative by summing the absolute differences between its actual rank in each criterion and its corresponding reference rank. The procedure is repeated for each alternative, and the resulting SRD values are used to rank them.

The formula for SRD for a single alternative is:

$$SRD_i = \sum_{j=1}^{m} |R_{ij} - R_{ref,i}|$$

where R_{ij} is the rank of alternative i in criterion j, $R_{ref,i}$ is the reference rank for alternative i, and m is the total number of criteria. A smaller SRD value indicates that the alternative's ranking is closer to the reference ranking, implying better performance.

Value

A list containing the following components:

`srdvals` A named numeric vector of calculated SRD values for each alternative.

`srdrank` A named numeric vector representing the ranking of alternatives based on their SRD values, from lowest (best) to highest (worst).

`refrank` A named numeric vector of the reference ranks used for each alternative.

Note

Ensure that the input `rankmat` contains actual rank values (e.g., 1, 2, 3...) and not raw performance scores. The ranks should be consistent (e.g., 1 is always best, higher numbers are worse).

Author(s)

Cagatay Cebeci

References

Héberger, K. (2010). Sum of Ranking Differences (SRD) for method comparison and its relationship to the principle of parsimony. *Chemometrics and Intelligent Laboratory Systems*, 101(1), 1-8.

See Also

rank

Examples

```
# Sample rank matrix (8 alternatives, 15 criteria)
rankmat <- data.frame(
  C1 = c(7, 3, 2, 8, 1, 6, 5, 4), C2 = c(2, 1, 4, 7, 6, 3, 8, 5),
  C3 = c(5, 8, 1, 7, 4, 6, 2, 3), C4 = c(8, 2, 1, 4, 3, 5, 6, 7),
  C5 = c(1, 5, 4, 6, 2, 3, 7, 8), C6 = c(6, 8, 2, 3, 1, 4, 5, 7),
  C7 = c(6, 7, 1, 8, 2, 3, 5, 4), C8 = c(6, 3, 4, 5, 2, 1, 7, 8),
  C9 = c(1, 8, 3, 4, 6, 5, 7, 2), C10 = c(7, 8, 5, 2, 4, 6, 3, 1),
  C11 = c(6, 2, 3, 8, 5, 7, 4, 1), C12 = c(6, 8, 3, 7, 4, 1, 5, 2),
  C13 = c(2, 3, 7, 5, 6, 1, 8, 4), C14 = c(2, 1, 4, 5, 3, 8, 6, 7),
  C15 = c(1, 6, 4, 5, 8, 7, 3, 2),
  row.names = c("AltA", "AltB", "AltC", "AltD", "AltE", "AltF", "AltG", "AltH")
)

# Using mean rank as reference (default)
ressrd_mean <- ranksrd(rankmat, refopt = "mean")
print(ressrd_mean)

# Using the best possible rank (rank 1) as reference
ressrd_best <- ranksrd(rankmat, refopt = "best")
print(ressrd_best$srdvals)
print(ressrd_best$srdrank)
print(ressrd_best$rank)

# Using the worst possible rank (max rank in data) as reference
ressrd_worst <- ranksrd(rankmat, refopt = "worst")
print(ressrd_worst$srdvals)
print(ressrd_worst$srdrank)

# Using user defined reference
usrrank <- c(4, 6, 1, 8, 2, 3, 7, 5)
names(usrrank) <- rownames(rankmat)

ressrd_user <- ranksrd(rankmat, refopt = "custom", defrank = usrrank)
print(ressrd_user$srdvals)
print(ressrd_user$srdrank)
```

---

rankwilcox *Pairwise Wilcoxon Rank Sum Tests for Comparing Rows of a Matrix*

Description

Performs pairwise Wilcoxon Rank Sum tests /WSRT) to compare the rows of a given matrix. P-values are adjusted for multiple comparisons using the specified method.

Usage

```
rankwilcox(rankmat, alpha = 0.05, padjmethod = "none")
```

Arguments

rankmat	A numeric matrix where rows represent different groups or conditions and columns represent paired observations or measurements.
alpha	The significance level for determining statistical significance (default: 0.05).
padjmethod	The method for adjusting p-values for multiple comparisons. See p.adjust.methods for available options (default: "none").

Details

This function iterates through all unique pairs of rows in the input matrix and performs a Wilcoxon signed-rank test for each pair. The Wilcoxon signed-rank test is used to assess the significance of the difference between two related samples. The p-values obtained from these pairwise tests are then adjusted for multiple comparisons to control the family-wise error rate. The results are presented in a matrix format, with adjusted p-values in the upper triangle and test statistics in the lower triangle.

Value

A list containing the following components:

pvalmat	A matrix of raw p-values from the pairwise Wilcoxon signed-rank tests.
adjpvalmat	A matrix of p-values adjusted for multiple comparisons.
displaymat	A data frame where the upper triangle contains the adjusted p-values (formatted to two decimal places) and the lower triangle contains the Wilcoxon test statistics (formatted to two decimal places). The diagonal is empty.
alpha	The significance level used.
padjmet	The method used for p-value adjustment.

Author(s)

Cagatay Cebeci

See Also

rankcompare, rankentboot,rankentperm, rankmia, rankwssim, rankspearman, rankrangesim

Examples

```
# Rank matrix
rankmat <- matrix(c(
    3, 2, 3, 5, 6, 3, 3, 5, 5, 2, 5, 5, 2, 1, 3,
    7, 6, 7, 8, 8, 7, 7, 7, 7, 4, 7, 8, 7, 7, 7,
    1, 1, 1, 1, 2, 1, 1, 1, 1, 5, 1, 2, 1, 2, 1,
    8, 8, 8, 3, 5, 6, 8, 4, 3, 7, 3, 3, 8, 3, 3,
    2, 5, 2, 2, 1, 2, 2, 2, 2, 8, 2, 1, 3, 4, 2,
    4, 7, 5, 7, 7, 8, 4, 8, 8, 1, 8, 7, 6, 8, 8,
    6, 4, 6, 4, 3, 5, 6, 3, 4, 6, 4, 4, 5, 6, 6,
    5, 3, 4, 6, 4, 4, 5, 6, 6, 3, 6, 6, 4, 5, 5),
    nrow = 5, byrow = TRUE)
rownames(rankmat) <- paste0("MET", 1:nrow(rankmat))
colnames(rankmat) <- paste0("ALT", 1:ncol(rankmat))

res_wilcox <- rankwilcox(rankmat)
print(res_wilcox$displaymat)
```

`rankwssim` *Wojciech Sałabun Coefficient of Rankings Similarity (WS)*

Description

This function calculates the Wojciech Sałabun Coefficient of Rankings Similarity (WS) between different rankings of examined MCDA methods. The WS coefficient quantifies the similarity between two rankings of MCDA methods, giving higher priority to the similarity of positions at the top of the ranking.

