Package ‘MCDA’

April 5, 2021

Version 0.0.21
Date 2021-04-05
Title Support for the Multicriteria Decision Aiding Process
Author Patrick Meyer, Sébastien Bigaret, Richard Hodgett, Alexandru-Liviu Olteanu
Maintainer Patrick Meyer <patrick.meyer@imt-atlantique.fr>
Imports Rglpk, glpkAPI, methods, RColorBrewer, combinat
Suggests Rgraphviz, cplexAPI
License EUPL (== 1.1)
Encoding UTF-8
URL https://github.com/paterijk/MCDA
NeedsCompilation no
Repository CRAN
Date/Publication 2021-04-05 14:40:06 UTC

R topics documented:

additiveValueFunctionElicitation .................................................. 2
AHP .............................................................................................. 5
applyPiecewiseLinearValueFunctionsOnPerformanceTable .................. 6
assignAlternativesToCategoriesByThresholds .................................. 8
ELECTREIIIDistillation ................................................................. 10
LPDMRSort .................................................................................. 11
LPDMRSortIdentifyIncompatibleAssignments .................................. 14
additiveValueFunctionElicitation

Elicitation of a general additive value function.

Description

Elicits a general additive value function from a ranking of alternatives.
Usage

additiveValueFunctionElicitation(performanceTable,
    criteriaMinMax, epsilon,
    alternativesRanks = NULL,
    alternativesPreferences = NULL,
    alternativesIndifferences = NULL,
    alternativesIDs = NULL,
    criteriaIDs = NULL)

Arguments

- **performanceTable**
  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).

- **criteriaMinMax**
  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **epsilon**
  Numeric value containing the minimal difference in value between two consecutive alternatives in the final ranking.

- **alternativesRanks**
  Optional vector containing the ranks of the alternatives. The elements are named according to the IDs of the alternatives. If not present, then at least one of alternativesPreferences or alternativesIndifferences should be given.

- **alternativesPreferences**
  Optional matrix containing the preference constraints on the alternatives. Each line of the matrix corresponds to a constraint of the type alternative a is strictly preferred to alternative b. If not present, then either alternativesRanks or alternativesIndifferences should be given.

- **alternativesIndifferences**
  Optional matrix containing the indifference constraints on the alternatives. Each line of the matrix corresponds to a constraint of the type alternative a is indifferent to alternative b. If not present, then either alternativesRanks or alternativesPreferences should be given.

- **alternativesIDs**
  Vector containing IDs of alternatives, according to which the data should be filtered.

- **criteriaIDs**
  Vector containing IDs of criteria, according to which the data should be filtered.

Value

The function returns a list structured as follows:

- **optimum**
  The value of the objective function.

- **valueFunctions**
  A list containing the value functions which have been determined. Each value function is defined by a matrix of breakpoints, where the first row corresponds to the abscissa (row labelled "x") and where the second row corresponds to the ordinate (row labelled "y").
overallValues A vector containing the overall values of the input alternatives.
ranks A vector containing the ranks of the alternatives obtained via the elicited value functions. Ties method = "min".
Kendall Kendall’s tau between the input ranking and the one obtained via the elicited value functions.
errors The errors (sigma) which have to be added to the overall values of the alternatives in order to respect the input ranking.

References

Based on the UTA algorithm (E. Jacquet-Lagreze, J. Siskos, Assessing a set of additive utility functions for multicriteria decision-making, the UTA method, European Journal of Operational Research, Volume 10, Issue 2, 151–164, June 1982) except that the breakpoints of the value functions are the actual performances of the alternatives on the criteria.

Examples

# ----------------------------------------
# ranking some cars (from original article on UTA by Siskos and Lagreze, 1982)
# the separation threshold
epsilon <-0.01

# the performance table

performanceTable <- rbind(
  c(173, 11.4, 10.01, 10, 7.88, 49500),
  c(176, 12.3, 10.48, 11, 7.96, 46700),
  c(142, 8.2, 7.30, 5, 5.65, 32100),
  c(148, 10.5, 9.61, 7, 6.15, 39150),
  c(178, 14.5, 11.05, 13, 8.06, 64700),
  c(180, 13.6, 10.40, 13, 8.47, 75700),
  c(182, 12.7, 12.26, 11, 7.81, 68593),
  c(145, 14.3, 12.95, 11, 8.38, 55000),
  c(161, 8.6, 8.42, 7, 5.11, 35200),
  c(117, 7.2, 6.75, 3, 5.81, 24800)
)

colnames(performanceTable) <- c("MaximalSpeed",
                            "ConsumptionTown",
                            "Consumption120kmh",
                            "HP",
                            "Space",
                            "Price")

