Package 'MCDASupport'

May 11, 2022

Type Package

Title Support functions for solving Multiple-criteria Decision-making Problems

Version 0.21

Date 2022-05-09

Author Pavel Šenovský pavel.senovsky@vsb.cz [aut, cre]

Maintainer Pavel Šenovský <pavel.senovsky@vsb.cz>

Description Functions to process "outranking" using various methods existing in the literature.

Depends igraph, diagram, dplyr, stats, graphics

License GPL (>= 3)

NeedsCompilation no

Encoding UTF-8

Imports mathjaxr

RdMacros mathjaxr

RoxygenNote 7.1.2

R topics documented:

`MCDASupport-package` *Functions for Solving Multiple-criteria Decision-making Problems*

Description

The outranking methods constitute one of the most fruitful approach in the field of Multiple Criteria Decision Making (MCDM).

They main feature is to compare all feasible alternatives or actions by pair building up some binary relations, crisp or fuzzy, and then exploit in appropriate way these relations in order to obtain final recommendations.

This package contains functions to process ELECTRE methods existing in the literature. See, e.g., `http://en.wikipedia.org/wiki/ELECTRE` about the outranking approach and the foundations of ELECTRE methods.

At present time package supports:

- ELECTRE I
- ELECTRE 1S (experimental implementation - do not use in production environment)
- ELECTRE II

- ELECTRE III
- ELECTRE IV
- ELECTRE TRI
- PROMETHEE I (experimental implementation - do not use in production environment)
- PROMETHEE II (experimental implementation - do not use in production environment)
- PROMETHEE III (experimental implementation - do not use in production environment)
- WSM - weighted sum method
- various normalization approaches (norm_* functions)

Main purpose of the package is to study inner workings of various methods used in multicriteria decision making and experiment with it. The provided functions may contain errors or the approaches represented by the functions may not be applicable to every decision problem. So bo carefull if you want to use the functions as actual decision support tool.

Authors of the package are not to be held liable for possible bad decision, you make, even if it is based on results of the functions contained in this package.

Details

Package:	MCDASupport
Type:	Package
Version:	0.11
Date:	2022-02-01
License:	GPL (>= 3)

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Roy, B. (1996) Multiple Criteria Methodology for Decision Aiding, Dordrecht: Kluwer Academic.

Roy, B. and Bouyssou, D. (1985). An example of comparison of two decision-aid models,in G. Fandel and J. Spronk (eds)

Ballestero, E. and Romero, C. (1998) Multiple Criteria Decision Making and its Applications to Economic Problems, Boston-Dordrecht-London: Kluwer Academic.

Vincke, P. (1992) Multi-criteria Decision-Aid, John Wiley, Chichester.

Roy B. (1968) Classement et choix en presence de points de vue multiples (la methode Electre), Revue Francaise d Informatique et de Recherche Operationnelle.

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

ELECTRE1_Kernel *ELECTRE1_Kernel - computes so called kernell of the decision for ELECTRE I and 1S methods*

Description

Computes kernell of the solution as the set of alternatives which are not dominated by any other alternative. Such alternatives then for example can be excluded from decision making as they are clearly suboptimal.

The results are presented in graphical form as network diagram with flows representing domination relation and two vectors with alternatives in kernel and dominated alternatives.

Usage

```
ELECTRE1_Kernel(am)
```

Arguments

am	adjacancy matrix or credibility matrix (1S)

Value

Returns:

graph	graphical representation of domination of one alternative over another
dominated	vector of alternatives identified as dominated
kernel	oposite to dominated vector. Consist for alternatives not dominated by other alternatives, forming kernel of the solution.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

See Also

ELECTRE I: Electre_1

ELECTRE 1S: Electre_1S

Electre3_ConcordanceIndex

Electre3_ConcordanceIndex : Method used to compute concordance matrix for ELECTRE III and 1S methods

Description

Internal method for computing concordance matrix for ELECTRE III and 1S methods. Concordance matrix is one of two angles ELECTRE methods use to derive preference for the alternatives, the other being Discordance matrix.

Concordancce matrix (index) measures strength of the statement that alternative a outranks alternative b, while discordance matrix (index) together with discordance threshold (exceeding this threshold) can prevent such outranking.

To compute overal concordance matrix, partial concordance matrix of the criteria needs to be computed first:

$$c_j(a,b) = 1\ if\ PM_j(a) + Q_j(a) \geq PM_j(b)$$

$$c_j(a,b) = 0\ if\ PM_j(a) + P_j(a) < PM_j(b))$$

$$else\ c(a,b) = \frac{PM_j(a) - PM_j(b) + P_j(a)}{P_j(a) - Q_j(a)}$$

$$C(a,b) = \frac{\sum w_j \cdot c_j(a,b)}{\sum w_j}$$

where

PM ... performance of alternative in criterion, Q ... indifference threshold, P ... prefference threshold, w ... weights.

Usage

```
Electre3_ConcordanceIndex(PM, P, Q, w)
```

Arguments

`PM`	Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named and criteria are expected to be maximized (you can use function util_prepare_minmax to do that).
`P`	preference threshold vector
`Q`	indefference threshold
`w`	vector containing the weights of the criteria.

Value

Returns computed concordancce matrix.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Roy B. : "The outranking approach and the foundations of ELECTRE methods", Theory and Decision 31, 1991, 49-73.

Vallée, D.; Zielniewicz, P. 1994. ELECTRE III-IV, version 3.x, Aspects Méthodologiques (tome 1), Guide d'utilisation (tome 2). Document du LAMSADE 85 et 85bis, Université Paris Dauphine

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

See Also

ELECTRE III: Electre_3

ELECTRE 1S: Electre_1S

Examples

```
# the performance table
PM <- cbind(
  c(-14,129,-10,44,-14),
  c(90,100,50,90,100),
  c(0,0,0,0,0),
  c(40,0,10,5,20),
  c(100,0,100,20,40)
)
rownames(PM) <- c("Project1","Project2","Project3","Project4","Project5")
colnames(PM) <- c( "CR1","CR2","CR3","CR4","CR5")
Q <- c(25,16,0,12,10) #Indifference thresholds
P <- c(50,24,1,24,20) #Preference thresholds
w <- c(1,1,1,1,1) #weights
v <- Electre3_ConcordanceIndex(PM, P, Q, w)
```

Electre_1 — *Electre_1 : ELECTRE I method used to solve multiple criteria decision making*

Description

The acronym ELECTRE stands for: ELimination Et Choix Traduisant la REalite (ELimination and Choice Expressing REality).ELECTRE I method is then designed to rank reliability design scheme in order of decision maker preference.This method is based on the concept of concordance and discordance.

ELECTRE I does not provide rank, but allows the user to identify so called kernel of the decision - the alternatives, which cannot be eleminated from decision making as obviously inefficient, thus simplifying the decision making problem.

Technically the function implements approach authored by M. Balamurali in pyDecisions (in Python) and reimplements it in R.

Usage

```
Electre_1(PM,
w,
minmaxcriteria,
concordance_threshold = 1,
discordance_threshold = 0)
```

Arguments

`PM` Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.

`w` vector containing the weights of the criteria.

`minmaxcriteria` criteriaMinMax Vector containing the preference direction on each of the criteria. "min" (resp."max") indicates that the criterion has to be minimized (maximized).

`concordance_threshold`
parameter defining concordance threshold . The default value is 1. The user can set a new value between 0 and 1

`discordance_threshold`
parameter defining discordance threshold . The default value is 0. The user can set a new value between 0 and 1.

Value

The function returns a list structured as follows :

`PerformanceMatrix`
A matrix containing the performance table. Each row corresponds to an alternative, and each column to a criterion

`ConcordanceMatrix`
Concordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. For an outranking aSb to be validated, a sufficient majority of criteria should be in favor of this assertion.

`DiscordanceMatrix`
Discordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. The concept of discordance is complementary to the one of (concordance and

represents the discomfort experienced in the choosing of alternative a above alternative b

PreferenceMatrix
: A matrix with computed preferences for the alternatives. Represents domination relation between the alternatives. Provides value 1 where for relation between alternatives where such domination exists

Kernel
: Kernel consist of subset of non-dominated alternatives

Dominated
: Identified clearly dominated alternatives.

GraphResult
: Visualization of the preference matrix using network graph. A -> B means, that A outranks B.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

Examples

```
PM <- cbind(
  c(103000,101300,156400,267400,49900,103600,103000,170100,279700,405000),
  c(171.3,205.3,221.7,230.7,122.6,205.1,178.0,226.0,233.8,265.0),
  c(7.65,7.90,7.90,10.50,8.30,8.20,7.20,9.10,10.90,10.30),
  c(352,203,391,419,120,265,419,419,359,265),
  c(11.6,8.4,8.4,8.6,23.7,8.1,11.4,8.1,7.8,6.0),
  c(88.0,78.3,81.5,64.7,74.1,81.7,77.6,74.7,75.5,74.7),
  c(69.7,73.4,69.0,65.6,76.4,73.6,66.2,71.7,70.9,72.0))
rownames(PM) <- c("CBX16","P205G","P405M","P605S","R4GTL",
  "RCLIO","R21TS","R21TU","R25BA","ALPIN")
colnames(PM) <- c("Prix","Vmax","C120","Coff","Acce","Frei","Brui")
minmaxcriteria <-c("min","max","min","max","min","min","min")
w <- c(0.3,0.1,0.3,0.2,0.1,0.2,0.1)
M <- Electre_1(PM, w, minmaxcriteria,
  concordance_threshold = 0.8,
  discordance_threshold = 0.1)
```

`Electre_1S` *Electre_1S : ELECTRE 1S method used to solve multiple criteria decision making*

Description

Method for supporting multicriteria decision making used to identify so called kernel of solution as set of alternatives which are not dominated by any other alternative. Dominated alternatives can be ommited from decision making as they clearly represent sub-optimal solution for the problem.

Method does not provide ranking.

Computationally the method can be seen as a hybrid of ELECTRE III and I methods. From ELECTRE III it takes concordance matrix computation as it works with fuzzy defined preference (P), indifference (Q) and veto (V) thresholds. From ELECTRE I method it takes procedure to identify kernel of the solution.

Usage

```
Electre_1S(PM, w, Q, P, V, minmaxcriteria = 'max', lambda = 0.5)
```

Arguments

`PM` Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.

`w` vector containing the weights of the criteria.

`Q` vector of indifference thresholds

`P` vector of preference thresholds

`V` vector of veto thresholds

`minmaxcriteria` criteriaMinMax Vector containing the preference direction on each of the criteria. "min" (resp."max") indicates that the criterion has to be minimized (maximized).

`lambda` parameter defining concordance threshold. The default value is 0.5, but the value can be in interval <0.5;1>.

Value

The function returns a list structured as follows :

`PerformanceMatrix`
A matrix containing the performance table. Each row corresponds to an alternative, and each column to a criterion

`ConcordanceMatrix`
Concordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. For an outranking aSb to be validated, a sufficient majority of criteria should be in favor of this assertion.

`DiscordanceIndex`
Discordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. The concept of discordance is complementary to the one of (concordance and represents the discomfort experienced in the choosing of alternative a above alternative b

`CredibilityIndex`
basically preference matrix as computed by ELECTRE I method. Represents domination relation between the alternatives. Provides value 1 where for relation between alternatives where such domination exists

`Kernel`	Kernel consist of subset of non-dominated alternatives
`Dominated`	Identified clearly dominated alternatives.
`graph`	Visualization of the preference matrix using network graph. A -> B means, that A outranks B.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Examples

```
# the performance table
PM <- cbind(
  c(-14,129,-10,44,-14),
  c(90,100,50,90,100),
  c(0,0,0,0,0),
  c(40,0,10,5,20),
  c(100,0,100,20,40)
)
rownames(PM) <- c("Project1","Project2","Project3","Project4","Project5")
colnames(PM) <- c( "CR1","CR2","CR3","CR4","CR5")
minmaxcriteria <- 'max'
Q <- c(25,16,0,12,10) #Indifference thresholds
P <- c(50,24,1,24,20) #Preference thresholds
V <- c(100,60,2,48,90) #Veto thresholds
w <- c(1,1,1,1,1) #weights
Electre_1S(PM, w, Q, P, V, minmaxcriteria)
```

`Electre_2` *Electre_2 : ELECTRE II method used to solve multiple criteria decision making*

Description

The acronym ELECTRE stands for: ELimination Et Choix Traduisant la REalite (ELimination and Choice Expressing REality). ELECTRE II method is then designed for ranking purposes as oposed to ELECTRE I method which is useful for identification of kernel - as alternatives, which cannot be eliminated, are not obviously dominated by other alternatives.