Usage

```
rankwssim(rankmat)
```

Arguments

`rankmat`	A numerical matrix containing the rankings. Rows represent different rankings (e.g., results from different MCDM methods), and columns represent the ranked items (e.g., alternatives). The row names of the matrix should be the names of the rankings.

Details

The `wranksim` function takes a rank matrix as input and computes the WS coefficient for each pair of rankings within the matrix. The WS coefficient, introduced by Sałabun and Urbaniak, is calculated using the following formula: $WS = 1 - \sum_{i=1}^{n} \left(2^{-R_{xi}} \cdot \frac{|R_{xi} - R_{yi}|}{\max(|1-R_{xi}|, |N-R_{xi}|)}\right)$ where:

WS represents the value of the similarity coefficient.

N denotes the length of the ranking (number of alternatives).

R_{xi} represents the rank of the i-th element in ranking x.

R_{yi} represents the rank of the i-th element in ranking y.

The WS coefficient ranges from 0 to 1, where 1 indicates perfect similarity and 0 indicates complete dissimilarity between the two rankings. This metric prioritizes the agreement in the top ranks, which is often crucial in Multiple Criteria Decision Making (MCDM) for identifying the most preferred alternatives.

Due to the dependence of the WS coefficient value on the reference dataset (x), the resulting matrix of WS coefficient values is asymmetrical, unlike Spearman's rank correlation coefficient. Therefore, for accurate and fair comparisons, each MCDM method's ranking set should be used as the reference (x) when determining the WS coefficient value.

Based on Sałabun and Urbaniak (2020), Rachman et al (2024) states that two rankings are considered to have high ranking similarity if $WS > 0.689$, while $WS < 0.352$ indicates low similarity. MCDA methods are assumed to have a medium level of likeness if the WS value falls within the range from 0.352 to 0.689. Two MCDA methods are defined having totally similar if they reach absolute consensus by achieving Spearman Rank Correlation = +1 with WS = 1.

Value

The function returns a data frame containing the WS coefficients between all pairs of input rankings. The row and column names correspond to the names of the rankings from the input matrix. Each cell \((i, j)\) in the data frame represents the WS coefficient calculated with the \(i\)-th ranking as the reference (x) and the \(j\)-th ranking as the compared ranking (y). The coefficients are rounded to two decimal places and formatted as character strings.

Author(s)

Cagatay Cebeci

References

Rachman, A. P., Ichwania, C., Mangkuto, R. A., Pradipta, J., Koerniawan, M. D., & Sarwono, J. (2024). Comparison of multi-criteria decision-making methods for selection of optimum passive design strategy. *Energy and Buildings, 314*, 114285. <doi:10.1016/j.enbuild.2024.114285>

Sałabun, W., Urbaniak, K. (2020). A New Coefficient of Rankings Similarity in Decision-Making Problems. In: Krzhizhanovskaya, V., et al. Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science, vol 12138. Springer, Cham. <doi:10.1007/978-3-030-50417-5_47>

See Also

rankcompare, rankentboot, rankentperm, rankmia, rankwilcox, rankspearman, rankrangesim

Examples

```
# Rank matrix
rankmat <- as.matrix(read.table(text = "
                     G1   G2 G3 G4   G5 G6 G7   G8   G9 G10  G11  G12
ARAS              3 12.0  1  8  4.0  7  6 10.0  2.0   5 11.0  9.0
EDAS              3 12.0  2  8  4.0  9  6 10.0  1.0   5 11.0  7.0
```

```
ELECTRE4          3 11.5  4  8  1.5  7  5  9.5  1.5   6 11.5  9.5
FUCA              4  3.0 10  7  2.0  6  1  9.0  5.0  11  8.0 12.0
GRA               6  3.0 10  5  2.0  7  1  9.0  4.0  11  8.0 12.0
MABAC             3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
MAIRCA           10  1.0  9  5 12.0  6  7  4.0 11.0   8  3.0  2.0
MARCOS            3 12.0  1  8  4.0  7  6 10.0  2.0   5 11.0  9.0
MAUT              3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
MEGAN             5  3.0 11  7  1.0  6  2  9.0  4.0  10  8.0 12.0
MOORA             7  1.0 11  9  2.0  4  3  8.0  6.0  10  5.0 12.0
PROMET1           4 10.0  6  8  1.0  5  3  9.0  2.0   7 11.0 12.0
PROMET2           3  9.0  6  8  1.0  5  4 10.0  2.0   7 11.0 12.0
PROMET6           4 10.0  6  8  1.0  5  3  9.0  2.0   7 11.0 12.0
RAM               7  1.0 11  9  2.0  4  3  8.0  6.0  10  5.0 12.0
ROV               3 12.0  4  8  1.0  7  6  9.0  2.0   5 10.0 11.0
SMART             5  3.0 11  7  2.0  6  1  9.0  4.0  10  8.0 12.0
TOPSIS            2 12.0  4  8  1.0  7  6  9.0  3.0   5 11.0 10.0
VIKOR             1 12.0  4  8  2.0  5  7 10.0  3.0   6  9.0 11.0
WASPAS            2 12.0  1  8  4.0  7  6 10.0  3.0   5 11.0  9.0
WPM               2 12.0  1  7  4.0  8  6 10.0  3.0   5 11.0  9.0
WSM               4  7.0  9  5  1.0  6  2  8.0  3.0  11 10.0 12.0
", header = TRUE, row.names = 1))

print(rankmat)

# Calculate the WS similarity matrix using rank matrix
reswsm <- rankwssim(rankmat)
print(reswsm$displaymat)
```

`renewable` *Renewable Energy Grids*

Description

The `renewable` dataset provides a decision matrix (`dmat`), a benefit-cost vector (`bcvec`), and a weights vector (`weights`) for evaluating and ranking renewable energy grid alternatives using multi-criteria decision-making methods.