# ranks of the alternatives
alternativesRanks <- c(1,2,3,4,5,6,7,8,9,10)

names(alternativesRanks) <- row.names(performanceTable)

# criteria to minimize or maximize
criteriaMinMax <- c("max", "min", "min", "max", "max", "min")

names(criteriaMinMax) <- colnames(performanceTable)

x <- additiveValueFunctionElicitation(performanceTable,
                                      criteriaMinMax, epsilon,
                                      alternativesRanks = alternativesRanks)

---

**AHP**  

*Analytic Hierarchy Process (AHP) method*

**Description**

AHP is a multi-criteria decision analysis method which was originally developed by Thomas L. Saaty in 1970s.

**Usage**

AHP(criteriaWeightsPairwiseComparisons, alternativesPairwiseComparisonsList)

**Arguments**

criteriaWeightsPairwiseComparisons

Matrix or data frame containing the pairwise comparison matrix for the criteria weights. Lines and columns are named according to the IDs of the criteria.

alternativesPairwiseComparisonsList

A list containing a matrix or data frame of pairwise comparisons (comparing alternatives) for each criterion. The elements of the list are named according to the IDs of the criteria. In each matrix, the lines and the columns are named according to the IDs of the alternatives.
applyPiecewiseLinearValueFunctionsOnPerformanceTable

Value

The function returns a vector containing the AHP score for each alternative.

References


Examples

```r
style <- t(matrix(c(1,0.25,4,1/6,4,1,4,0.25,0.25,0.25,1,0.2,6,4,5,1),
nrow=4,ncol=4))
colnames(style) = c("Corsa","Clio","Fiesta","Sandero")
rownames(style) = c("Corsa","Clio","Fiesta","Sandero")

reliability <- t(matrix(c(1,2,5,1,0.5,1,3,2,0.2,1/3,1,0.25,1,0.5,4,1),
nrow=4,ncol=4))
colnames(reliability) = c("Corsa","Clio","Fiesta","Sandero")
rownames(reliability) = c("Corsa","Clio","Fiesta","Sandero")

fuel <- t(matrix(c(1,2,4,1,0.5,1,3,2,0,25,1/3,1,0.2,1,0.5,5,1),nrow=4,ncol=4))
colnames(fuel) = c("Corsa","Clio","Fiesta","Sandero")
rownames(fuel) = c("Corsa","Clio","Fiesta","Sandero")

alternativesPairwiseComparisonsList <- list(style=style,
                                           reliability=reliability,
                                           fuel=fuel)
criteriaWeightsPairwiseComparisons <- t(matrix(c(1,0.5,3,2,1,4,1/3,0.25,1),
nrow=3,ncol=3))
colnames(criteriaWeightsPairwiseComparisons) = c("style","reliability","fuel")
rownames(criteriaWeightsPairwiseComparisons) = c("style","reliability","fuel")

overall1 <- AHP(criteriaWeightsPairwiseComparisons,
                 alternativesPairwiseComparisonsList)
```

Description

Transforms a performance table via given piecewise linear value functions.
Usage

applyPiecewiseLinearValueFunctionsOnPerformanceTable(valueFunctions, performanceTable, alternativesIDs = NULL, criteriaIDs = NULL)

Arguments

valueFunctions  A list containing, for each criterion, the piecewise linear value functions defined by the coordinates of the break points. Each value function is defined by a matrix of breakpoints, where the first row corresponds to the abscissa (row labelled "x") and where the second row corresponds to the ordinate (row labelled "y").

performanceTable 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).

alternativesIDs Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs Vector containing IDs of criteria, according to which the data should be filtered.

Value

The function returns a performance table which has been transformed through the given value functions.

Examples

# the value functions
v<-list(
  Price = array(c(30, 0, 16, 0, 2, 0.0875),
    dim=c(2,3), dimnames = list(c("x", "y"), NULL)),
  Time = array(c(40, 0, 30, 0, 20, 0.025, 10, 0.9),
    dim = c(2, 4), dimnames = list(c("x", "y"), NULL)),
  Comfort = array(c(0, 0, 1, 0, 2, 0.0125, 3, 0.0125),
    dim = c(2, 4), dimnames = list(c("x", "y"), NULL)))

# the performance table
performanceTable <- rbind(
  c(3,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
assignAlternativesToCategoriesByThresholds

Assign alternatives to categories according to thresholds.

Description
Assign alternatives to categories according to thresholds representing the lower bounds of the categories.