ELECTRE II uses concordance and discordance indexes to construct two partial pre-orders by trying do formally describe so called strong and weak preferences between the alternatives. Strenght is this regard is taken as strength of belief, that alternative a is better then alternative b.

Two partial preorders and then used in aggregation procedure to construct final patrial preorder.

The code is partially inspired by code authored by M. Balamurali in pyDecisions (in Python) and reimplements it in R, thou some portions of the function are solved differently in here. For example whole graph simplification process in here relies on functionality of iGraph package. This method also differently implements the final recommendation (total partial order).

Usage

```
Electre_2(PM,
  w,
  minmaxcriteria = 'max',
  c_minus = 0.65,
  c_zero = 0.75,
  c_plus = 0.85,
  d_minus = 0.25,
  d_plus = 0.5)
```

Arguments

`PM`	Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.
`w`	vector containing the weights of the criteria.
`minmaxcriteria`	criteriaMinMax Vector containing the preference direction on each of the criteria. "min" (resp."max") indicates that the criterion has to be minimized (maximized).
`c_minus`	first of three concordance threshold parameters. ELECTRE II method implements the concotdance threshold as fuzzy triangle. Value is alway between 0-1 and 0 <= c- <= c0 <= c+ <= 1, default 0.65.
`c_zero`	second of three concordance threshold parameters. ELECTRE II method implements the concotdance threshold as fuzzy triangle. Value is alway between 0-1 and 0 <= c- <= c0 <= c+ <= 1, default 0.75.
`c_plus`	third of three concordance threshold parameters. ELECTRE II method implements the concotdance threshold as fuzzy triangle. Value is alway between 0-1 and 0 <= c- <= c0 <= c+ <= 1, degault 0.85.
`d_minus`	first of two parameters defining discordance threshold. Thershold is defined as range, where 0 <= d- <= d+ <= 1,default value 0.25.
`d_plus`	first of two parameters defining discordance threshold. Thershold is defined as range, where 0 <= d- <= d+ <= 1,default value 0.25.

Value

The function returns a list structured as follows:

`PerformanceMatrix`
: A matrix containing the performance table. Each row corresponds to an alternative, and each column to a criterion

`ConcordanceMatrix`
: Concordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. For an outranking aSb to be validated, a sufficient majority of criteria should be in favor of this assertion.

`DiscordanceMatrix`
: Discordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. The concept of discordance is complementary to the one of (concordance and represents the discomfort experienced in the choosing of alternative a above alternative b

`StrongOutranking`
: A graph, expresed as adjancancy matrix of alternative for which it was possible to establish strong outranking relation. (We strongly believe that alternative a is better then alternative b: aSFb).

`WeakOutranking`
: A graph, expressed as adjancancy matrix of alternatives for which it was possible to establish weak outranking relation. Similar to strong outranking, only our belief of a outranking b is much weaker: aSfb.

`firstTotalPreorder`
: first total preorder V1

`secondTotalPreorder`
: second total preorder V2

`finalPreorderMatrix`
: final partial preorder expressed as adjancancy matrix.

`incomparableAlternatives`
: adjacancy matrix identifying the alternatives, which cannot be directly compared (are incomparable).

`graphResult`
: plot of finalPreorderMatrix

`finalPreorder`
: ordered vector of alternatives from best to worst, with identification of how many times was the alternative preffered to other alternatives.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

Examples

```
PM <- cbind(
  c(103000,101300,156400,267400,49900,103600,103000,170100,279700,405000),
  c(171.3,205.3,221.7,230.7,122.6,205.1,178.0,226.0,233.8,265.0),
  c(7.65,7.90,7.90,10.50,8.30,8.20,7.20,9.10,10.90,10.30),
```

```
  c(352,203,391,419,120,265,419,419,359,265),
  c(11.6,8.4,8.4,8.6,23.7,8.1,11.4,8.1,7.8,6.0),
  c(88.0,78.3,81.5,64.7,74.1,81.7,77.6,74.7,75.5,74.7),
  c(69.7,73.4,69.0,65.6,76.4,73.6,66.2,71.7,70.9,72.0))
rownames(PM) <- c("CBX16","P205G","P405M","P605S",
  "R4GTL","RCLIO","R21TS","R21TU","R25BA","ALPIN")
colnames(PM) <- c("Prix","Vmax","C120","Coff","Acce","Frei","Brui")
minmaxcriteria <-c("min","max","min","max","min","min","min")
w <- c(0.3,0.1,0.3,0.2,0.1,0.2,0.1)
M <- Electre_2(PM, w, minmaxcriteria)
```

`Electre_3` *ELECTRE III method for ranking alternatives*

Description

ELECTRE III method aims to answer the following question: considering a finite set of actions, A, evaluated on a coherent family of pseudo-criteria, F, how to make a partition of A in classes of quivalence and provide a necessarily complete pre-order expressing the relative position of these classes? In the first phase, ELECTRE III method involves the construction of a fuzzy outranking relation.

In the second phase, an algorithm is used for making a ranking in a final partial pre-order, that combines two complete pre-orders.

This implementation presents simplified version of the method using single value alpha, beta parameters. If more complex approach to ELECTRE III computation is required use OutrankingTools package for R, which provides variant allowing such computations.

Usage

```
Electre_3(PM, w, P, Q, V, minmaxcriteria = 'max', alpha = 0.3,
beta = 0.15, VERBOSE = F)
```

Arguments

`PM`	Performance matrix as matrix or data frame containing the performance table. Data frame with named columns (criteria) and rows (alternatives) is prefered.
`w`	vector of weights. Number of weights must be equal to number of criteria used in decision making.
`P`	Preference threshold - Vector containing the preference thresholds constraints defined for each criterion.
`Q`	Indifference threshold - Vector containing the indifference thresholds constraints defined for each criterion.
`V`	Veto threshold - Vector containing the veto thresholds constraints defined for each criterion.
`minmaxcriteria`	Vector containing the preference direction on each of the criteria either min or max. If all criteria are to be minimized or maximized, single min resp. max value can be used.
`alpha`	alpha and beta coefficients are used in downward and upward distilation procedure as wel as final ranking procerude to construct orders. Both coef. are used iteratively change thresholds limiting evaluation of the outranking relations between the alternatives.

beta — see alpha. Preset values of alpha = 0.3 and beta = 0.15 are set as per Vallee and Zielniewicz (1994). This version of method allows only to set single value for the purpose.

VERBOSE — Boolean value switching on/off verbose mode. If switched on method provides broder set of information on computations, which can be usefel when exploring some detected anomalies in recomendations or as part of study of the ELECTRE III behavior.

Value

The function returns a list structured as follows:

PM — Performance Matrix - A matrix containing the performance table. Each row corresponds to an alternative, and each column to a criterion

cm — Concordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. For an outranking aSb to be validated, a sufficient majority of criteria should be in favor of this assertion.

DiscordanceMatrixCriteria — Discordance matrix is one of two working relations (concordance and discordance) which are subsequently used to construct the final dominance relation. The concept of discordance is complementary to the one of (concordance and represents the discomfort experienced in the choosing of alternative a above alternative b. In comparison with ELECTRE I or II methods discordance matrix for ELECTRE III is computed separately for every criterion.

CredibilityMatrix — matrix assessing the strength of the assertion that a is at least as good as b.

rank_D — Descending distillation ranking - final partial preorder orders the alternatives from the best to the worst.

rank_A — Ascending distillation ranking - final partial preorder orders the alternatives from worst to best.

rank_P — Pre-order matrix specifying identified relations between the alternatives - values are P+ (a prefered to b), P- (b prefered to a), I (indifferent), R (incomparable)

adjancancyMatrix — Adjancency Matrix allows to visualize results as network diagram

graph — processed adajncency matrix into network digram.

final_ranking — final order of the alternatives.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Roy B. : "The outranking approach and the foundations of ELECTRE methods", Theory and Decision 31, 1991, 49-73.

Vallée, D.; Zielniewicz, P. 1994. ELECTRE III-IV, version 3.x, Aspects Méthodologiques (tome 1), Guide d'utilisation (tome 2). Document du LAMSADE 85 et 85bis, Université Paris Dauphine

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

Examples

```
# the performance table
PM <- cbind(
  c(-14,129,-10,44,-14),
  c(90,100,50,90,100),
  c(0,0,0,0,0),
  c(40,0,10,5,20),
  c(100,0,100,20,40)
)
rownames(PM) <- c("Project1","Project2","Project3","Project4","Project5")
colnames(PM) <- c( "CR1","CR2","CR3","CR4","CR5")
minmaxcriteria <- 'max'
Q <- c(25,16,0,12,10) #Indifference thresholds
P <- c(50,24,1,24,20) #Preference thresholds
V <- c(100,60,2,48,90) #Veto thresholds
w <- c(1,1,1,1,1) #weights
Electre_3(PM, w, P, Q, V, minmaxcriteria)
```

Electre_4 — *ELECTRE IV method for ranking alternatives*

Description

ELECTRE IV is alternative ranking approach based on ELECTRE III method, but without using any weighting of the criteria. Since weighting is required for concordance and discordance matrixes, these are not available in ELECTRE IV. Instead method uses more complex system of outranking relations:

- mp(b,a) - number of criteria for which option b is strictly preferred to a
- mq(b,a) - number of criteria for which option b is weakly preferred to a
- mj(b,a) - number of criteria for which option b is judged indifferent to a
- mo(b,a) = mo(a,b) - number of criteria on which options a and b perform identically

These are used as a base for computation of credibility matrix.Reamining computations are same as for ELECTRE III methods including construction of descending and ascending pre-order and deriving final outranking relation.

Usage

```
Electre_4(PM, P, Q, V, minmaxcriteria = 'max')
```

Arguments

PM
: Performance matrix as matrix or data frame containing the performance table. Data frame with named columns (criteria) and rows (alternatives) is prefered.

P
: Preference threshold - Vector containing the preference thresholds constraints defined for each criterion.

Q
: Indifference threshold - Vector containing the indifference thresholds constraints defined for each criterion.

V
: Veto threshold - Vector containing the veto thresholds constraints defined for each criterion.

minmaxcriteria
: Vector containing the preference direction on each of the criteria ("min" or "max"). If all criteria are to be minimized or maximized, single min resp. max value can be used.

Value

The function returns a list structured as follows:

PM
: Performance Matrix - A matrix containing the performance table. Each row corresponds to an alternative, and each column to a criterion

CredibilityMatrix
: matrix assessing the strength of the assertion that “a is at least as good as b".

rank_D
: Descending distillation ranking - final partial preorder orders the alternatives from the best to the worst.

rank_A
: Ascending distillation ranking - final partial preorder orders the alternatives from worst to best.

rank_P
: Pre-order matrix specifying identified relations between the alternatives - values are P+ (a prefered to b), P- (b prefered to a), I (indifferent), R (incomparable)

adjancancyMatrix
: Adjancency Matrix allows to visualize results as network diagram

graph
: processed adajncency matrix into network digram.

final_ranking
: final order of the alternatives.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

Roy B. : "The outranking approach and the foundations of ELECTRE methods", Theory and Decision 31, 1991, 49-73.

Vallée, D.; Zielniewicz, P. 1994. ELECTRE III-IV, version 3.x, Aspects Méthodologiques (tome 1), Guide d'utilisation (tome 2). Document du LAMSADE 85 et 85bis, Université Paris Dauphine

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

Examples

```
# the performance table
PM <- cbind(
  c(-14,129,-10,44,-14),
  c(90,100,50,90,100),
  c(0,0,0,0,0),
  c(40,0,10,5,20),
  c(100,0,100,20,40)
)
rownames(PM) <- c("Project1","Project2","Project3","Project4","Project5")
colnames(PM) <- c( "CR1","CR2","CR3","CR4","CR5")
minmaxcriteria <- 'max'
Q <- c(25,16,0,12,10) #Indifference thresholds
P <- c(50,24,1,24,20) #Preference thresholds
V <- c(100,60,2,48,90) #Veto thresholds
Electre_4(PM, P, Q, V, minmaxcriteria)
```

Electre_asc_dist   *Electre_asc_dist: algrithm for ascending distilation*

Description

Algorithm to establish partial preorder by the means of ascending distilation. Preorder is achieved by distilling alternatives using progresively lower cut-off thresholds.

This aproach is complementary to descending distilation process which creates second pre-order.

The algorithm is used in Electre_3 and 4 methods.