Usage

```
data(egrids)
```

Format

A list with the following components:

`dmat` A numeric matrix with 10 rows and 14 columns, where each row represents a grid alternative (G1 to G10) and each column represents a criterion. The criteria based on Prior et al (2025), are defined as follows:

- C1 - Local Community (%): Measures the impact of the energy system on improving local living standards and economic opportunities by providing reliable and clean energy services. Higher percentages indicate greater benefits for the community.
- C2 - Social Equity (%): Evaluates the fairness of energy access and benefit distribution among community members. Higher percentages reflect more equitable outcomes across different social groups.
- C3 - Education Awareness (%): Assesses the public's knowledge and awareness about renewable energy and sustainable practices. An elevated percentage implies a better-informed community ready to embrace clean energy solutions.
- C4 - Emission (Score): Quantifies the harmful emissions (e.g., CO_2 or other pollutants) generated by the energy system. This is a cost-type criterion; lower scores are more favourable as they signify reduced environmental harm.
- C5 - Biodiversity Loss (Score): Captures the negative impact of the energy project on local ecosystems and species diversity. Being a cost indicator, lower scores are preferable since they represent less biodiversity degradation.
- C6 - Sustainable Resources (%): Represents the efficient and sustainable use of natural resources within the energy system, including aspects like recycling and conservation. Higher percentages demonstrate improved resource sustainability.
- C7 - Technological Innovations (Score): Reflects the level of new technology adoption and innovations integrated into the energy system. Higher scores indicate more advanced technological implementations.
- C8 - Energy Efficiency (%): Measures how effectively the energy system converts input into useful output with minimal losses. A higher percentage shows a more efficient operation of the grid.
- C9 - Support Infrastructure (Score): Assesses the quality and modernity of the supporting infrastructure, such as smart grids, communication systems, and storage facilities. Higher scores denote a stronger, more modern infrastructure.
- C10 - Implementation Cost ($M): Quantifies the initial capital investment required for deploying the energy system, measured in millions of dollars. As a cost-type criterion, lower amounts are typically more desirable. In our simulation, advanced systems may involve higher costs indicative of superior technology.
- C11 - Economic Efficiency (
- C12 - Governmental Incentives (Score): Gauges the level and strength of governmental support, such as subsidies and incentives, available for the energy project. Higher scores suggest a more supportive policy environment.
- C13 - Macroeconomics (Score): Represents the broader economic benefits of the energy system, including impacts on investment, job creation, and infrastructure development. Higher scores are associated with positive macroeconomic contributions.
- C14 - Microeconomics (Score): Focuses on project-specific financial indicators (e.g., ROI, NPV) that reflect the viability and profitability of the energy system. Higher scores indicate more attractive economic prospects at the micro level.

For example, the first row (G1) of dmat contains the values: 78, 74, 66, 18, 11, 75, 65, 78, 78, 171, 69, 44, 59, and 50.

bcvec A numeric vector of length 14 indicating the type of each criterion:

- 1: Benefit criterion (higher is better).
- -1: Cost criterion (lower is better).

In this dataset, the benefit-cost vector is: 1, 1, 1, -1, -1, 1, 1, 1, 1, -1, 1, 1, 1, 1.

`weights` A numeric vector of length 14 containing the weights assigned to each criterion. The weights are as follows: 0.10, 0.05, 0.08, 0.08, 0.09, 0.13, 0.05, 0.06, 0.08, 0.09, 0.08, 0.04, 0.03, and 0.04, which sum to 1.

Details

The dataset is a synthetic dataset designed for evaluating and comparing renewable grid alternatives using multi-criteria decision-making methods. Each row in the decision matrix (`dmat`) represents a grid alternative (labeled G1 through G10) and each column corresponds to a specific criterion as described above. The benefit-cost vector (`bcvec`) indicates whether a criterion is to be maximized (benefit, 1) or minimized (cost, -1), while the weights vector (`weights`) reflects the relative importance of each criterion.

Author(s)

Cagatay Cebeci

References

Prior, J. F. P., Siluk, J. C. M. , Rigo, P. D. & Numes-Ramos, V. (2025). Factor for Decision-Making on Renewable Energy Modes in Rural Properties. *Power System Techonlogy, 49*(2), 354-393.

Examples

```
data(egrids)
# Display the decision matrix.
print(egrids$dmat)
# Display the benefit-cost vector.
print(egrids$bcvec)
# Display the criteria weights.
print(egrids$weights)
```

`rov` *ROV: Range of Value*

Description

Applies the Rank of Value (ROV) method on a decision matrix using specified weights and a benefit-cost vector.

Usage

```
rov(dmatrix, bcvec, weights, normethod = "maxmin", tiesmethod="average")
```

Arguments

dmatrix	The decision matrix. Rows represent alternatives, and columns represent criteria.
bcvec	A vector indicating the type of each criterion. Use 1 for benefit criteria and -1 for cost criteria.
weights	A vector of weights representing the relative importance of each criterion. Defaults to "equal" to apply equal weighting for criteria. See calcweights for details.
normethod	The normalization method to be applied. Defaults to "maxmin". See calcnormal for details.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The rov function implements the Rank Order Centroid (ROV) method, a multi-criteria decision-making (MCDM) technique used to rank alternatives. The function takes a decision matrix, a benefit-cost vector, and criterion weights as input. The decision matrix is first normalized using the specified normalization method. Then, the criterion weights are either provided directly or calculated using a specified method. Finally, a utility score is calculated for each alternative, and the alternatives are ranked based on these scores.

Value

A list includes the following results:

u_i	The utility score obtained for each alternative.
rank	The ranking of the alternatives based on their utility scores. Alternatives with the same score receive the average rank.

Author(s)

Cagatay Cebeci

References

Madic, M., Radovanovic, M., & Manic, M. (2016). Application of the ROV method for the selection of cutting fluids. Decision Science Letters, 5(2), 245-254.

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply ROV using user-defined weights
resrov_1 <- rov(dmatrix=dmat, bcvec=bc, weights=uw)
print(resrov_1)

# Apply ROV using equal weights
equw <- calcweights(dmatrix=dmat, bcvec=bc, type="equal", normethod="maxmin")
resrov_2 <- rov(dmatrix=dmat, bcvec=bc, weights = equw)
print(resrov_2)
```

---

sankeyplot *Sankey Diagram*

Description

Generates a Sankey diagram to visualize the relationship between ranking methods and alternatives.

Usage

```
sankeyplot(rankmat, fs = 12, nw = 30)
```

Arguments

rankmat	A numeric matrix of rankings, with rows representing ranking methods and columns representing alternatives. Lower rank indicates higher preference.
fs	Font size for node labels. Default is 12.
nw	Width of each node in pixels. Default is 30.

Details

Ideal for MCDM contexts where ranking comparison is crucial. The Sankey diagram shows the mapping between ranking *methods* (sources) and *alternatives* (targets), where link thickness reflects the strength of preference. Each link's width is computed as `k - Rank`, meaning that lower ranks (i.e., better positions) result in thicker flows.

This visualization allows quick identification of which alternatives are consistently favored across multiple methods, and highlights divergence in method behavior. For example, an alternative consistently ranked highly will attract wider links from several methods, creating a visually dominant path toward that target.

Value

An interactive Sankey diagram rendered using the `networkD3` package.

See Also

`sankeyNetwork`

Examples