Usage
assignAlternativesToCategoriesByThresholds(alternativesScores, categoriesLowerBounds, alternativesIDs = NULL, categoriesIDs = NULL)

Arguments
- alternativesScores: Vector representing the overall scores of the alternatives. The elements are named according to the IDs of the alternatives.
- categoriesLowerBounds: Vector containing the lower bounds of the categories. An alternative is assigned to a category if its score is higher or equal to the lower bound of the category, and strictly lower to the lower bound of the category above.
- alternativesIDs: Vector containing IDs of alternatives, according to which the data should be filtered.
- categoriesIDs: Vector containing IDs of categories, according to which the data should be filtered.

Value
The function returns a vector containing the assignments of the alternatives to the categories.

Examples
# the separation threshold
epsilon <- 0.05

# the performance table
assignAlternativesToCategoriesByThresholds

`performanceTable <- rbind(
  c(3,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))`

rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")

# ranks of the alternatives
alternativesAssignments <- c("good","medium","medium","bad","bad")
names(alternativesAssignments) <- row.names(performanceTable)

# criteria to minimize or maximize
criteriaMinMax <- c("min","min","max")
names(criteriaMinMax) <- colnames(performanceTable)

# number of break points for each criterion
criteriaNumberOfBreakPoints <- c(3,4,4)
names(criteriaNumberOfBreakPoints) <- colnames(performanceTable)

# ranks of the categories
categoriesRanks <- c(1,2,3)
names(categoriesRanks) <- c("good","medium","bad")

x <- UTADIS(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, alternativesAssignments, categoriesRanks, 0.1)
npt <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(x$valueFunctions, performanceTable)
scores <- weightedSum(npt, c(1,1,1))

# add a lower bound for the "bad" category
lbs <- c(x$categoriesLBs,0)
names(lbs) <- c(names(x$categoriesLBs),"bad")

assignments <- assignAlternativesToCategoriesByThresholds(scores, lbs)
ELECTREIII\textit{Distillation}

\textit{ELECTRE III ranking}

\textbf{Description}
This function computes the two ELECTRE III distillations, or rankings.

\textbf{Usage}
\begin{verbatim}
ELECTREIII\textit{Distillation}(\text{performanceTable, criteriaWeights, minMaxcriteria, preferenceThresholds, indifferenceThresholds, vetoThresholds})
\end{verbatim}

\textbf{Arguments}
\begin{itemize}
\item \textbf{performanceTable} 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).
\item \textbf{criteriaWeights} Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.
\item \textbf{minMaxcriteria} Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.
\item \textbf{preferenceThresholds} Vector containing preference thresholds for each criterion.
\item \textbf{indifferenceThresholds} Vector containing indifferences thresholds for each criterion.
\item \textbf{vetoThresholds} Vector containing veto thresholds for each criterion.
\end{itemize}

\textbf{Value}
The function returns two lists, one for each distillation.

\textbf{Examples}
\begin{verbatim}
performanceTable <- rbind(
c(10, 20, 5, 10, 16),
c(0, 5, 5, 16, 10),
c(0, 10, 0, 16, 7),
c(20, 5, 10, 10, 13),
c(20, 10, 15, 10, 13),
\end{verbatim}
c(20,10,20,13,13))
rownames(performanceTable) <-c("P1","P2","P3","P4","P5","P6")
colnames(performanceTable) <-c("CRIT1","CRIT2","CRIT3","CRIT4","CRIT5")
## vector indicating the direction of the criteria evaluation.
minMaxcriteria <-c("max","max","max","max","max")
names(minMaxcriteria) <- colnames(performanceTable)
## criteriaWeights vector
criteriaWeights <- c(3,2,3,1,1)
names(criteriaWeights) <- colnames(performanceTable)
indifferenceThresholds<-c(3,3,3,3,3)
names(indifferenceThresholds) <- colnames(performanceTable)
preferenceThresholds<-c(5,5,5,5,5)
names(preferenceThresholds) <- colnames(performanceTable)
vetoThresholds<-c(11,11,11,11,11)
names(vetoThresholds) <- colnames(performanceTable)
ELECTREIIIDistillation(performanceTable,criteriaWeights,minMaxcriteria,
preferenceThresholds,indifferenceThresholds,
vetoThresholds)

LPDMRSort

MRSort that takes into account large performance differences.

Description

MRSort is a simplified ElectreTRI method that uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. LPDMRSort considers both a binary discordance and a binary concordance conditions including several interactions between them.