Usage

```
Electre_asc_dist(sm)
```

Arguments

sm	confidence matrix

Value

returns list of alternatives in ranks from worst-to best.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

ELECTRE_ConcordanceMatrix

ELECTRE_ConcordanceMatrix : Method used compute concordance matrix

Description

Internal method for computing concordance matrix for ELECTRE I and II methods. Concordance matrix is one of two angles ELECTRE methods use to derive preference for the alternatives, the other being Discordance matrix.

Concordancce matrix (index) measures strength of the statement that alternative a outranks alternative b, while discordance matrix (index) together with discordance threshold (exceeding this threshold) can prevent such outranking.

Code is inspired by pyDecisions package.

Computationally concoradce matrix C(a,b) is defined as:

$$C(a,b) = \frac{1}{W} \sum_{\forall j: g_j(a) \geq g_j(b)} w_j$$

where

$$W = \sum_{j=1}^{n} w_j$$

Where gj(x) ... performance of alternative x in criterium j, wj ... weight of criterium j.

Usage

```
ELECTRE_ConcordanceMatrix(PM, w)
```

Arguments

`PM`	Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named and criteria are expected to be maximized (you can use function util_pm_minmax to do that).
`w`	vector containing the weights of the criteria.

Value

Returns computed concordancce matrix.

Author(s)

Pavel Šenovský `<pavel.senovsky@vsb.cz>`

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

See Also

ELECTRE I: Electre_1

discorance matrix: ELECTRE_DiscordanceMatrix

Examples

```
PM <- cbind(
  c(103000,101300,156400,267400,49900,103600,103000,170100,279700,405000),
  c(171.3,205.3,221.7,230.7,122.6,205.1,178.0,226.0,233.8,265.0),
  c(7.65,7.90,7.90,10.50,8.30,8.20,7.20,9.10,10.90,10.30),
  c(352,203,391,419,120,265,419,419,359,265),
  c(11.6,8.4,8.4,8.6,23.7,8.1,11.4,8.1,7.8,6.0),
  c(88.0,78.3,81.5,64.7,74.1,81.7,77.6,74.7,75.5,74.7),
  c(69.7,73.4,69.0,65.6,76.4,73.6,66.2,71.7,70.9,72.0))
rownames(PM) <- c("CBX16","P205G","P405M","P605S",
  "R4GTL","RCLIO","R21TS","R21TU","R25BA","ALPIN")
colnames(PM) <- c("Prix","Vmax","C120","Coff","Acce","Frei","Brui")
minmaxcriteria <-c("min","max","min","max","min","min","min")
w <- c(0.3,0.1,0.3,0.2,0.1,0.2,0.1)
PMmax <- util_pm_minmax(PM, minmaxcriteria)
cm <- ELECTRE_ConcordanceMatrix(PMmax, w)
```

Electre_desc_dist *Electre_desc_dist - algrithm for descending distilation*

Description

Algorithm to establish partial preorder by the means of descending distilation. Preorder is achieved by distilling alternatives using progresively lower cutoff thresholds.

This aproach is complementary to ascending distilation process which creates second preorder.

The algorithm is used in Electre_3 and 4 methods.

Usage

```
Electre_desc_dist(sm)
```

Arguments

sm	confidence matrix

Value

returns list of alternatives in ranks from worst to best.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

`ELECTRE_DiscordanceMatrix`

ELECTRE_DiscordanceMatrix : Method used compute discordance matrix

---

Description

Internal method for computing discordance matrix for ELECTRE I and II methods. Discordance matrix is one of two angles ELECTRE methods use to derive preference for the alternatives, the other being concordance matrix.

Concordancce matrix (index) measures strength of the statement that alternative a outranks alternative b, while discordance matrix (index) together with discordance threshold (exceeding this threshold) can prevent such outranking.

Code is inspired by pyDecisions package.

Computationally discoradce matrix D(a,b) is defined for:

$$g_j(a) \geq g_j(b) \forall j : D(a,b) = 0$$

and for everything else:

$$D(a,b) = \max_j \frac{g_j(b) - g_j(a)}{\delta_j}$$

where

$$\delta_j = \max c, d, j g_j(c) - g_j(d)$$

Where gj(x) ... performance of alternative x in criterium j.

Usage

```
ELECTRE_DiscordanceMatrix(PM)
```

Arguments

`PM`	Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named and criteria are expected to be maximized (you can use function util_prepare_minmax to do that).

Value

Returns computed discordancce matrix.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Balamurali, M.: pyDecisions - A Python Library of management decision making techniques. Avilable on-line from https://github.com/Valdecy/pyDecisions

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

See Also

ELECTRE I: Electre_1

concorance matrix: ELECTRE_ConcordanceMatrix

Examples

```
PM <- cbind(
  c(103000,101300,156400,267400,49900,103600,103000,170100,279700,405000),
  c(171.3,205.3,221.7,230.7,122.6,205.1,178.0,226.0,233.8,265.0),
  c(7.65,7.90,7.90,10.50,8.30,8.20,7.20,9.10,10.90,10.30),
  c(352,203,391,419,120,265,419,419,359,265),
  c(11.6,8.4,8.4,8.6,23.7,8.1,11.4,8.1,7.8,6.0),
  c(88.0,78.3,81.5,64.7,74.1,81.7,77.6,74.7,75.5,74.7),
  c(69.7,73.4,69.0,65.6,76.4,73.6,66.2,71.7,70.9,72.0))
rownames(PM) <- c("CBX16","P205G","P405M","P605S",
  "R4GTL","RCLIO","R21TS","R21TU","R25BA","ALPIN")
colnames(PM) <- c("Prix","Vmax","C120","Coff","Acce","Frei","Brui")
minmaxcriteria <-c("min","max","min","max","min","min","min")
PMmax <- util_pm_minmax(PM, minmaxcriteria)
dm <- ELECTRE_DiscordanceMatrix(PMmax)
```

Electre_TRI *ELECTRE TRI Method*

Description

The Electre Tri is a multiple criteria decision aiding method, designed to deal with sorting problems. Electre Tri method has been developed by LAMSADE (Paris-Dauphine University, Paris, France).

The method itself is very interesting as it does not directly compare alternatives against each other. The evaluation is performed against the "profile". The profile presents scale in which criterion exists and is divided in the categories. Since we presume imperfect information we are basically evaluating in which category the performance of the alternative in criterion is.

These categories are then aggregated across the criteria to form rank using "pessimistic" or "optimistic" agregation procedure. Results of these procedures is then used to form ranking.

As oposed to ELECTRE IV, ELECTRE TRI does not provide guidance to establish "final" ranking by agregating outcomes of optimistic and pesimistic procedures.

Usage

```
Electre_TRI(PM, profiles, profiles_names,
  w, Q, P, V,
  minmaxcriteria = 'max', lambda=0.75)
```

Arguments

`PM` Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. Rows (resp. columns) must be named according to the IDs of the alternatives (resp. criteria).

`profiles` Matrix containing, in each row, the lower profiles of the categories. The columns are named according to the criteria, and the rows are named according to the categories. The index of the row in the matrix corresponds to the rank of the category.

`profiles_names` Vector containing profiles'names

`w` Vector containing the weights of the criteria.

`Q` Vector containing the indifference thresholds constraints defined for each criterion.

`P` Vector containing the preference thresholds constraints defined for each criterion.

`V` Vector containing the veto thresholds constraints defined for each criterion

`minmaxcriteria` Vector containing the preference direction on each of the criteria. "min" (resp."max") indicates that the criterion has to be minimized (maximized). If all criteria are directed in same way use single min or max instead (the method prepars required parameters on its own)

`lambda` Lambda parameter is used as cut-off criterion for outranking evaluation. Should be in range of 0.5 and 1.0. Default value=0.75

Value

The function returns a list structured as follows:

`performanceMatrix`
A matrix containing the performance table. Each row corresponds to an alternative, and each column to a criterion. Criterion direction is recomputed to represent maximize direction.

`partialConcordanceIndex`
Partial concordance index indicates the relative dominance of one option over profile aSbh where bh is profile value.Partial concordance index is being constructed separately for each criterion.

`partialConcordanceIndexInverse`
Same as partialConcordanceIndex but evaluates bhSa: dominance of profile value over performance of the alternative in criterion.

`overalConcordanceIndex`
Consolidates information from partial concordance indexes into single matrix describing dominance of the alternative over the profile aSbh across the criteria

`overallConcordanceIndexInverse`
same as overallConcordanceIndex, but describes bhSa instead, again across all criteria

`discordanceIndex`
: discordance index is oposite to concordance index. It is a value (matrix of values) used to establish that a!Sbh (a does not dominate bh). Discordance index is being computed separately for each criterion.

`discordanceIndexInverse`
: Same as discordanceIndex but helps evaluate bhSa.

`credibilityIndex`
: represents a degree of credibility of the assertion that a outranks bh (aSbh).

`credibilityIndexInverse`
: represents a degree of credibility of the assertion that bh outranks a (bhSa).

`preferenceRelation`
: determination of preference situation between alternatives (aSbh = ">", bhSa = "<", I = indifferent, R = incomparable).

`pessimistic`
: The direction of the ranking obtained from the pessimistic procedure is from best to worst.

`optimistic`
: the direction of the ranking obtained from optimistic procesure goes from worst to best.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Rogers, Martin and Myastre, Lucien-Yves. ELECTRE and Decision Support: Methods and Applications in Engineering and Infrastructure investment. Springer 2000, 208 p., ISBN 978-1-4757-5057-7

FIGUEIRA, J.R., Greco, S., Roy, B., Slowinski, R. ELECTRE Methods : Main Features and Recent Developments. Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision, Paris: 2010, 34 pp., available from https://hal.archives-ouvertes.fr/hal-00876980/document.

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

Shofade, Olanrewaju Joseph Soniran. Considering hierarchical structure of criteria in ELECTRE decision aiding methods. Universitat Rovira i Virgili, Escola Tecnica Superior d'Enginyeria, Tarragona: 2011 https://deim.urv.cat/~itaka/itaka2/PDF/acabats/ThesisJoseph-ELECTRE-H.pdf

Examples

```
# the performance table
PM <- cbind(
  c(-120.0,-150.0,-100.0,-60,-30.0,-80,-45.0),
  c(-284.0,-269.0,-413.0,-596,-1321.0,-734,-982.0),
  c(5.0,2.0,4.0,6,8.0,5,7.0),
  c(3.5,4.5,5.5,8,7.5,4,8.5),
  c(18.0,24.0,17.0,20,16.0,21,13.0)
)
# names of alternatives
rownames(PM) <- c("a1","a2","a3","a4","a5","a6","a7")
# names of criteria
colnames(PM) <- c( "g1","g2","g3","g4","g5")
w <- c(0.25,0.45,0.10,0.12,0.08) # criteria weights
```

```
# all criteria maxed - ommiting minmaxcriteria parameter
# lambda = 0.75 - ommiting lambda parameter
# Matrix containing the profiles.
profiles <- cbind(c(-100,-50), c(-1000,-500),
  c(4,7),c(4,7),c(15,20))
#  vector defining profiles' names
profiles_names <-c("b1","b2")
# thresholds vector
I <- c(15,80,1,0.5,1) # indifference threshold
P <- c(40,350,3,3.5,5) # prefference threshold
V <- c(100,850,5,4.5,8) # veto threshold
Electre_TRI(PM,
profiles, profiles_names,
w, I, P, V)
```

---

finalRanking *finalRanking - computes final ranking based on pre-order matrix*

Description

Takes pre-order matrix and computes final ranking from it

Usage

```
finalRanking(alt, rank_P)
```

Arguments

alt	alternatives
rank_P	pre-order matrix

Value

returns data frame with final ranking..

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

FuzzyTOPSIS *FuzzyTOPSIS : method used to solve multiple criteria decision making*

Description

The acronym TOPSIS stands for: Technique of Order Preference Similarity to the Ideal Solution. As name suggests TOPSIS provides its guidance based on evaluation of the similarity to both ideal and anti-ideal variant of the solution.

Original method uses 5 steps for the procedure. In first step the procedure normalizes values in performance matrix nad applies weights to it in step 2. In step 3 ideal variant A^* and anti-ideal variant A^- is computed as maximums and minimums of the criteria in performance matrix.

In step 4 distance to ideal D^* and anti-ideal variant D^- is computed and in step 5 used to compute closenes criterium (CC).

Criterium CC is then directly usable to rank alternatives. CC is always in interval of 0-1, the closer the value is to 1, the closer it is to ideal variant.

The approach described above is same as for TOPSIS method, thou in case of FuzzyTOPSIS triangular fuzzy numbers are used to describe the values in both criteria weights and performance matrix.

Note: in present version of the function only benefit criteria are being supported.