```
rankmat <- matrix(c(
  5,9,7,8,4,2,10,11,3,1,12,6,
  3,12,8,4,9,6,7,10,2,5,11,1,
  2,12,4,8,6,7,9,10,5,1,11,3,
  2,11,6,8,3,5,9,10,1,4,12,7,
  7,9,6,8,5,2,11,10,3,1,12,4,
  5,9,7,8,4,2,10,11,3,1,12,6,
  5,8,6,9,4,2,10,11,3,1,12,7,
  4,6,9,8,1,2,7,10,3,5,11,12,
  5,8,6,9,1,3,7,12,2,4,11,10,
  5,7,6,9,1,2,8,12,3,4,11,10
), nrow = 10, byrow = TRUE)
rownames(rankmat) <- c("Equal","Critic","StdDev","Gini","Geometric",
  "Merec","Mpsi","Entropy","Roc","Rs")
colnames(rankmat) <- paste0("G",1:12)
sankeyplot(rankmat)
```

`selectmcdm` *Selection of Multi-Criteria Decision-Making Methods*

Description

This functions selects the most suitable MCDM method for a given problem by evaluating the ranking results of various MCDM methods (`rankmat`) against the decision matrix of alternatives (`dmatrix`). The function proposes the MCDM method that best ranks the alternatives with the highest composite criterion value among the "top m" alternatives. The number of "top" alternatives (`mval`) can be determined manually or dynamically using various methods (percentage-based, entropy-based, rule-based, or top-k frequency).

Usage

```
selectmcdm(rankmat, dmatrix, bcvec, weights, mval = NULL,
  mvmet = "percent", mvpars = list(perc = 0.25, rule = c(2, 4, 5), k=5, r=3),
  verbose = TRUE)
```

Arguments

rankmat	A matrix or data frame of ranks generated by different MCDM methods for the alternatives. Rows should represent MCDM methods, and columns should represent alternatives. Each cell should contain the rank assigned by the respective MCDM method to that alternative (1 being the best).
dmatrix	A decision matrix (matrix or data frame) containing the performance values of alternatives across different criteria. Rows should represent alternatives, and columns should represent criteria.
bcvec	A numeric vector indicating the benefit/cost type for each criterion. 1 indicates a benefit criterion (to be maximized), and -1 indicates a cost criterion (to be minimized). The length of the vector must be equal to the number of columns in dmatrix.
weights	A numeric vector specifying the weights assigned to each criterion. The length of the vector must be equal to the number of columns in dmatrix.
mval	The number of "top" alternatives to consider. If NULL, it will be determined dynamically based on the mvmet and mvpars arguments.
mvmet	A character string specifying the method to use for dynamic determination of mval when mval is NULL. Options are: • "percent": Based on a specified percentage of alternatives (mvpars$perc). • "entropy": Based on the entropy of the rank distribution. • "rule": Based on predefined rules (mvpars$rule) related to the total number of alternatives. • "topk": Based on the frequency of alternatives appearing in the top k positions (mvpars$k, mvpars$r).
mvpars	A list of additional parameters for the mvmet method. • perc: Percentage of alternatives to select. A numeric value between 0 and 1 for mvmet = "percent". • rule: A numeric vector of length 3 for mvmet = "rule" (e.g., c(2, 4, 5) for n < 5, 5 <= n <= 10, n > 10 respectively). • k: Integer, specifying the number of top-ranked alternatives to consider for mvmet = "topk". • r: Integer, specifying the minimum frequency threshold for an alternative to be counted in the top-k positions for mvmet = "topk".
verbose	Logical; if TRUE, detailed messages about the process will be printed to the console.

Details

The function evaluates multiple MCDM methods and selects the most suitable one based on cumulative performance values. As the extensions to the proposed method by Bandyopadhyay (2021), if `mval` is not provided, it is determined dynamically using one of the following methods:

- `"percent"`: Selects a percentage of alternatives based on `mvpars$perc`.
- `"entropy"`: Uses Shannon entropy to determine `mval` dynamically.
- `"rule"`: Applies predefined thresholds to determine `mval`.
- `"topk"`: Selects alternatives appearing frequently in the top `k` positions.

Value

Returns a list with the following components:

`m`	The calculated or user-specified number of top alternatives.
`suggestedmcdm`	The name of the MCDM method that yielded the highest cumulative criterion value for the given `m` alternatives.
`highcumval`	The highest cumulative criterion value achieved by the suggested MCDM method for the `m` alternatives.
`all_cumulvalm`	A named vector containing the cumulative criterion values for each MCDM method at the `m`-th alternative.

Author(s)

Cagatay Cebeci

References

Bandyopadhyay, S. (2021). Comparison among multi-criteria decision analysis techniques: A novel method. *Progress in Artificial Intelligence, 10*(2), 195-216. <doi:10.1007/s13748-021-00235-5>

Examples

```
# Decison matrix
datadf <- data.frame(
  P = c(4800, 10000, 3890, 7800, 4450, 5000, 1990, 3000, 4400, 4000),
  MSS = c(2700, 3000, 4000, 2000, 3000, 2700, 3000, 2200, 2400, 2800),
  MSL = c(12.7, 20, 13, 8, 14.5, 12.8, 40, 12.7, 14.3, 15.2),
  LC = c(50, 70, 10, 8, 6.8, 10, 6, 6.5, 14.3, 8),
  MC = c(3000, 4000, 500, 1500, 1500, 2000, 350, 1500, 500, 1000),
  MR = c(980, 1000, 800, 900, 1040, 500, 510, 1200, 850, 900),
  W = c(150, 200, 210, 100, 500, 350, 200, 230, 200, 250),
  WRT = c(6, 1, 3, 1, 5, 1, 2, 1, 3, 2)
)
rownames(datadf) <- paste0("Rob.", 1:10)
dmat <- as.matrix(datadf)

# Benefit-Cost vector
bc <- c(
```

```
  P = -1,  # Price
  MSS = 1, # Maximum sewing speed (rpm)
  MSL = 1, # Maximum stitch length (mm)
  LC = 1,  # Load capacity (kg)
  MC = 1,  # Memory capacity
  MR = 1,  # Manipulator reach
  W = -1,  # Weight (kg)
  WRT = 1  # Warranty (years)
)

# Weights
uwei <- c(
  P = 0.172414,
  MSS = 0.172414,
  MSL = 0.137931,
  LC = 0.12069,
  MC = 0.013448,
  MR = 0.137931,
  W = 0.068966,
  WRT = 0.086207
)

rankdf <- data.frame(
  R1 = c(5, 10, 10, 8, 9, 8, 3, 2, 4, 2, 2, 7, 1, 4),
  R2 = c(10, 9, 9, 10, 7, 10, 1, 1, 1, 1, 1, 10, 2, 1),
  R3 = c(3, 6, 8, 4, 1, 5, 4, 5, 8, 2, 7, 3, 3, 6),
  R4 = c(9, 3, 4, 5, 8, 9, 6, 3, 2, 6, 4, 9, 7, 3),
  R5 = c(7, 7, 6, 9, 3, 6, 2, 4, 5, 4, 3, 5, 5, 2),
  R6 = c(8, 1, 1, 7, 4, 7, 7, 6, 3, 5, 5, 8, 10, 5),
  R7 = c(1, 5, 7, 1, 2, 1, 9, 8, 10, 3, 10, 1, 4, 10),
  R8 = c(2, 2, 5, 3, 6, 2, 10, 10, 9, 7, 9, 2, 6, 9),
  R9 = c(6, 4, 2, 2, 10, 3, 8, 9, 6, 9, 8, 6, 9, 8),
  R10 = c(4, 8, 3, 6, 5, 4, 5, 7, 7, 8, 6, 4, 8, 7)
)
rownames(rankdf) <- c("TOPSIS", "MAUT", "MACBETH",
   "ORESTE", "VIKOR", "SIR", "EVAMIX", "ARAS", "MOORA",
   "COPRAS", "WASPAS", "TODIM", "MABAC", "MARE")
rankmat <- as.matrix(rankdf)