Usage

LPDMRSort(performanceTable, categoriesLowerProfiles, categoriesRanks,
criteriaWeights, criteriaMinMax, majorityThreshold,
criteriaVetos = NULL, criteriaDictators = NULL,
majorityRule = "M", alternativesIDs = NULL,
criteriaIDs = NULL, categoriesIDs = NULL)

Arguments

performanceTable
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).

categoriesLowerProfiles
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.
categoriesRanks
A vector containing the ranks of the categories (1 for the best, with higher values for increasingly less preferred categories). The vector needs to be named with the categories names, whereas the ranks need to be a range of values from 1 to the number of categories.

criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

majorityThreshold
The cut threshold for the concordance condition. Should be at least half of the sum of the weights.

criteriaVetos
Matrix containing in each row a vector defining the veto values for the lower profile of the category. NA values mean that no veto is defined. A veto threshold for criterion i and category k represents the performance below which an alternative is forbidden to outrank the lower profile of category k, and thus is forbidden to be assigned to the category k. The rows are named according to the categories, whereas the columns are named according to the criteria.

criteriaDictators
Matrix containing in each row a vector defining the dictator values for the lower profile of the category. NA values mean that no veto is defined. A dictator threshold for criterion i and category k represents the performance above which an alternative is guaranteed to outrank the lower profile of category k, and thus may no be assigned below category k. The rows are named according to the categories, whereas the columns are named according to the criteria.

majorityRule
String denoting how the vetoes and dictators are combined in order to form the assignment rule. The values to choose from are "M", "V", "D", "v", "d", "dV", "Dv", "dv". "M" corresponds to using only the majority rule without vetoes or dictators, "V" considers only the vetoes, "D" only the dictators, "v" is like "V" only that a dictator may invalidate a veto, "d" is like "D" only that a veto may invalidate a dictator, "dV" is like "V" only that if there is no veto we may then consider the dictator, "Dv" is like "D" only that when there is no dictator we may consider the vetoes, while finally "dv" is identical to using both dictator and vetoes only that when both are active they invalidate each other, so the majority rule is considered in that case.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.
categoriesIDs
Vector containing IDs of categories, according to which the data should be filtered.

Value
The function returns a vector containing the assignments of the alternatives to the categories.
LPDMRSort

References


Examples

# the performance table

performanceTable <- rbind(c(10,10,9), c(10,9,10), c(9,10,10), c(9,9,10),
                         c(9,10,9), c(10,9,9), c(10,10,7), c(10,7,10),
                         c(7,10,10), c(9,10,9), c(9,10,7), c(9,7,10),
                         c(7,10,17), c(10,17,7), c(17,7,10), c(7,17,10),
                         c(7,10,17), c(10,7,17), c(7,9,17), c(9,17,7),
                         c(17,7,9), c(7,17,9), c(17,9,7), c(9,7,17))

profilesPerformances <- rbind(c(10,10,10),c(0,0,0))

vetoPerformances <- rbind(c(7,7,7),c(0,0,0))

dictatorPerformances <- rbind(c(17,17,17),c(0,0,0))

rownames(performanceTable) <- c("a1", "a2", "a3", "a4", "a5", "a6", "a7",
                            "a8", "a9", "a10", "a11", "a12", "a13",
                            "a14", "a15", "a16", "a17", "a18", "a19",
                            "a20", "a21", "a22", "a23", "a24")

rownames(profilesPerformances) <- c("P","F")

rownames(vetoPerformances) <- c("P","F")

rownames(dictatorPerformances) <- c("P","F")

colnames(performanceTable) <- c("c1","c2","c3")

colnames(profilesPerformances) <- c("c1","c2","c3")

colnames(vetoPerformances) <- c("c1","c2","c3")

colnames(dictatorPerformances) <- c("c1","c2","c3")

lambda <- 0.5

weights <- c(1/3,1/3,1/3)

names(weights) <- c("c1","c2","c3")

categoriesRanks <-c(1,2)
LPDMRSortIdentifyIncompatibleAssignments

Identifies all sets of assignment examples which are incompatible with the MRSort sorting method extended to handle large performance differences.