Usage

```
FuzzyTOPSIS(PM, dictionaryPM, w, dictionaryW, alt)
```

Arguments

`PM`	performance matrix with n columns and no. criteria x no. of alternatives rows. I. e. with 3 criteria and 2 alternatives the rows are: 1: 1.cri-1.alt., 2: 1.cri.-2.alt, 3: 2.cri-1.alt, 4: 2.cri-2.alt, ...
`dictionaryPM`	dictionary of linguistic variables for criteria
`w`	weights (matrix of weights - decision makers in columns and criteria in rows)
`dictionaryW`	dictionary for wights (single matrix|dataframe)
`alt`	vector with names of the alternatives

Value

The function returns a list structured as follows:

`fuzzy_weights`	fuzzy weights of the criteria
`fuzzy_decision_matrix`	fuzzy decision matrix of the criteria
`fuzzy_norm_decision_matrix`	fuzzy normalized decision matrix
`weighted_fuzzy_norm_decision_matrix`	weighted fuzzy normalized decision matrix
`a_plus`	ideal solution
`a_minus`	anti-ideal solution
`CC`	list of alternatives with computed closeness criterion [0-1], the closer to 1, the closer to ideal solution

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Papathanasiou, Jason, Ploskas, Nikolaos. Multiple Criteria Decision Aid Methods, Examples and Python Implementations. Springer, 173 p., ISBN 978-3-319-91648-4.

Examples

```
dictionaryW <- rbind(
  c(0, 0, 0.1),
  c(0, 0.1, 0.3),
  c(0.1, 0.3, 0.5),
  c(0.3, 0.5, 0.7),
  c(0.5, 0.7, 0.9),
  c(0.7, 0.9, 1),
  c(0.9, 1, 1)
)
#V very, L low, M medium, H high
rownames(dictionaryW) <- c('VL', 'L', 'ML', 'M', 'MH', 'H', 'VH')
w <- rbind(
  c('H', 'VH', 'VH'),
  c('M', 'H', 'VH'),
  c('M', 'MH', 'ML'),
  c('H', 'VH', 'MH')
)
rownames(w) <- c('Investment cost', 'Employment needs', 'Social impact',
  'Environmental impact')
dictionaryPM <- rbind(
  c(0,0,1),
  c(0,1,3),
  c(1,3,5),
  c(3,5,7),
  c(5,7,9),
  c(7,9,10),
  c(9,10,10)
)
#V very, P poor, M medium, F fair, G good
rownames(dictionaryPM) <- c('VP', 'P', 'MP', 'F', 'MG', 'G', 'VG')
PM <- rbind(
  c('VG', 'G', 'MG'),
  c('MP', 'F', 'F'),
  c('MG', 'MP', 'F'),
  c('MG', 'MG', 'VG'),
  c('VP', 'P', 'G'),
  c('F', 'G', 'G'),
  c('F', 'MG', 'MG'),
  c('F', 'VG', 'G'),
  c('MG', 'MG', 'VG'),
  c('G', 'G', 'VG'),
  c('P', 'VP', 'MP'),
  c('F', 'MP', 'MG'),
  c('P', 'P', 'MP'),
  c('MG', 'VG', 'G'),
  c('MP', 'F', 'F'),
```

```
  c('MG', 'VG', 'G'),
  c('G', 'G', 'VG'),
  c('VG', 'MG', 'F'),
  c('G', 'VG', 'G'),
  c('MG', 'F', 'MP'),
  c('MP', 'P', 'P'),
  c('VP', 'F', 'P'),
  c('G', 'MG', 'MG'),
  c('P', 'MP', 'F')
)
alternatives <- c('site 1', 'site 2', 'site 3', 'site 4', 'site 5', 'site 6')
result <- FuzzyTOPSIS(PM, dictionaryPM, w, dictionaryW, alternatives)
```

FuzzyVIKOR   *FuzzyVIKOR : method used to solve multiple criteria decision making*

Description

The acronym VIKOR stands for: VlseKriterijumska Optimizacija I Kompromisno Resenje, in Serbian multicriteria optimization and compromise solution. The method has been especially designed to deal with problematic situations when the alternatives are characterized by non-commensurable and conflicting criteria, for which VIKOR provides compromise solution. Methodologically VIKOR is close to another method TOPSIS. Original VIKOR uses five steps to derive such compromise solution.

FuzzyVIKOR uses trapezoidal fuzzy number for computation purposes. Other then using fuzzy number for computation, the results are basically the same: S, R and Q, which can be used to compute compromise solution. Look in the VIKOR function description to better understand the computation proces (or the source code).

At present time FuzzyVIKOR function does not have implemented establishment of compromise solution.

Usage

```
FuzzyVIKOR(PM, dictionaryPM, w, dictionaryW, alt)
```

Arguments

PM	performance matrix with n columns and no. criteria x no. of alternatives rows. I. e. with 3 criteria and 2 alternatives the rows are: 1: 1.cri-1.alt., 2: 1.cri.-2.alt, 3: 2.cri-1.alt, 4: 2.cri-2.alt, ...
dictionaryPM	dictionary of linguistic variables for criteria
w	weights (matrix of weights - decision makers in columns and criteria in rows)
dictionaryW	dictionary for wights (single matrix|dataframe)
alt	vector with names of the alternatives

Value

The function returns a list structured as follows :

S	list of alternatives using S-metric
R	list of alternatives using R-metric
Q	list of alternatives using Q-metric

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

Papathanasiou, Jason, Ploskas, Nikolaos. Multiple Criteria Decision Aid Methods, Examples and Python Implementations. Springer, 173 p., ISBN 978-3-319-91648-4.

Examples

```
dictionaryW <- rbind(
  c(0, 0, 0.1, 0.2),
  c(0.1, 0.2, 0.2, 0.3),
  c(0.2, 0.3, 0.4, 0.5),
  c(0.4, 0.5, 0.5, 0.6),
  c(0.5, 0.6, 0.7, 0.8),
  c(0.7, 0.8, 0.8, 0.9),
  c(0.8, 0.9, 1, 1)
)
#V very, L low, M medium, H high
rownames(dictionaryW) <- c('VL', 'L', 'ML', 'M', 'MH', 'H', 'VH')
w <- rbind(
  c('H', 'VH', 'VH'),
  c('M', 'H', 'VH'),
  c('M', 'MH', 'ML'),
  c('H', 'VH', 'MH')
)
rownames(w) <- c('Investment cost', 'Employment needs', 'Social impact',
                 'Environmental impact')
dictionaryPM <- rbind(
  c(0, 0, 0.1, 0.2),
  c(0.1, 0.2, 0.2, 0.3),
  c(0.2, 0.3, 0.4, 0.5),
  c(0.4, 0.5, 0.5, 0.6),
  c(0.5, 0.6, 0.7, 0.8),
  c(0.7, 0.8, 0.8, 0.9),
  c(0.8, 0.9, 1, 1)
)
#V very, P poor, M medium, F fair, G good
rownames(dictionaryPM) <- c('VP', 'P', 'MP', 'F', 'MG', 'G', 'VG')
PM <- rbind(
  c('VG', 'G', 'MG'),
  c('MP', 'F', 'F'),
  c('MG', 'MP', 'F'),
  c('MG', 'MG', 'VG'),
  c('VP', 'P', 'G'),
  c('F', 'G', 'G'),
  c('F', 'MG', 'MG'),
  c('F', 'VG', 'G'),
  c('MG', 'MG', 'VG'),
  c('G', 'G', 'VG'),
  c('P', 'VP', 'MP'),
  c('F', 'MP', 'MG'),
  c('P', 'P', 'MP'),
```

```
  c('MG', 'VG', 'G'),
  c('MP', 'F', 'F'),
  c('MG', 'VG', 'G'),
  c('G', 'G', 'VG'),
  c('VG', 'MG', 'F'),
  c('G', 'VG', 'G'),
  c('MG', 'F', 'MP'),
  c('MP', 'P', 'P'),
  c('VP', 'F', 'P'),
  c('G', 'MG', 'MG'),
  c('P', 'MP', 'F')
)
alternatives <- c('site 1', 'site 2', 'site 3', 'site 4', 'site 5', 'site 6')
result <- FuzzyVIKOR(PM, dictionaryPM, w, dictionaryW, alternatives)
```

Graph2AdjancancyMatrix *Graph2AdjancancyMatrix : Transforms graph (a->b format) to adjncancy matrix*

Description

Function transforms back graph specified by edges to adjancancy matrix, with 1 where edge exists and 0 where it does not.

Usage

```
Graph2AdjancancyMatrix(G, alt)
```

Arguments

G	Graph which should be transformed
alt	alternatives names

Value

Returns adjancancy matrix for the graph with 0/1 representing graph.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

mcda_del_dominated *mcda_del_dominated : Deletes dominated alternatives from preference matrix*

Description

Deletes dominated alternatives from preference matrix (should such alternatives exist in the matrix) and return the result.

Usage

```
mcda_del_dominated(M, minmaxcriteria, digits)
```

Arguments

M	Normalized performance matrix - criteria in columns and alternatives in rows. Expected, that all criteria are to be maximized.
minmaxcriteria	either value (min or max) or vector providing guidance for orientation of the criteria in preference matrix. If parameter provides only single max or min value, the function will presume usige of this orientation for all criteria. Default value is 'max'.
digits	number of digits the resulting matrix cells should be rounded to. Default value is 2 (round to 2 digits).

Value

Returns matrix containing only such alternatives, which were not clearly dominated by some other alternative.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

De Brouwer, Philippe J. S.: "The The Big R-Book: From Data Science to Learning Machines and Big Data ", Wiley, 2020, 928 p., ISBN 978-1119632726.

Examples

```
alternatives <- c('BLR', 'BOM', 'DEL', 'MNL', 'HYD', 'GRU', 'DUB', 'KRK', 'MAA', 'EZE')
criteria <- c('tlnt', 'stab', 'cost', 'infl', 'tm-zn', 'infr', "life")
M <- rbind(
  c(0.8181818, 0.1814159, 1.0000000, 0.1198582, 0, 0.6, 0.750),
  c(1.0000000, 0.1814159, 0.6666667, 0.1198582, 0, 0.6, 0.375),
  c(1.0000000, 0.1814159, 0.8333333, 0.1198582, 0, 0.6, 0.125),
  c(0.8181818, 0.0000000, 1.0000000, 0.3482143, 0, 0.6, 0.375),
  c(0.1818182, 0.1814159, 1.0000000, 0.1198582, 0, 0.2, 0.375),
  c(0.1818182, 0.1814159, 0.5000000, 0.1198582, 0, 0.2, 0.125),
  c(0.0000000, 1.0000000, 0.0000000, 0.5741667, 1, 1.0, 1.000),
  c(0.3636364, 0.7787611, 0.6666667, 1.0000000, 1, 0.0, 0.500),
  c(0.4545455, 0.1814159, 0.9166667, 0.1198582, 0, 0.4, 0.000),
```

```
  c(0.1818182, 0.6283186, 0.5833333, 0.0000000, 0, 0.4, 0.125)
)
rownames(M) <- alternatives
colnames(M) <- criteria
mcda_del_dominated(M)
```

`mcda_get_dominated` *mcda_get_dominated : Identify alternatives, which clearly dominate others*

Description

Takes normalized performance matrix (criteria in columns, alternatives in rows) and identifies the alternatives, that are being dominated by other alternatives. Dominated alternatives are prime candidates for deletion and thus simplification of decision problem.

Function returns matrix m to m with 1 identifying alternative being dominated.

Usage

```
mcda_get_dominated(M, minmaxcriteria = 'max')
```

Arguments

`M`	Normalized performance matrix - criteria in columns and alternatives in rows.
`minmaxcriteria`	either value (min or max) or vector providing guidance for orientation of the criteria in preference matrix. If parameter provides only single max or min value, the function will presume usige of this orientation for all criteria.

Value

Returns matrix m to m with 1 identifying alternative being dominated. 1 in position ij if alternative i dominates alternative j.

Author(s)

Pavel Šenovský `<pavel.senovsky@vsb.cz>`

References

De Brouwer, Philippe J. S.: "The The Big R-Book: From Data Science to Learning Machines and Big Data ", Wiley, 2020, 928 p., ISBN 978-1119632726.