# With a user-defined specified mval
results_user <- selectmcdm(rankmat, dmat, bc, uwei, mval = 4, verbose = FALSE)
print(results_user)

# With percentage-based mval
results_percent <- selectmcdm(rankmat, dmat, bc, uwei,
   mvmet = "percent", mvpars = list(perc = 0.25), verbose = FALSE)
print(results_percent)

# With entropy-based mval
results_entropy <- selectmcdm(rankmat, dmat, bc, uwei,
   mvmet = "entropy", verbose = FALSE)
print(results_entropy)
```

```
# With rule-based mval
results_rule <- selectmcdm(rankmat, dmat, bc, uwei, mvmet = "rule",
   mvpars = list(rule = c(2, 6, 8)), verbose = FALSE)
print(results_rule)

# With top-k based mval
results_topk_m <- selectmcdm(rankmat, dmat, bc, uwei, mvmet = "topk",
   mvpars = list(k = 3, r = nrow(dmat)/2), verbose = FALSE)
print(results_topk_m)
```

---

sensana *Sensitivity and Stability Analyses for MCDA*

Description

Performs various sensitivity and stability analyses on a matrix of rank results from Multi-Criteria Decision Analysis (MCDA).

Usage

```
sensana(rankmat)
```

Arguments

rankmat	A numeric matrix where rows represent alternatives and columns represent different scenarios (e.g., different weighting vectors or MCDA methods). Each element in the matrix is the rank of the alternative in the corresponding scenario.

Details

This function calculates several measures to assess the sensitivity and stability of alternative rankings across different scenarios. The computed measures include:

- SD: The standard deviation of the ranks for each alternative across all scenarios, indicating the dispersion of its ranks.
- RSI: Rank Stability Index (Stability scores) for each alternative, calculated as 1 minus the proportion of rank changes across scenarios. A higher score indicates greater rank stability.
- RVOL: Rank Volatility scores for each alternative, representing the proportion of rank changes between consecutive scenarios. A lower score indicates less rank volatility.
- sensscores: Weight Sensitivity scores for each scenario, representing the average absolute rank change of all alternatives when compared to the first scenario. Lower scores suggest higher stability of the overall ranking relative to the first scenario.

Value

A list containing the following components:

`stability_table`
: A data frame of stability measures for each alternative.

`spearmancor`
: A numeric matrix of Spearman's rank correlation coefficients between scenarios (between -1 and 1).

`sensscores`
: A numeric vector of weight sensitivity scores for each scenario.

Author(s)

Cagatay Cebeci

Examples

```
# Example rank matrix
rankmat <- matrix(
  c(1, 2, 1, 3,
    2, 1, 3, 1,
    3, 3, 2, 2),
  nrow = 3,
  byrow = TRUE,
  dimnames = list(c("ALT1", "ALT2", "ALT3"), paste0("S_", 1:4))
)

# Perform sensitivity and stability analysis
results <- sensana(rankmat)

print(results)
```

---

sensplot *Plot Weight Sensitivity Analysis Results*

Description

Creates a bar or pie chart to visualize the frequency of ranking patterns resulting from a sensitivity analysis performed by `weisana`.

Usage

```
sensplot(senstable, topn = 10, colpal = NULL, type = "bar", mtitle = NULL)
```

Arguments

`senstable`
: A data frame containing ranking patterns and their frequencies, typically the `sensitivity_table` output from `weisana()`. Must include `Pattern` and `Percent` columns.

`topn`
: Number of top-ranked patterns to include in the plot. Defaults to `10`.

colpal	Optional color palette for the bars or pie slices. If NULL, a rainbow() palette is used.
type	Type of plot. Choose between "bar" (default) or "pie".
mtitle	Main title of the plot. If NULL, defaults to "Sensitivity Analysis".

Details

This function is a companion to weisana() and is designed to offer a quick graphical summary of how frequently different ranking outcomes appear across weight scenarios. The bar plot also includes percentage labels above the bars for easy interpretation.

Value

No return value.

Author(s)

Cagatay Cebeci

See Also

weisana for generating the sensitivity table input.

Examples

```
# Simulated example
set.seed(123)
results <- data.frame(
  Pattern = c("2,1,2,4", "2,1,3,4", "1,3,2,4", "1,3,4,2"),
  Count = c(50, 30, 15, 5),
  Percent = c(50, 15, 20, 5),
  stringsAsFactors = FALSE
)

sensplot(results, topn = 5, type = "bar")
sensplot(results, type = "pie", mtitle = "Ranking Pattern Distribution")
```

---

smart *SMART: Simple Multi-Attribute Rating Technique*

Description

This function applies the Simple Multi-Attribute Rating Technique (SMART) for multi-criteria decision analysis. The SMART evaluates alternatives based on weighted aggregation of their values determined by value functions.

Usage