Examples

```
alternatives <- c('BLR', 'BOM', 'DEL', 'MNL', 'HYD', 'GRU', 'DUB', 'KRK', 'MAA', 'EZE')
criteria <- c('tlnt', 'stab', 'cost', 'infl', 'tm-zn', 'infr', "life")
M <- rbind(
  c(0.8181818, 0.1814159, 1.0000000, 0.1198582, 0, 0.6, 0.750),
  c(1.0000000, 0.1814159, 0.6666667, 0.1198582, 0, 0.6, 0.375),
  c(1.0000000, 0.1814159, 0.8333333, 0.1198582, 0, 0.6, 0.125),
  c(0.8181818, 0.0000000, 1.0000000, 0.3482143, 0, 0.6, 0.375),
  c(0.1818182, 0.1814159, 1.0000000, 0.1198582, 0, 0.2, 0.375),
  c(0.1818182, 0.1814159, 0.5000000, 0.1198582, 0, 0.2, 0.125),
```

```
  c(0.0000000, 1.0000000, 0.0000000, 0.5741667, 1, 1.0, 1.000),
  c(0.3636364, 0.7787611, 0.6666667, 1.0000000, 1, 0.0, 0.500),
  c(0.4545455, 0.1814159, 0.9166667, 0.1198582, 0, 0.4, 0.000),
  c(0.1818182, 0.6283186, 0.5833333, 0.0000000, 0, 0.4, 0.125)
)
rownames(M) <- alternatives
colnames(M) <- criteria
dominated <- mcda_get_dominated(M)
dominated
```

---

mcda_rescale_pm   *mcda_rescale_pm : Rescale performance matrix*

Description

Rescales performance matrix resulting in matrix wich all criteria in same scale. Only works with numeric values.

Usage

```
mcda_rescale_pm(M)
```

Arguments

M	Performance matrix - criteria in columns and alternatives in rows. Expected, that all criteria are to be maximized and are expressed as numbers.

Value

Returns scaled performance matrix.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

De Brouwer, Philippe J. S.: "The The Big R-Book: From Data Science to Learning Machines and Big Data ", Wiley, 2020, 928 p., ISBN 978-1119632726.

Examples

```
alternatives <- c('BLR', 'BOM', 'DEL', 'MNL', 'HYD', 'GRU', 'DUB', 'KRK', 'MAA', 'EZE')
criteria <- c('tlnt', 'stab', 'cost', 'infl', 'tm-zn', 'infr', "life")
M <- rbind(
  c(0.8181818, 0.1814159, 1.0000000, 0.1198582, 0, 0.6, 0.750),
  c(1.0000000, 0.1814159, 0.6666667, 0.1198582, 0, 0.6, 0.375),
  c(1.0000000, 0.1814159, 0.8333333, 0.1198582, 0, 0.6, 0.125),
  c(0.8181818, 0.0000000, 1.0000000, 0.3482143, 0, 0.6, 0.375),
  c(0.1818182, 0.1814159, 1.0000000, 0.1198582, 0, 0.2, 0.375),
  c(0.1818182, 0.1814159, 0.5000000, 0.1198582, 0, 0.2, 0.125),
  c(0.0000000, 1.0000000, 0.0000000, 0.5741667, 1, 1.0, 1.000),
  c(0.3636364, 0.7787611, 0.6666667, 1.0000000, 1, 0.0, 0.500),
  c(0.4545455, 0.1814159, 0.9166667, 0.1198582, 0, 0.4, 0.000),
```

```
  c(0.1818182, 0.6283186, 0.5833333, 0.0000000, 0, 0.4, 0.125)
)
rownames(M) <- alternatives
colnames(M) <- criteria
rescaled <- mcda_rescale_pm(M)
```

---

mcda_wsm *mcda_wsm : applies weighted sum method on performace matrix*

Description

Weighted sum method (WSM) is one of simpliest and at the same time one of most used methods for evaluation of alternatives using different criteria. For the method to work it is neccessary to prepare performance matrix with alternatives in the rows and criteria in the columns.

Values in the matrix must be notmalized. Current version of the implementation also presumes, that all criteria in performance matrix are to be maximized.

The methods takes performance matrix, applies weights for the criteria on it and sums the results accros the criteria to get score for the alternatives. Such score is usable to rank the alternatives. Score is provided in both raw (just sum of values) and percentage forms.

Usage

```
mcda_wsm(M, w, minmaxcriteria = 'max')
```

Arguments

M	Performance matrix - criteria in columns and alternatives in rows. Expected, that all values in matrix are normalized and that the criteria are to be maximized. Only numeric values in the matrix are expected.
w	weights for the critera in the performance matrix.
minmaxcriteria	criteriaMinMax Vector containing the preference direction on each of the criteria. "min" (resp."max") indicates that the criterion has to be minimized (maximized).

Value

Returns list of results in following structure:

performanceMatrix	A matrix containing the performance table. Each row corresponds to an alternative, and each column to a criterion. Output provides rescaled version of matrix.
weightedPM	Weighted performance matrix
weightedSum	vector containing weighted sum for alternatives accross the criteria
weightedSumPrc	weingtedSum expresed as percentage - max of weingtedSum value = 100 %
scoreM	visualized contibution of criteria to overal score of the variant

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Šenovský, P: "Modeling of Decision Processes (in czech)", 4th edition, VŠB - Technical University of Ostrava, 2012, 113 p.

Examples

```
alternatives <- c('BLR', 'BOM', 'DEL', 'MNL', 'HYD', 'GRU', 'DUB', 'KRK', 'MAA', 'EZE')
criteria <- c('tlnt', 'stab', 'cost', 'infl', 'tm-zn', 'infr', "life")
M <- rbind(
  c(0.8181818, 0.1814159, 1.0000000, 0.1198582, 0, 0.6, 0.750),
  c(1.0000000, 0.1814159, 0.6666667, 0.1198582, 0, 0.6, 0.375),
  c(1.0000000, 0.1814159, 0.8333333, 0.1198582, 0, 0.6, 0.125),
  c(0.8181818, 0.0000000, 1.0000000, 0.3482143, 0, 0.6, 0.375),
  c(0.1818182, 0.1814159, 1.0000000, 0.1198582, 0, 0.2, 0.375),
  c(0.1818182, 0.1814159, 0.5000000, 0.1198582, 0, 0.2, 0.125),
  c(0.0000000, 1.0000000, 0.0000000, 0.5741667, 1, 1.0, 1.000),
  c(0.3636364, 0.7787611, 0.6666667, 1.0000000, 1, 0.0, 0.500),
  c(0.4545455, 0.1814159, 0.9166667, 0.1198582, 0, 0.4, 0.000),
  c(0.1818182, 0.6283186, 0.5833333, 0.0000000, 0, 0.4, 0.125)
)
rownames(M) <- alternatives
colnames(M) <- criteria
w = c(0.125, 0.2, 0.2, 0.2, 0.175, 0.05, 0.05)
wsm <- mcda_wsm(M, w)
wsm$weightedPM
wsm$weightedSum
wsm$weightedSumPrc
```

---

norm_LaiHwang *Normalize values using Lai and Hwang approach*

Description

Normalize values using approach by Lai and Hwang (1994), see equations bellow.

For benefit criteria

$$z = \frac{x}{max(x) - min(x)}$$

For cost criteria

$$z = \frac{x}{min(x) - max(x)}$$

Usage

```
norm_LaiHwang(tonorm, minmax)
```

Arguments

`tonorm`	vector of numeric values to be normalized
`minmax`	'min' or 'max' to specify cost or benefit criterion

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Lai, Young-hou & Ching-Lai Hwang (1994). Fuzzy Multiple Objective Decision Making Method and Applications. Im Lecture Notes in Economics and Mathematical Systems 404. Berlin, Heidelberg: Springer-Verlag. DOI: 10.007/978-3-642-57949-3.

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_LaiHwang(tonorm, 'max')
```

---

norm_linearagreg — *Linear normalization using aggregation of values*

Description

Normalized values are computed as follows.

For maximization transformation

$$z = \frac{x}{\sum_{i=1}^{m} x_i}$$

For minimization transformation

$$z = \frac{\frac{1}{x}}{\sum_{i=1}^{m} \frac{1}{x_i}}$$

where m is number of observation in vector to normalize

Usage

```
norm_linearagreg(tonorm, minmax = 'max')
```

Arguments

tonorm	vector of numeric values to be normalized
minmax	min or max, provided to support situation when we want to revert direction of used scale. If value not provided, max presumed.

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_linearagreg(tonorm)
```

---

norm_logarithm — *Normalize values by applying logarithmic transformation.*

Description

Normalized values are computed as follows.

For maximization transformation

$$z = \frac{ln(x)}{ln(\prod_{i=1}^{m} x_i)}$$

For minimization transformation

$$z = \frac{1 - \frac{ln(x)}{ln(\prod_{i=1}^{m} x_i)}}{m - 1}$$

where m is number of observation in vector to normalize

Usage

```
norm_logarithm(tonorm, minmax = 'max')
```

Arguments

`tonorm`	vector of numeric values to be normalized
`minmax`	min or max, provided to support situation when we want to revert direction of used scale. If value not provided, max presumed.

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_logarithm(tonorm)
```

`norm_markovic` *Normalize values Markovic approach*

Description

Normalize values using approach by Markovic (2010), see equation bellow. Main benefit is that the normalization works bor both benefit and cost criteria.

$$z = 1 - \frac{x - min(x)}{max(x)}$$

Usage

```
norm_markovic(tonorm)
```

Arguments

`tonorm`	vector of numeric values to be normalized

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

MARKOVIC, Z. Modification of TOPSIS Method For Solving Multicriteria Tasks. Yugoslav Journal of Operations Researc, vol. 20, no. 1, pp. 117-143, DOI: 10.2298/YJOR1001117M

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_markovic(tonorm)
```

`norm_minmax` *Min-max method for normalization of the values*

Description

Min-max normalization is one of most used normalization methods. It takes values in any ascale an normalizes them into 0-1 scale.

Normalized values z are computed:

$$z = \frac{x - min(x)}{max(x) - min(x)}$$

Usage

```
norm_minmax(tonorm, minmax = 'max')
```

Arguments

`tonorm`	vector of numeric values to be normalized
`minmax`	min or max, provided to support situation when we want to revert direction of used scale. If value not provided, max presumed.

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Normalization. Code academy, available from https://www.codeacademy.com/articles/normalization [cit. 2021-09-28]

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_minmax(tonorm)
```

`norm_nonlinear` *Normalize values using nonlinear normalization*

Description

Normalize values using approach by Oeldschus et al (1983), see equations bellow.

For benefit criteria

$$z = (\frac{x}{max(x)})^2$$

For cost criteria

$$z = (\frac{max(x)}{x})^2$$

Usage

```
norm_nonlinear(tonorm, minmax)
```

Arguments

`tonorm`	vector of numeric values to be normalized
`minmax`	'min' or 'max' to specify cost or benefit criterion

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Peldschus, F., Vaigauskas, E. and Zavadskas, E.K. (1983) Technologische Entscheidungen bei der Berücksichtigung mehrerer Ziele. Bauplanung - Bautechnik, vol. 37, no. 4, pp. 173-175.

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_nonlinear(tonorm, 'max')
```

---

norm_toaverage — *Normalize values by comparing them to average.*

Description

Normalized values are computed by comparing the values to average.

$$z = \frac{x}{\mu} \cdot 100$$

Usage

```
norm_toaverage(tonorm)
```

Arguments

`tonorm`	vector of numeric values to be normalized

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_toaverage(tonorm)
```

`norm_tobest` *Normalize values by comparing them to maximum in the vector.*

Description

Normalized values are computed by comparing the values to maximum. Results show how close the values are to this maximum (in percents).

$$z = \frac{x}{max(x)} \cdot 100$$

Usage

```
norm_tobest(tonorm)
```

Arguments

`tonorm`	vector of numeric values to be normalized

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_tobest(tonorm)
```

`norm_TzengHuang` *Normalize values using Tzeng and Huang approach*

Description

Normalize values using approach by Tzeng and Huang (2011), see equation bellow. Main benefit is that the normalization works bor both benefit and cost criteria.

$$z = \frac{max(x)}{x}$$

Usage

```
norm_TzengHuang(tonorm)
```

Arguments

`tonorm`	vector of numeric values to be normalized

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Tzeng, G. and Huang, J. (2011) Multiple Attribute Decision Making Methods and Applications. CRC Press, Taylor and Francis Group, A Chapman & Hall Book, Boca Raton. 350 p., ISBN 978-1-4398-6157-8

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_TzengHuang(tonorm)
```

`norm_vector` *Performs so called vector normalization*

Description

Normalized values are computed as follows.

For maximization transformation

$$z = \frac{x}{\sqrt{\sum_{i=1}^{m} x_i^2}}$$

For minimization transformation

$$z = \frac{\frac{1}{x}}{\sqrt{\sum_{i=1}^{m} \frac{1}{x_i^2}}}$$

where m is number of observation in vector to normalize

Usage

```
norm_vector(tonorm, minmax = 'max')
```

Arguments

`tonorm`	vector of numeric values to be normalized
`minmax`	min or max, provided to support situation when we want to revert direction of used scale. If value not provided, max presumed.

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_vector(tonorm)
```

---

norm_ZavadskasTurskis *Normalize values using Zavadskas and Turskis approach*

Description

Normalize values using approach by Zavadskas and Turskis (2008), see equations bellow.

For benefit criteria

$$z = 1 - |\frac{max(x) - x}{max(x)}|$$

For cost criteria

$$z = 1 - |\frac{min(x) - x}{min(x)}|$$

Usage

```
norm_ZavadskasTurskis(tonorm, minmax)
```

Arguments

`tonorm`	vector of numeric values to be normalized
`minmax`	'min' or 'max' to specify cost or benefit criterion

Value

returns normalized vector of values.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Zavadskas, E.K., Turskis, Z. (2008). A New Logarithmic Normalization Method in Games Theory. Informatica, vol. 19, no. 2, pp. 303-314

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_ZavadskasTurskis(tonorm, 'max')
```

`norm_zscore` *Z-Score - standard score*

Description

Standard score is the number of standard deviations by which the value of raw score is above or below mean value. Values > mean will have positive scores, while values under mean will be negative. Normalized values z are computed:

$$z = \frac{x - \mu}{\sigma}$$

Usage

```
norm_zscore(tonorm)
```

Arguments

`tonorm`	vector of numeric values to be normalized

Value

returns standard score for presented vector.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Normalization. Code academy, available from https://www.codeacademy.com/articles/normalization [cit. 2021-09-28]

Examples

```
tonorm <- c(1,2,3)
normalized <- norm_zscore(tonorm)
```

`plot.prefM` *plot.prefM : Function to plot preference direction using plotmat function*

Description

Takes preference matrix and uses it to plot the relations between the alternatives (what altrenative is prefered in comparison to which alternative).

Usage

```
plot(x, ...)
```

Arguments

x	preference matrix - values in matrix used to descibe strength of the relation.
...	allows user to specify other parameters of plotmat function from diagram package.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

De Brouwer, Philippe J. S.: "The The Big R-Book: From Data Science to Learning Machines and Big Data ", Wiley, 2020, 928 p., ISBN 978-1119632726.

Examples

```
alternatives <- c('BLR', 'BOM', 'DEL', 'MNL', 'HYD', 'GRU', 'DUB', 'KRK', 'MAA', 'EZE')
M <- rbind(
  c(0, 0, 0, 1, 1, 0, 0, 0, 1, 0),
  c(0, 0, 1, 1, 1, 1, 0, 0, 1, 0),
  c(0, 0, 0, 1, 1, 0, 0, 0, 1, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 1, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 1, 1, 0, 0, 0, 0)
)
rownames(M) <- alternatives
colnames(M) <- alternatives
class(M)    <- 'prefM'
plot(M)
```

---

plot.scoreM — *plot.scoreM : Function to plot visualise contribution of the criteria to overal performance of the alternatives.*

Description

Takes weighted preference matrix and uses it to plot bar chart describing how the criteria contribute to the overal score of the alternatives. Visualized in form of the ordered bar plot.

Usage

```
plot.scoreM(x, ...)
```

Arguments

x	weighted preference matrix - values in matrix used to descibe strength of the relation.
...	allows user to specify other parameters of plotmat function from diagram package.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

De Brouwer, Philippe J. S.: "The The Big R-Book: From Data Science to Learning Machines and Big Data ", Wiley, 2020, 928 p., ISBN 978-1119632726.

Examples

```
  KoboGlo <- c(80, 97, 50, 100, 100, 94)
SonyPRST3 <- c(80, 88, 100, 20, 70, 70)
KindlePaperwhite2 <- c(100, 85, 50, 100, 70, 100)
PBTouchLux <- c(80, 90, 100, 100, 70, 94)
BookeenCybookOdyssey <- c(80, 100, 100, 100, 85, 50)
criteria <- c("Display", "váha", "HWTlačítka", "FrontLight", "baterie", "cena")
PM <- as.data.frame(
  rbind(KoboGlo,
        SonyPRST3,
        KindlePaperwhite2,
        PBTouchLux,
        BookeenCybookOdyssey))
names(PM) <- criteria
w <- c(5, 3, 4, 5, 2, 1)
preferences <- mcda_wsm(PM, w, 'max')
plot.scoreM(preferences$weightedPM)
```

---

pre_order_matrix — *pre_order_matrix - function to create preorder matrix*

Description

Function takes outputs of descending and ascending distilation pre-order and created pre-order matrix describing outranking relation between the alternatives.

The relation can be P+ (a outranks b), P- (b outranks a), I (indifference) and R (incomparable). Information from preorder matrix can be utilized to create adjancancy matrix and construct final ranking.

Usage

```
pre_order_matrix(rank_D, rank_A, alt)
```

Arguments

`rank_D`	descending distilation ranking in the form of ordered dataframe
`rank_A`	ascending distilation ranking in the form of ordered dataframe
`alt`	vector of alternatives names

Value

returns pre-order matrix.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

Prombo, M. Package OutrankingTools, CRAN: 2015, available from: https://cran.r-project.org/web/packages/OutrankingTools/

PROMETHEE — *PROMETHEE : general method for computations of preference flows in PROMETHEE methods*

Description

PROMETHEE stands for Preference ranking organization method for enrichment evaluation. Is family of populat alternatives outranking computation methods.It constructs its recomendations based on positive and negative preference flows between the alternatives.

This function provides general function implementing computation of these flows to be used by PROMETHEE I, II and III methods.

Usage

```
PROMETHEE(PM, preferenceFunction, w, indifferenceTreshold = NULL,
prefferenceThreshold = NULL, intermediateThreshold = NULL)
```

Arguments

`PM` Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.

`preferenceFunction` vector, specifies type of function used to compute preferences. Need to be set for each criterion. Possible values are: 'default', 'U-shape', 'V-shape', 'level', 'linear', 'Gaussian'. Choice of function type will decide on what type of threshold (if any) is required for computation. Each criterion can use different preference function.

`w` vector containing the weights of the criteria. Values need to 0 <= wi <= 1, sum(wi) = 1

`indifferenceTreshold` vector containing indifference threshods for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'U-shape', 'level', 'linear' functions need this threshold.

`prefferenceThreshold`
: vector containing prefference threshods for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'V-shape', 'level', 'linear' functions need this threshold.

`intermediateThreshold`
: vector containing intermetiate thresholds for criteria. only Gaussian type performance functions rewuire this type of threshold. If prefference and indifference thresholds are present, the PROMETHEE function will try to 'gues' intermediate threshold as value right in the middle between these thresholds.

Value

The function returns a list structured as follows:

`positiveFlowCriteria`
: matrix of size no. alternatives x no. of criteria representing how the alternative is preffered in criterium compared to other alternatives

`negativeFlowCriteria`
: matrix of size no. alternatives x no. of criteria representing how the alternative is outranked in criterium compared to other alternatives

`netFlowCriteria`
: matrix of size no. alternatives x no. of criteria representing overal evaluation of the flows

`weightedPositiveFlowCriteria`
: matrix of size no. alternatives x no. of criteria representing how the alternative is preffered in criterium compared to other alternatives with weights applied to them

`weightedNegativeFlowCriteria`
: matrix of size no. alternatives x no. of criteria representing how the alternative is outranked in criterium compared to other alternatives with weights applied to them

`weightedNetFlowCriteria`
: matrix of size no. alternatives x no. of criteria representing overal evaluation of the flows (with weights applied to them)

`positiveFlow` vector representing how the alternative is preffered to other alternatives

`negativeFlow` vector representing how altenative is outranked by other alternatives

`netFlow` vector representing diferences between positive and negative flows for the alternative across criteria

`preferenceDegreeUnw`
: list of matrixes with unweighted preferences, separate matrix for each criterion

`pairweiseComparison`
: list of matrixes measuring nominal differences in alternatives performance in criterions. Separate matrixes are constructed for each criterion.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

Examples

```
#Example from Fuzzy TOPSIS book (see references)
#ammended error in tab. 4.9, the computation presumes maximization of all criteria
PM <- cbind(
  c(80, 65, 83, 40, 52, 94),
  c(90, 58, 60, 80, 72, 96),
  c(600, 200, 400, 1000, 600, 700),
  c(54, 97, 72, 75, 20, 36),
  c(8, 1, 4, 7, 3, 5),
  c(5, 1, 7, 10, 8, 6)
)
colnames(PM) <- c('C1', 'C2', 'C3', 'C4', 'C5', 'C6')
rownames(PM) <- c('A1', 'A2', 'A3', 'A4', 'A5', 'A6')
minmax <- 'max'
shape <- c('U-shape', 'V-shape', 'linear', 'level', 'default', 'Gaussian')
q <- c(10, 0, 450, 50, 0, 0) #indifference threshold
p <- c(0, 30, 50, 10, 0, 0) #prefference threshold
s <- c(0,0,0,0,0,5) #intermediate threshold
w <- c(0.1667, 0.1667, 0.1667, 0.1667, 0.1667, 0.1665)
result <- PROMETHEE(PM, shape, w, q, p, s)
```

PROMETHEE_I — *PROMETHEE I: method for computations of partial preference using PROMETHEE I method*

Description

PROMETHEE stands for Preference ranking organization method for enrichment evaluation. Promethee I methot is intended for establishment of partial ranking of the alternatives, by evaluation of positive and negative prefference flows in pairweise comparisons of the alternatives.

Method uses general PROMETHEE function and computes comparison states on top of it

Usage

```
PROMETHEE_I(PM, preferenceFunction, w, minmax, indifferenceTreshold = NULL,
prefferenceThreshold = NULL, intermediateThreshold = NULL)
```

Arguments

`PM` Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.

`preferenceFunction` vector, specifies type of function used to compute preferences. Need to be set for each criterion. Possible values are: 'default', 'U-shape', 'V-shape', 'level', 'linear', 'Gaussian'. Choice of function type will decide on what type of threshold (if any) is required for computation. Each criterion can use different preference function.

`w` vector containing the weights of the criteria. Values need to 0 <= wi <= 1, sum(wi) = 1 `minmax` can be set to either value or vector. Value (min or max) is usable in situation when all criteria are either benefit or cost (are not mixed). If Criteria orientation is mixed, vector is required to set criterion orientation right.

`indifferenceTreshold`
vector containing indifference threshods for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'U-shape', 'level', 'linear' functions need this threshold.

`prefferenceThreshold`
vector containing prefference threshods for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'V-shape', 'level', 'linear' functions need this threshold.

`intermediateThreshold`
vector containing intermetiate thresholds for criteria. only Gaussian type performance functions rewuire this type of threshold. If prefference and indifference thresholds are present, the PROMETHEE function will try to 'gues' intermediate threshold as value right in the middle between these thresholds.

Value

The function returns a list structured as follows:

`positiveFlow` vector representing how the alternative is preffered to other alternatives

`negativeFlow` vector representing how altenative is outranked by other alternatives

`preferenceDegree`
matrix representing aggregated weighted preferences of the alternatives across the criteria

`preferenceDegreeUnw`
list of matrixes with unweighted preferences, separate matrix for each criterion

`pairweiseComparison`
list of matrixes measuring nominal differences in alternatives performance in criterions. Separate matrixes are constructed for each criterion.

`preferenceMatrix`
preference matrix with specified P (prefered), I (indifferent) and R (incomparable) for every pair of alternatives

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

Examples

```
#Example from Fuzzy TOPSIS book (see references)
#ammended error in tab. 4.9, the computation presumes maximization of all criteria
PM <- cbind(
```

```
  c(80, 65, 83, 40, 52, 94),
  c(90, 58, 60, 80, 72, 96),
  c(600, 200, 400, 1000, 600, 700),
  c(54, 97, 72, 75, 20, 36),
  c(8, 1, 4, 7, 3, 5),
  c(5, 1, 7, 10, 8, 6)
)
colnames(PM) <- c('C1', 'C2', 'C3', 'C4', 'C5', 'C6')
rownames(PM) <- c('A1', 'A2', 'A3', 'A4', 'A5', 'A6')
minmax <- 'max'
shape <- c('U-shape', 'V-shape', 'linear', 'level', 'default', 'Gaussian')
q <- c(10, 0, 450, 50, 0, 0) #indifference threshold
p <- c(0, 30, 50, 10, 0, 0) #prefference threshold
s <- c(0,0,0,0,0,5) #intermediate threshold
w <- c(0.1667, 0.1667, 0.1667, 0.1667, 0.1667, 0.1665)
result <- PROMETHEE_I(PM, shape, w, minmax, q, p, s)
```

PROMETHEE_II *PROMETHEE II: method for computations of partial preference using PROMETHEE II method*

Description

PROMETHEE stands for Preference ranking organization method for enrichment evaluation. Promethee II methotdis intended for establishment of full ranking of the alternatives, by evaluation of positive and negative prefference flows in pairweise comparisons of the alternatives.

Difference between positive and negative flows forms net outranking flow, which is indicator usable directly to order the alternatives.

Method uses general PROMETHEE function and computes net outranking flow on top of it and ranks.

Usage