```
smart(dmatrix, bcvec=NULL, weights, normethod=NULL, valfuncs = NULL, tiesmethod="average")
```

Arguments

dmatrix	A numeric matrix representing the decision matrix. Rows are alternatives and columns are criteria.
bcvec	A numeric vector indicating the benefit/cost nature of each criterion. Use 1 for benefit (to be maximized) and -1 for cost (to be minimized). The length of bcvec must equal the number of columns in dmatrix. Default is NULL because SMART uses a different techique.
weights	A vector of weights representing the relative importance of each criterion. Defaults to "equal" to apply equal weighting for criteria. See calcweights for details.
normethod	The normalization method to be applied. Defaults to NULL. See calcnormal for details.
valfuncs	An optional list of value functions for each criterion. Each element of the list should be a function that takes a numeric value (criterion performance) and returns a scaled value (typically between 0 and 1). If NULL (default), linear value functions (MaxMin scaling) are applied to each criterion.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The SMART method involves defining value functions for each criterion to scale the performance values and assigning weights to the criteria to reflect their importance. The overall score for each alternative is then calculated as the weighted sum of its values across all criteria. Alternatives are ranked based on their overall scores in descending order.

If weights is a numeric vector, these weights are used directly after normalization.

If valfuncs is not provided, linear value functions (min-max scaling) are used to normalize the performance values of each criterion to a 0-1 range.

Value

A list containing the following components:

Value_Matrix	A numeric matrix containing the scaled values of each alternative for each criterion based on the value functions.
Weighted_Value_Matrix	A numeric matrix containing the weighted scaled values of each alternative for each criterion.
Overall_Scores	A numeric vector of the overall scores for each alternative.
Ranking	A data frame with the ranked alternatives, their SMART scores, and their ranks.
rank	A numeric vector representing the rank of each alternative (lower rank is better).

Author(s)

Cagatay Cebeci

References

Olson, D.L. (1996). Smart. In: *Decision Aids for Selection Problems*. Springer Series in Operations Research. Springer, New York, NY. <doi:10.1007/978-1-4612-3982-6_4>

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)
ressmart_1 <- smart(dmatrix=dmat, bcvec=bc, weights=uw)
print(ressmart_1)

# Criteria weights with std. dev. method
sdw <- calcweights(dmat, bcvec=bc, type="sdev")
ressmart_2 <- smart(dmatrix=dmat, bcvec=bc, weights = sdw)
print(ressmart_2)

# Custom value functions
vf <- list(
  function(x) 1 - (x - min(dmat[, 1])) / (max(dmat[, 1]) - min(dmat[, 1])),
  function(x) (x - min(dmat[, 2])) / (max(dmat[, 2]) - min(dmat[, 2])),
  function(x) 1 - (x - min(dmat[, 1])) / (max(dmat[, 1]) - min(dmat[, 1])),
  function(x) (x - min(dmat[, 2])) / (max(dmat[, 2]) - min(dmat[, 2]))
)
ressmart_3 <- smart(dmatrix=dmat, bcvec=bc, weights=uw, valfuncs = vf)
print(ressmart_3)
```

---

topsis *TOPSIS: Technique for Order of Preference by Similarity to Ideal Solution*

Description

This function implements the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method for multi-criteria decision-making by evaluating alternatives based on their distances from the ideal and anti-ideal solutions.

Usage

```
topsis(dmatrix, bcvec, weights, normethod = "maxmin", tiesmethod = "average")
```

Arguments

`dmatrix`	Decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	Benefit-cost vector, specifying for each criterion whether it is a benefit (`1`) or a cost (`-1`).
`weights`	A numeric vector of user-defined weights representing the relative importance of each criterion, or a character string to calculate it internally. Defaults to `"equal"` to apply equal weighting for criteria. See `calcweights` for details.
`normethod`	The normalization method to be applied. Defaults to `"maxmin"`. See `calcnormal` for details.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method for Multi-Criteria Decision Making (MCDM) was originally developed by Ching-Lai Hwang and Kwangsun Yoon in 1981. Their seminal work, "Multiple Attribute Decision Making: Methods and Applications", published by Springer-Verlag in 1981, introduced the fundamental concepts and methodology of TOPSIS. Later, there were further developments and refinements of the method by Yoon in 1987, and by Hwang, Lai, and Liu in 1993. The TOPSIS method evaluates alternatives by normalizing the decision matrix, applying weights, and calculating distances to the positive ideal solution (`pis`) and the negative ideal solution (`nis`). The closeness coefficient (`p`) is computed as the ratio of the distance to the negative ideal solution (`Dm`) to the sum of the distances to both ideal solutions (`Dm + Dp`).

Value

A list containing the following components:

`normalized_matrix`	The normalized decision matrix.
`weighted_matrix`	The weighted decision matrix.
`pis`	The positive ideal solution for each criterion.
`nis`	The negative ideal solution for each criterion.
`Dm`	The distance from each alternative to the negative ideal solution.

Dp	The distance from each alternative to the positive ideal solution.
p	The closeness coefficient for each alternative.
rank	The rank of closeness coefficients.

Author(s)

Cagatay Cebeci

References

Hwang, C.L. & Yoon, K. (1981). Multiple Attribute Decision Making: Methods and Applications. New York: Springer-Verlag.

Yoon, K. (1987). A reconciliation among discrete compromise solutions. *Journal of the Operational Research Society, 38*(3), 277-286. <doi:10.1057/jors.1987.44>

Hwang, C. L., Lai, Y. J., & Liu, T. Y. (1993). A new approach for multiple objective decision making. *Computers & Operations Research, 20*(8), 889-899.

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply TOPSIS with user-defined weights
restopsis_1 <- topsis(dmatrix = dmat, bcvec = bc, weights = uw)
print(restopsis_1)

# Apply TOPSIS with entropy weighting method
entw <- calcweights(dmat, bcvec=bc, type="entropy")
restopsis_2 <- topsis(dmatrix = dmat, bcvec = bc, weights = entw)
print(restopsis_2$p) # Closeness coefficients for ranking alternatives
print(restopsis_2$rank) # Rank of alternatives
```

vikor *VIKOR: VIsekriterijumska optimizacija i KOmpromisno Resenje*

Description

Implements the VIsekriterijumska optimizacija i KOmpromisno Resenje (VIKOR) (Multi-criteria optimization and Compromise Solution in English) method to rank alternatives by evaluating their compromise solutions based on the closeness to ideal solutions.

Usage

```
vikor(dmatrix, bcvec, weights, normethod = "maxmin", v = 0.5, tiesmethod = "average")
```

Arguments

dmatrix	Decision matrix where rows represent alternatives and columns represent criteria.
bcvec	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
weights	A numeric vector of user-defined weights representing the relative importance of each criterion, or a character string to calculate it internally. Defaults to "equal" to apply equal weighting for criteria. See calcweights for details.
normethod	The normalization method to be applied. Defaults to "maxmin". See calcnormal for details.
v	The weight of the strategy for maximum group utility. A value between 0 and 1. Default is 0.5.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The VIKOR method normalizes the decision matrix, applies weights, and calculates the utility measure (S), the regret measure (R), and the compromise measure (Q). These measures are used to rank the alternatives, with the best compromise solution identified as the one with the smallest Q value.

Value

A list containing the following components:

nmatrix	The normalized decision matrix.
fminus	A vector of the worst performance values (minimum values) for each criterion.
fstar	A vector of the best performance values (maximum values) for each criterion.
S	A vector of the group utility values for each alternative.
R	A vector of the individual regret values for each alternative.
Q	A vector of the compromise values for each alternative.
rank	The ranking of alternatives based on the Q values.

Author(s)

Cagatay Cebeci

References

Opricović, S. (1979). Multicriteria Optimization in Civil Engineering. (Ph.D. Dissertation, in Serbian), Faculty of Civil Engineering, Belgrade.

Duckstein, L., & Opricovic, S. (1980). Multiobjective optimization in river basin development. *Water Resources Research, 16*(1), 14-20.

Opricović, S. (1998). Multicriteria optimization of civil engineering systems. *Faculty of civil engineering, Belgrade, 2*(1), 5-21.

Opricovic, S., & Tzeng, G. H. (2004). Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. *European journal of Operational Research, 156*(2), 445-455. <doi:10.1016/S0377-2217(03)00020-1>

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

resvikor_1 <- vikor(dmatrix=dmat, bcvec=bc, weights=uw, v = 0.5)
print(resvikor_1)

resvikor_2 <- vikor(dmatrix=dmat, bcvec=bc, weights=uw, v = 0.1)
print(resvikor_2)

criw <- calcweights(dmat, bcvec=bc, type="critic")
resvikor_3 <- vikor(dmatrix=dmat, bcvec=bc, weights=criw,
  normethod="vector", v = 0.5)
print(resvikor_3)
```