```
PROMETHEE_II(PM, preferenceFunction, w, minmax = 'max', indifferenceTreshold = NULL,
prefferenceThreshold = NULL, intermediateThreshold = NULL)
```

Arguments

`PM` Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.

`preferenceFunction` vector, specifies type of function used to compute preferences. Need to be set for each criterion. Possible values are: 'default', 'U-shape', 'V-shape', 'level', 'linear', 'Gaussian'. Choice of function type will decide on what type of threshold (if any) is required for computation. Each criterion can use different preference function.

`w` vector containing the weights of the criteria. Values need to 0 <= wi <= 1, sum(wi) = 1

`minmax` can be set to either value or vector. Value (min or max) is usable in situation when all criteria are either benefit or cost (are not mixed). If Criteria orientation is mixed, vector is required to set criterion orientation right.

`indifferenceTreshold`
: vector containing indifference threshods for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'U-shape', 'level', 'linear' functions need this threshold.

`prefferenceThreshold`
: vector containing prefference threshods for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'V-shape', 'level', 'linear' functions need this threshold.

`intermediateThreshold`
: vector containing intermetiate thresholds for criteria. only Gaussian type performance functions rewuire this type of threshold. If prefference and indifference thresholds are present, the PROMETHEE function will try to 'gues' intermediate threshold as value right in the middle between these thresholds.

Value

The function returns a list structured as follows:

`positiveFlow` vector representing how the alternative is preffered to other alternatives

`negativeFlow` vector representing how altenative is outranked by other alternatives

`netFlow` positive - negative flow, forms an indicator PROMETHEE II uses to directly rank the alternatives

`preferenceDegree`
: matrix representing aggregated weighted preferences of the alternatives across the criteria

`preferenceDegreeUnw`
: list of matrixes with unweighted preferences, separate matrix for each criterion

`pairweiseComparison`
: list of matrixes measuring nominal differences in alternatives performance in criterions. Separate matrixes are constructed for each criterion.

`rankedList` ranked list of alternatives

`rankedListNOF` sorted list of alternatives with assigned net outranking flow indicator, which can be used to sort the list.

`preferenceMatrix`
: preference matrix describing ourranking relation between the alternatives

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

Examples

```
#Example from Fuzzy TOPSIS book (see references)
#ammended error in tab. 4.9, the computation presumes maximization of all criteria
PM <- cbind(
  c(80, 65, 83, 40, 52, 94),
  c(90, 58, 60, 80, 72, 96),
  c(600, 200, 400, 1000, 600, 700),
  c(54, 97, 72, 75, 20, 36),
  c(8, 1, 4, 7, 3, 5),
  c(5, 1, 7, 10, 8, 6)
)
colnames(PM) <- c('C1', 'C2', 'C3', 'C4', 'C5', 'C6')
rownames(PM) <- c('A1', 'A2', 'A3', 'A4', 'A5', 'A6')
minmax <- 'max'
shape <- c('U-shape', 'V-shape', 'linear', 'level', 'default', 'Gaussian')
q <- c(10, 0, 450, 50, 0, 0) #indifference threshold
p <- c(0, 30, 50, 10, 0, 0) #prefference threshold
s <- c(0,0,0,0,0,5) #intermediate threshold
w <- c(0.1667, 0.1667, 0.1667, 0.1667, 0.1667, 0.1665)
result <- PROMETHEE_II(PM, shape, w, minmax, q, p, s)
```

PROMETHEE_III *PROMETHEE III: method for computations of partial preference using PROMETHEE III method*

Description

PROMETHEE stands for Preference ranking organization method for enrichment evaluation. Promethee II methotdis intended for establishment of full ranking of the alternatives, by evaluation of positive and negative prefference flows in pairweise comparisons of the alternatives.

Difference between positive and negative flows forms net outranking flow. PROMETHEE III function is usable for partial preorder, similarly to PROMETHEE I function, but uses intervals for establishing preorder.

Method uses general PROMETHEE function and computes net outranking flow on top of it. Standard error of the net flow will then be used to compute the intervals for the alternatives. These intevals are then used to establish preference system.

In a way we can think about these interval as an alternative implementation of indifference thresholds.

Note: the implementation of function is partially inspired by implementation of PROMETHEE III portion of promethee123 package, thou this implementation provides different output and utilizes general PROMETHEE developed for PROMETHEE I and II functions.

Usage

```
PROMETHEE_III(PM, preferenceFunction, w, minmax = 'max', indifferenceTreshold = NULL,
prefferenceThreshold = NULL, intermediateThreshold = NULL)
```

Arguments

`PM` Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.

`preferenceFunction` vector, specifies type of function used to compute preferences. Need to be set for each criterion. Possible values are: 'default', 'U-shape', 'V-shape', 'level', 'linear', 'Gaussian'. Choice of function type will decide on what type of threshold (if any) is required for computation. Each criterion can use different preference function.

`w` vector containing the weights of the criteria. Values need to 0 <= wi <= 1, sum(wi) = 1

`minmax` can be set to either value or vector. Value (min or max) is usable in situation when all criteria are either benefit or cost (are not mixed). If Criteria orientation is mixed, vector is required to set criterion orientation right.

`indifferenceTreshold` vector containing indifference thresholds for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'U-shape', 'level', 'linear' functions need this threshold.

`prefferenceThreshold` vector containing prefference threshods for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'V-shape', 'level', 'linear' functions need this threshold.

`intermediateThreshold` vector containing intermetiate thresholds for criteria. only Gaussian type performance functions rewuire this type of threshold. If prefference and indifference thresholds are present, the PROMETHEE function will try to 'gues' intermediate threshold as value right in the middle between these thresholds.

Value

The function returns a list structured as follows:

`positiveFlow` vector representing how the alternative is preffered to other alternatives

`negativeFlow` vector representing how altenative is outranked by other alternatives

`netFlow` positive - negative flow, forms an indicator PROMETHEE II uses to directly rank the alternatives

`preferenceDegree` matrix representing aggregated weighted preferences of the alternatives across the criteria

`preferenceDegreeUnw` list of matrixes with unweighted preferences, separate matrix for each criterion

`pairweiseComparison` list of matrixes measuring nominal differences in alternatives performance in criterions. Separate matrixes are constructed for each criterion.

`preferenceMatrix` preference matrix describing ourranking relation between the alternatives

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

BRANS, Jean-Pierre; DE SMET, Yves. PROMETHEE methods. In: Multiple criteria decision analysis. Springer, New York, NY, 2016. p. 187-219. DOI: 10.1007/978-1-4939-3094-4_6.

Moreira, M.A.L., dos Santos, M., Gomes, C.F.S. promethee123 package. <https://cran.r-project.org/web/packages/prome

Examples

```
#Example from Fuzzy TOPSIS book (see references)
#ammended error in tab. 4.9, the computation presumes maximization of all criteria
PM <- cbind(
  c(80, 65, 83, 40, 52, 94),
  c(90, 58, 60, 80, 72, 96),
  c(600, 200, 400, 1000, 600, 700),
  c(54, 97, 72, 75, 20, 36),
  c(8, 1, 4, 7, 3, 5),
  c(5, 1, 7, 10, 8, 6)
)
colnames(PM) <- c('C1', 'C2', 'C3', 'C4', 'C5', 'C6')
rownames(PM) <- c('A1', 'A2', 'A3', 'A4', 'A5', 'A6')
minmax <- 'max'
shape <- c('U-shape', 'V-shape', 'linear', 'level', 'default', 'Gaussian')
q <- c(10, 0, 450, 50, 0, 0) #indifference threshold
p <- c(0, 30, 50, 10, 0, 0) #prefference threshold
s <- c(0,0,0,0,0,5) #intermediate threshold
w <- c(0.1667, 0.1667, 0.1667, 0.1667, 0.1667, 0.1665)
result <- PROMETHEE_III(PM, shape, w, minmax, q, p, s)
```

---

rankDF *rankDF - function to unpack list of ranks into ordered dataframe of alternatives in ranks*

Description

Takes list of ranks (ie. [1] "A1" [2] "A4" "A5" [3] "A2" [4] "A3") and unpacks them into dataframe with columns action and rank.

Example leads to dataframe: (action, rak) (A1, 1), (A4, 2), (A5, 2), (A2, 3), (A3, 4).

Usage

```
rankDF(ranklist)
```

Arguments

`ranklist`	ordered list of ranked alternatives.

Value

returns dataframe wtih ranked alternatives from worst-to best.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Meyer, P. at al. MCDA package. GitHub: 2021, available from: https://github.com/paterijk/MCDA/blob/master/

`SIR` *SIR : Superiority and inferiority ranking (SIR) method*

Description

SIR stands for Superiority and inferiority ranking (SIR) method.The method is interesting as it is computationwise closely related to PROMETHEE and TOPSIS methods. Similarly to them it computes preference based on flows.

The flows are computed based on S and I (superiority and inferiority) matrixes. Based on transformation of preference P(A, A2) from natural units to 0-1 according to 6 function (in same way as PROMETHEE computes them). S and I matrixes are dexined:

$$S_j(A_i) = \sum_{k=1}^{m} P_j(A_i, A_k) = \sum_{k=1}^{m} f_j(g_j(A_i) - g_j(A_k))$$

and

$$I_j(A_i) = \sum_{k=1}^{m} P_j(A_k, A_i) = \sum_{k=1}^{m} f_j(g_j(A_k) - g_j(A_i))$$

Where m is the number of alternatives, fj is transformed value using function d.

Results in S and I matrixes need to be agregated.The method supports two agregation methods SIR-SAW or SIR-TOPSIS.

For SIR-SAW:

$$SFlow(A_i) = \sum_{j=1}^{n} w_j S_j(A_i)$$

$$IFlow(A_i) = \sum_{j=1}^{n} w_j I_j(A_i)$$

SIR-TOPSIS on the other hand uses computation of ideal and antiideal solution and computes S- and I- flows from it. See TOPSIS documentation for details.

This approach is used separately for both S and I matrixes. For computation of ideal and anti-ideal solution min and max functions are switched.

Directly from these flow complete rankings can be derived, or simple aggregation of them as net flow (diference between S and I flows) and relative flow as a closeness to addition of the S and I flows.

Usage

```
SIR(PM, d, w, indifferenceTreshold = NULL,
prefferenceThreshold = NULL, intermediateThreshold = NULL, SAW = T)
```

Arguments

`PM`	Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. only numeric values expercted. Rows and columns are expected to be named.
`d`	vector, specifies type of function used to compute preferences. Need to be set for each criterion. Possible values are: 'default', 'U-shape', 'V-shape', 'level', 'linear', 'Gaussian'. Choice of function type will decide on what type of threshold (if any) is required for computation. Each criterion can use different preference function.
`w`	vector containing the weights of the criteria. Values need to 0 <= wi <= 1, sum(wi) = 1
`indifferenceTreshold`	vector containing indifference thresholds for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'U-shape', 'level', 'linear' functions need this threshold.
`prefferenceThreshold`	vector containing prefference thresholds for criteria. Not all types of performance functions require it. The parameter must be used if there is at least one criterion, for which it is required. Values for all other criteria should be 0 (and will not be used during computations). Only 'V-shape', 'level', 'linear' functions need this threshold.
`intermediateThreshold`	vector containing intermetiate thresholds for criteria. only Gaussian type performance functions rewuire this type of threshold. If prefference and indifference thresholds are present, the PROMETHEE function will try to 'gues' intermediate threshold as value right in the middle between these thresholds.
`SAW`	implicit TRUE, if set TRUE will aggregate S and I matrixes using SIR-SAW method, otherwise it will use SIR-TOPSIS.

Value

The function returns a list structured as follows:

`superiorityFlow`	vector of the alternatives derived from S matrix
`inferityFlow`	vector of the alternatives derived from I matrix
`rankSFlow`	vector of alternatives with assigned ranks (is possible to directly derive from superiorityFlow)
`rankIFlow`	vector of alternatives with assigned ranks (is possible to directly derive from inferiorityFlow)
`partialRanking`	matrix providing partial order of alternatives with P-prefered, I-indifferent and R-incomparable values
`netFlow`	differencce between superiority and inferiority flows
`relativeFlow`	expresses closeness to addition of the S and I flows
`flows`	matrix with all types of flows in one place

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

Papathanasiou, Jason, Ploskas, Nikolaos. Multiple Criteria Decision Aid Methods, Examples and Python Implementations. Springer, 173 p., ISBN 978-3-319-91648-4.

Examples

```
  #Example from the book (see references)
  PM <- rbind(
  c(8,7,2,1),
  c(5,3,7,5),
  c(7,5,6,4),
  c(9,9,7,3),
  c(11,10,3,7),
  c(6,9,5,4)
)
rownames(PM) <- c('Site 1', 'Site 2', 'Site 3', 'Site 4', 'Site 5', 'Site 6')
colnames(PM) <- c('Investment costs (million EUR)', 'Employment needs (hundred employees)', 'Social impact (1-7
                  'Environmental impact (1-7)')
minmax <- 'max'
w <- c(0.4, 0.4, 0.1, 0.2)
shape <- c('linear', 'linear', 'linear', 'linear')
q <- c(1, 1, 1, 1)
p <- c(2,2,2,2)
s <- c(0,0,0,0)
result <- SIR(PM, w, shape, minmax, q, p, s, SAW=T)
```

`TOPSIS` *TOPSIS : method used to solve multiple criteria decision making*

Description

The acronym TOPSIS stands for: Technique of Order Preference Similarity to the Ideal Solution. As name suggests TOPSIS provides its guidance based on evaluation of the similarity to both ideal and anti-ideal variant of the solution.

Original method uses 5 steps for the procedure. In first step the procedure normalizes values in performance matrix nad applies weights to it in step 2. In step 3 ideal variant A^* and anti-ideal variant A^- is computed as maximums and minimums of the criteria in performance matrix.

In step 4 distance to ideal D^* and anti-ideal variant D^- is computed and in step 5 used to compute closenes criterium

$$C_i = \frac{D_i^-}{D_i^- + D_i^*}$$

Criterium C is then directly usable to rank alternatives. C is always in interval of 0-1, the closer the value is to 1, the closer it is to ideal variant.

Usage

```
TOPSIS(PM, w, minmax)
```

Arguments

PM	Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. Only numeric values expercted. Rows and columns are expected to be named.
w	vector containing the weights of the criteria.
minmax	criteria MinMax Vector containing the preference direction on each of the criteria. "min" (resp."max") indicates that the criterion has to be minimized (maximized).

Value

The function returns a list structured as follows :

C	ordered list of alternatives using C criterium
normPM	normalized performance matrix
weightPM	weighted normalized performance matrix
A_ideal	positive ideal solution
A_anti	anti ideal solution
D_ideal	alternative closeness to ideal variant
D_anti	alternative closeness to anti-ideal variant

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

Papathanasiou, Jason, Ploskas, Nikolaos. Multiple Criteria Decision Aid Methods, Examples and Python Implementations. Springer, 173 p., ISBN 978-3-319-91648-4.

Examples

```
PM <- cbind(
  c(8,7,2,1),
  c(5,3,7,5),
  c(7,5,6,4),
  c(9,9,7,3),
  c(11,10,3,7),
  c(6,9,5,4)
)
colnames(PM) <- c('Site 1', 'Site 2', 'Site 3', 'Site 4', 'Site 5', 'Site 6')
rownames(PM) <- c('Investment costs (million EUR)', 'Employment needs (hundred employees)',
                  'Social impact (1-7)', 'Environmental impact (1-7)')
PM <- t(PM)
minmax <- 'max'
w <- c(0.4, 0.4, 0.1, 0.2)
v <- 0.5
result <- TOPSIS(PM, w, minmax)
```

util_pm_minmax — *util_pm_minmax : Process performance metrix - reverts minimalized criteria in the matrix*

Description

Internal function the package uses to prepare performace matrix for computations requiring the criteria to use only maximized direction.

Usage

```
util_pm_minmax(PM, minmaxcriteria)
```

Arguments

PM	performance matrix - criteria are expected in the columns, alternatives in rows, only numeric values allowed.
minmaxcriteria	provides either 'min' or 'max' (default) values if all criteria should be minimalized or maximalized. If it is not case provide vector specifying either 'max' or 'min' for each criterion.

Value

Returns processed performance matrix with all criteria to be maximized

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

Examples

```
alternatives <- c('BLR', 'BOM', 'DEL', 'MNL', 'HYD', 'GRU', 'DUB', 'KRK', 'MAA', 'EZE')
criteria <- c('tlnt', 'stab', 'cost', 'infl', 'tm-zn', 'infr', "life")
M <- rbind(
  c(0.8181818, 0.1814159, 1.0000000, 0.1198582, 0, 0.6, 0.750),
  c(1.0000000, 0.1814159, 0.6666667, 0.1198582, 0, 0.6, 0.375),
  c(1.0000000, 0.1814159, 0.8333333, 0.1198582, 0, 0.6, 0.125),
  c(0.8181818, 0.0000000, 1.0000000, 0.3482143, 0, 0.6, 0.375),
  c(0.1818182, 0.1814159, 1.0000000, 0.1198582, 0, 0.2, 0.375),
  c(0.1818182, 0.1814159, 0.5000000, 0.1198582, 0, 0.2, 0.125),
  c(0.0000000, 1.0000000, 0.0000000, 0.5741667, 1, 1.0, 1.000),
  c(0.3636364, 0.7787611, 0.6666667, 1.0000000, 1, 0.0, 0.500),
  c(0.4545455, 0.1814159, 0.9166667, 0.1198582, 0, 0.4, 0.000),
  c(0.1818182, 0.6283186, 0.5833333, 0.0000000, 0, 0.4, 0.125)
)
rownames(M) <- alternatives
colnames(M) <- criteria
minmaxcriteria <- c('max', 'max', 'min', 'min', 'max', 'max', 'max')
maxed <- util_pm_minmax(M, minmaxcriteria)
```

`util_prepare_minmaxcriteria`

util_prepare_minmaxcriteria : validates and prepares minmax vector for processing

Description

Internal function package uses to validate vector containing information on criteria optimization direction (minimizaing or maximizing). Since this operation is required for almost all decision analysis functions, it has been refactored into separate function.

If provided with direction to min or max - it will create the minmaxcriteria vector minimizing or maximizing all the criteria.

Usage

```
util_prepare_minmaxcriteria(ncrit, minmaxcriteria)
```

Arguments

`ncrit`	number of criteria used in decision problem (must be > 2)
`minmaxcriteria`	provides either 'min' or 'max' (default) values if all criteria should be minimalized or maximalized. If it is not case provide vector specifying either 'max' or 'min' for each criterion.

Value

Returns validated and completed vector with information on minimizing or maximizing criteria.

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

Examples

```
#5 maximizzed criteria
minmax <- util_prepare_minmaxcriteria(5)
#5 minimized criteria
minmax <- util_prepare_minmaxcriteria(5, minmaxcriteria = 'min')
#mix
minmax <- util_prepare_minmaxcriteria(5, c('min', 'min', 'max', 'max', 'min'))
```

VIKOR *VIKOR : method used to solve multiple criteria decision making*

Description

The acronym VIKOR stands for: VlseKriterijumska Optimizacija I Kompromisno Resenje, in Serbian multicriteria optimization and compromise solution. The method has been especially designed to deal with problematic situations when the alternatives are characterized by non-commensurable and conflicting criteria, for which VIKOR provides compromise solution. Methodologically VIKOR is close to another method TOPSIS. Original VIKOR uses five steps to derive such compromise solution.

Step 1: determine best and worst values of all criteria by serchinch for min and max values in the performance matrix for all criteria.

$$f_j^* = max_i f_{ij}$$

$$f_j^- = min_i f_{ij}$$

Step 2: compute values of Si and Ri

$$S_i = \sum_{j=1}^{n} \frac{w_j(f_j^* - f_{ij})}{f_j^* - f_j^-}, i = 1,2,...,m, j = 1,2,...,n$$

$$R_i = max_i \frac{w_j(f_j^* - f_{ij})}{f_j^* - f_j^-}, i = 1,2,...,m, j = 1,2,...,n$$

where n is number of criteria, m number of alternatives and w are criteria weights.

Step 3: compute values of Qi

$$Q_i = v\frac{S_i - S^*}{S^- - S^*} + (1-v)\frac{R_i - R^*}{R^- - R^*}, i = 1,2,...,m$$

where $S^* = min_i S_i$, $S^- = max_i S_i$, $R^* = min_i R_i$, $R^- = max_i R_i$ and v is the weight for strategy of majority of the criteria.

Note that the v has in Qi connection to 1 - v (individual regret). By specifying various values of v one can influence impact left or right term of the equation has on overal result. $v \in (0,1)$, where v = 0.5 means balance between both terms. If function's parameter v is not set the procedure will approximate its value by:

$$v = \frac{n+1}{2n}$$

Step 4: order alternatives by S, R, Q

Step 5: propose compromise solution

The compromise solution is identified by order of Q, looking at the S or R depending on v.If the winner is the same, we have solution. Otherwise compromise solution needs to be made using 1/(m - 1) value as test criterion.Compromise is formed from all alternatives in Q where the difference between first and tested value is lover than this criterion.

Usage

```
VIKOR(PM, w, minmax, v)
```

Arguments

PM	Matrix or data frame containing the performance table. Each row corresponds to an alternative, and each column to a criterion. Only numeric values expercted. Rows and columns are expected to be named.
w	vector containing the weights of the criteria.
minmax	criteria MinMax Vector containing the preference direction on each of the criteria. "min" (resp."max") indicates that the criterion has to be minimized (maximized).
v	weight for strategy of majority of the criteria in interval (0-1). If no value provided procedure assigns value 0.5

Value

The function returns a list structured as follows :

S	ordered list of alternatives using S-metric
R	ordered list of alternatives using R-metric
Q	ordered list of alternatives using Q-metric
compromiseSolution	list of alternatives forming compromise solution (based on Q, S, R metrics)

Author(s)

Pavel Šenovský <pavel.senovsky@vsb.cz>

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

Papathanasiou, Jason, Ploskas, Nikolaos. Multiple Criteria Decision Aid Methods, Examples and Python Implementations. Springer, 173 p., ISBN 978-3-319-91648-4.

Examples

```
PM <- cbind(
  c(8,7,2,1),
  c(5,3,7,5),
  c(7,5,6,4),
  c(9,9,7,3),
  c(11,10,3,7),
  c(6,9,5,4)
)
colnames(PM) <- c('Site 1', 'Site 2', 'Site 3', 'Site 4', 'Site 5', 'Site 6')
rownames(PM) <- c('Investment costs (million EUR)', 'Employment needs (hundred employees)',
                  'Social impact (1-7)', 'Environmental impact (1-7)')
PM <- t(PM)
minmax <- 'max'
w <- c(0.4, 0.4, 0.1, 0.2)
v <- 0.5
result <- VIKOR(PM, w, minmax, v)
```

`VIKORIndexes` *VIKORIndexes : subroutine to compute S & R index in VIKOR and FuzzyVIKOR methods*

Description

The acronym VIKOR stands for: VlseKriterijumska Optimizacija I Kompromisno Resenje, in Serbian multicriteria optimization and compromise solution.

The method presents generalized solution for computation of S and R indexes using wights, performance matrix (crisp performance matrix if using fuzzy numbers) and information on best and worst values in criteria.

Usage

```
VIKORIndexes(car, bw_perf, cw)
```

Arguments

`car`	array with the performances (crisp alternative ratings in fuzzy variant). Alternatives are in rows, criteria in columns
`bw_perf`	matrix with the best, worst performances and differences between them. Has columns (best, worst, difference) and rows for criteria
`cw`	vector containing the weights of the criteria.

Value

The function returns a list structured as follows:

`S`	ordered list of alternatives using S-metric
`R`	ordered list of alternatives using R-metric
`Q`	ordered list of alternatives using Q-metric
`compromiseSolution`	list of alternatives forming compromise solution (based on Q, S, R metrics)

Author(s)

Pavel Šenovský `<pavel.senovsky@vsb.cz>`

References

ALAOUI, Mohamed El. Fuzzy TOPSIS: Logic, Approaches, and Case Studies. Boca Raton: CRC Press, 2021. 216 s. ISBN 978-0-367-76748-8.

Papathanasiou, Jason, Ploskas, Nikolaos. Multiple Criteria Decision Aid Methods, Examples and Python Implementations. Springer, 173 p., ISBN 978-3-319-91648-4.

Index