waspas *WASPAS: Weighted Aggregated Sum Product Assessment*

Description

An implementation of the Weighted Aggregated Sum Product Assessment (WASPAS) method, which combines the Weighted Sum Model (WSM) and the Weighted Product Model (WPM), to evaluate and rank alternatives based on normalized scores and assigned weights.

Usage

```
waspas(dmatrix, bcvec, weights, normethod = "linear", v = 0.5, tiesmethod = "average")
```

Arguments

dmatrix	Decision matrix where rows represent alternatives and columns represent criteria.
bcvec	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
weights	A numeric vector of user-defined weights representing the relative importance of each criterion. See calcweights for details.
normethod	The normalization method to be applied. Defaults to "linear". See calcnormal for details.
v	A parameter that balances the contributions of WSM and WPM. A value between 0 and 1, where v = 0.5 gives equal weight to both models. Default is 0.5.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The WASPAS method integrates the advantages of the Weighted Sum Model (WSM) and the Weighted Product Model (WPM) through a parameterized combination. Alternatives are evaluated based on a sum-based score (qsum) and a product-based score (qprod), with the final score (p) calculated as a weighted average of the two.

Value

A list containing the following components:

nmatrix	The normalized decision matrix.
qsum	The aggregated score based on the Weighted Sum Model.
qprod	The aggregated score based on the Weighted Product Model.
p	The final score for each alternative, calculated as a combination of qsum and qprod.
rank	The ranks of final scores for alternatives.

Author(s)

Cagatay Cebeci

References

Chakraborty, S., Zavadskas, E. K., & Antuchevičienė, J. (2015). Applications of WASPAS method as a multi-criteria decision-making tool. URI <https://etalpykla.vilniustech.lt/handle/123456789/151097>.

Zavadskas, E. K., Vilutienė, T., Turskis, Z., & Šaparauskas, J. (2014). Multi-criteria analysis of Projects' performance in construction. *Archives of Civil and Mechanical Engineering, 14*, 114-121. <doi:10.1016/j.acme.2013.07.006>

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

# Apply WASPAS with user-defined weights
reswaspas_1 <- waspas(dmatrix = dmat, bcvec=bc,
   weights=uw, normethod = "linear", v = 0.5)
print(reswaspas_1)

# Apply WASPAS with GINI weights
giniw <- calcweights(dmat, bcvec=bc, type="gini")
reswaspas_2 <- waspas(dmatrix = dmat, bcvec=bc,
   weights=giniw, normethod = "linear", v = 0.5)
print(reswaspas_2)
```

`weisana` *Weight Sensitivity Analysis for Multi-Criteria Decision-Making*

Description

Performs a weight sensitivity analysis by generating modified weight scenarios and applying a user-defined MCDA ranking method (e.g., TOPSIS, VIKOR) to each. Summarizes unique ranking patterns in rankings.

Usage

```
weisana(dmatrix, bcvec, weights, weimethod = "gradual", weipars = NULL,
        mcdamethod = topsis, methodpars, sensplot = TRUE)
```

Arguments

`dmatrix`	Decision matrix (numeric matrix or data frame) where rows represent alternatives and columns represent criteria.
`bcvec`	Numeric vector with values -1 or 1, indicating whether each criterion is a cost (-1) or benefit (+1).
`weights`	Numeric vector of weights for criteria. If omitted, equal weights are assigned.
`weimethod`	Weight perturbation method: `"gradual"` or `"random"`.
`weipars`	List of parameters for the weight generation method. Includes `rp`, `ss`, and `niters`.
`mcdamethod`	Ranking function to apply, such as `topsis` or `vikor`. Must return a list with a `$rank` vector.
`methodpars`	Named list of additional parameters passed to `mcdamethod`.
`sensplot`	Plot for ranking distributions obtained with different weighting scenarios.

Details

This function allows for analyzing the robustness of MCDA results under different weight variations. It supports both gradual and random perturbation schemes and is agnostic to the MCDA method used.

Value

A named list with the following components:

`computing_time`	Time taken to perform ranking computations.
`weights_matrix`	Matrix of generated weight scenarios.
`ranking_matrix`	Matrix of rankings per weight set.
`sensitivity_table`	Data frame showing unique ranking patterns, counts, and percentages.

Author(s)

Cagatay Cebeci

Examples

```
# Sample decision matrix and parameters
dmat <- matrix(runif(20), nrow=5)
rownames(dmat) <- paste0("A", 1:nrow(dmat))
colnames(dmat) <- paste0("C", 1:ncol(dmat))
print(dmat)
# Benefit-cost vector
bc <- c(1, 1, -1, 1)

# A very simple custom MCDA function
customfunc <- function(dmatrix, bcvec, weights, v=0.5, tiesmethod="average") {
  scores <- (rowSums(scale(dmatrix) * weights))^v
  ranks <- rank(-scores, ties.method=tiesmethod)
  list(rank = ranks)
}

# Sensitivity of the custom MCDM with gradual weight modification
mp <- list(v = 0.6, tiesmethod="average")
wp <- list(rp = c(0.1, 0.5, 0.7))
resgrawei <- weisana(dmatrix = dmat, bcvec = bc,
     weimethod = "gradual", weipars = wp,
     mcdamethod = customfunc, methodpars = mp)
print(resgrawei)

# Sensitivity of the custom MCDM with random weight modification
mp <- list(v = 0.6)
wp <- list(ss = 0.1, niters=20)
resrandwei <- weisana(dmatrix = dmat, bcvec = bc,
     weimethod = "random", weipars = wp,
     mcdamethod = customfunc, methodpars = mp)
print(resrandwei)

# Test TOPSIS
mp <- list()
wp <- list(rp = seq(0.01, 0.5, 0.05))

topsisgrawei <- weisana(dmatrix = dmat, bcvec = bc,
     weimethod = "gradual", weipars = wp,
     mcdamethod = topsis, methodpars = mp)
print(topsisgrawei)

# Test VIKOR
mp <- list(v=0.5)
wp <- list(rp = seq(0.01, 0.5, 0.05))

vikorgrawei <- weisana(dmatrix = dmat, bcvec = bc,
     weimethod = "gradual", weipars = wp,
     mcdamethod = vikor, methodpars = mp, sensplot=FALSE)
```

```
print(vikorgrawei)
sensplot(vikorgrawei$sensitivity_table,
   mtitle="Weight Sensivity Analysis for VIKOR", colpal=terrain.colors(10))

# Test WASPAS
mp <- list(v=0.5, normethod="linear", tiesmethod="average")
wp <- list(rp = seq(0.01, 0.6, 0.01))

waspasgrawei <- weisana(dmatrix = dmat, bcvec = bc,
     weimethod = "gradual", weipars = wp,
     mcdamethod = waspas, methodpars = mp)
print(waspasgrawei)
sensplot(waspasgrawei$sensitivity_table,
   mtitle="Weight Sensivity Analysis for WASPAS", colpal=terrain.colors(10))

waspasrandwei <- weisana(dmatrix = dmat, bcvec = bc,
     weimethod = "random", weipars = list(ss=0.05, niters=50),
     mcdamethod = waspas, methodpars = mp)
print(waspasrandwei)
sensplot(waspasrandwei$sensitivity_table,
   mtitle="Weight Sensivity Analysis (random) for WASPAS", colpal=terrain.colors(10))
```

---

wpm *WPM: Weighted Product Method*

Description

Implements the Weighted Product Method (WPM) to evaluate and rank alternatives using multiplicative aggregation of weighted and normalized criteria.

Usage

```
wpm(dmatrix, bcvec, weights, normethod = "vector", tiesmethod = "average")
```

Arguments

`dmatrix`	Decision matrix where rows represent alternatives and columns represent criteria.
`bcvec`	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
`weights`	A numeric vector of user-defined weights representing the relative importance of each criterion, or a character string to calculate it internally. Defaults to `"equal"` to apply equal weighting for criteria. See `calcweights` for details.
`normethod`	The normalization method to be applied. Defaults to `"vector"`. See `calcnormal` for details.
`tiesmethod`	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

The WPM evaluates alternatives by normalizing the decision matrix, raising each criterion to the power of its respective weight, and aggregating the values multiplicatively. This method calculates the relative performance (p) of each alternative for ranking purposes.

Value

A list containing the following components:

nmatrix	The normalized decision matrix.
weighted_matrix	The decision matrix after applying weights using the power operator.
p	The final scores for each alternative, calculated as the product of weighted and normalized criteria.
rank	The ranks of final scores for alternatives.

Author(s)

Cagatay Cebeci

References

Miller, D. W., & Starr, M. K. (1960). Executive decisions and operations research (Vol. 40). Englewood Cliffs, NJ: Prentice-Hall.

Zavadskas, E. K., Vilutiene, T., Turskis, Z., & Šaparauskas, J. (2014). Multi-criteria analysis of Projects' performance in construction. *Archives of Civil and Mechanical Engineering, 14*, 114-121. <doi:10.1016/j.acme.2013.07.006>

See Also

calcweights, calcnormal

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)
```

```
# Apply WPM with user-defined weights
reswpm_1 <- wpm(dmatrix=dmat, bcvec=bc, weights=uw)
print(reswpm_1)

# Apply WPM with MEREC weights
merecw <- calcweights(dmat, bcvec=bc, type="merec")
reswpm_2 <- wpm(dmatrix=dmat, bcvec=bc, weights=merecw, normethod="vector")
print(reswpm_2)
```

---

wsm *WSM: Weighted Sum Method / SAW: Simple Additive Weighting*

Description

Implementation of the Weighted Sum Method (WSM) or Simple Addiitive Weighting (SAW) to evaluate and rank alternatives by aggregating weighted normalized scores for each criterion.

Usage

```
wsm(dmatrix, bcvec, weights, normethod = NULL, tiesmethod = "average")
```

Arguments

dmatrix	Decision matrix where rows represent alternatives and columns represent criteria.
bcvec	Benefit-cost vector, specifying for each criterion whether it is a benefit (1) or a cost (-1).
weights	A numeric vector of user-defined weights representing the relative importance of each criterion. See calcweights for details.
normethod	The normalization method to be applied. Defaults to NULL. See calcnormal for details.
tiesmethod	The ties method to be applied in rank calculation. The default value is 'average'. Other options are 'min', 'max', and 'none'.

Details

Simple Additive Weighting method emerged from the early developments in multi-attribute utility theory and decision analysis in the 1960s, with significant contributions from researchers like Ward Edwards and David E. Bell. The method was then further popularized and applied in various contexts within the MCDM field. The WSM evaluates alternatives by normalizing the decision matrix, applying weights to criteria using multiplication, and aggregating the weighted scores (p) for each alternative. This straightforward method provides a clear ranking of alternatives based on their overall performance.

Value

A list containing the following components:

`nmatrix`	The normalized decision matrix.
`weighted_matrix`	The decision matrix after applying weights.
`p`	The aggregated weighted scores for each alternative.
`rank`	The ranks of aggregated weighted scores for alternatives.

Author(s)

Cagatay Cebeci

References

Zavadskas, E. K., Vilutiene, T., Turskis, Z., & Šaparauskas, J. (2014). Multi-criteria analysis of Projects' performance in construction. *Archives of Civil and Mechanical Engineering, 14*, 114-121. <doi:10.1016/j.acme.2013.07.006>

Panjaitan, M. I. (2019). Simple Additive Weighting (SAW) method in determining beneficiaries of foundation benefits. *Login, 13*(1), 19-25. <doi:10.24224/login.v13i1.22>

See Also

`calcweights`, `calcnormal`

Examples

```
# Sample decision matrix
dmat <- matrix(c(
  84, 10, 5, 20,
  66, 18, 8, 15,
  74, 12, 6, 25,
  90, 22, 9, 18,
  68, 18, 7, 15
), nrow = 5, byrow = TRUE)
rownames(dmat) <- c("A1", "A2", "A3", "A4", "A5")
colnames(dmat) <- c("C1", "C2", "C3", "C4")

# Benefit-cost vector (1: benefit, -1: cost)
bc <- c(1, -1, 1, -1)

# User-defined weights
uw <- c(0.1, 0.4, 0.3, 0.2)

reswsm_1 <- wsm(dmatrix=dmat, bcvec=bc, weights=uw)
print(reswsm_1)

reswsm_2 <- wsm(dmatrix=dmat, bcvec=bc, weights=uw, normethod = "sum")
print(reswsm_2)
```

```
reswsm_3 <- wsm(dmatrix=dmat, bcvec=bc, weights=uw, normethod = "maxmin")
print(reswsm_3)
```

Index