Description

MRSort is a simplified ElectreTRI method that uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. LPDMRSort considers both a binary discordance and a binary concordance conditions including several interactions between them. This function outputs all (or a fixed number of) sets of incompatible assignment examples ranging in size from the minimal size and up to a given threshold. The retrieved sets are also not contained in each other.
Usage

LPDMRSortIdentifyIncompatibleAssignments(performanceTable, assignments, categoriesRanks, criteriaMinMax, majorityRule = "M", incompatibleSetsLimit = 100, largerIncompatibleSetsMargin = 0, alternativesIDs = NULL, criteriaIDs = NULL, solver="glpk", cplexIntegralityTolerance = NULL, cplexThreads = NULL)

Arguments

performanceTable
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).

assignments
Vector containing the assignments (IDs of the categories) of the alternatives to the categories. The elements are named according to the alternatives.

categoriesRanks
Vector containing the ranks of the categories. The elements are named according to the IDs of the categories.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

majorityRule
String denoting how the vetoes and dictators are combined in order to form the assignment rule. The values to choose from are "M", "V", "D", "v", "d", "dV", "Dv", "dv". "M" corresponds to using only the majority rule without vetoes or dictators, "V" considers only the vetoes, "D" only the dictators, "v" is like "V" only that a dictator may invalidate a veto, "d" is like "D" only that a veto may invalidate a dictator, "dV" is like "V" only that if there is no veto we may then consider the dictator, "Dv" is like "D" only that when there is no dictator we may consider the vetoes, while finally "dv" is identical to using both dictator and vetoes only that when both are active they invalidate each other, so the majority rule is considered in that case.

incompatibleSetsLimit
Positive integer denoting the upper limit of the number of sets to be retrieved.

largerIncompatibleSetsMargin
Positive integer denoting whether sets larger than the minimal size should be retrieved, and by what margin. For example, if this is 0 then only sets of the minimal size will be retrieved, if this is 1 then sets also larger by 1 element will be retrieved.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.
LPDMRSortIdentifyIncompatibleAssignments

criteriaIDs  Vector containing IDs of criteria, according to which the data should be filtered.
solver  String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.
cplexIntegralTolerance  If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).
cplexThreads  If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

Value

The function returns NULL if there is a problem, or a list containing a list of incompatible sets of alternatives as vectors and the status of the execution.

References


Examples

# the performance table

performanceTable <- rbind(c(10,10,9), c(9,10,10), c(9,9,10), c(10,9,9), c(10,7,10), c(10,10,7), c(9,9,17), c(9,17,9), c(17,9,9), c(7,10,17), c(7,7,17), c(7,17,7), c(7,17,10), c(7,10,17), c(7,7,17), c(7,9,17), c(9,9,17), c(9,17,7), c(7,9,7), c(7,7,17), c(7,7,7))

rownames(performanceTable) <- c("a1", "a2", "a3", "a4", "a5", "a6", "a7", "a8", "a9", "a10", "a11", "a12", "a13", "a14", "a15", "a16", "a17", "a18", "a19", "a20", "a21", "a22", "a23", "a24", "a25")

colnames(performanceTable) <- c("c1","c2","c3")

LPDMRSortIdentifyUsedDictatorProfiles

```r
```

colnames(assignments) <- rownames(performanceTable)

categoriesRanks <- c(1, 2)

names(categoriesRanks) <- c("P", "F")

criteriaMinMax <- c("max", "max", "max")

names(criteriaMinMax) <- colnames(performanceTable)

majorityRules <- c("V", "D", "v", "d", "vD", "Dv", "dv")

for(i in 1:1)# change to 7 in order to perform all tests {
  incompatibleAssignmentsSets <- LPDMRSortIdentifyIncompatibleAssignments(
    performanceTable, assignments[i,],
    categoriesRanks, criteriaMinMax,
    majorityRule = majorityRules[i])

  filteredAlternativesIDs <- setdiff(rownames(performanceTable),
    incompatibleAssignmentsSets[[1]][[1]])

  x <- LPDMRSortInferenceExact(performanceTable, assignments[i,],
    categoriesRanks, criteriaMinMax,
    majorityRule = majorityRules[i],
    readableWeights = TRUE,
    readableProfiles = TRUE,
    minmaxLPD = TRUE,
    alternativesIDs = filteredAlternativesIDs)

  ElectreAssignments <- LPDMRSort(performanceTable, x$profilesPerformances,
    categoriesRanks,
    x$weights, criteriaMinMax, x$lambda,
    criteriaVetos = x$vetoPerformances,
    criteriaDictators = x$dictatorPerformances,
    majorityRule = majorityRules[i],
    alternativesIDs = filteredAlternativesIDs)

  print(all(ElectreAssignments == assignments[i, filteredAlternativesIDs]))
}
```
LPDMRSortIdentifyUsedDictatorProfiles

Identify dictator profiles evaluations that have an impact on the final assignments of MRSort with large performance differences

Description

MRSort is a simplified ELECTRE-TRI approach which assigns alternatives to a set of ordered categories using delimiting profiles evaluations. In this case, we also take into account large performance differences. This method is used to identify which dictator profiles evaluations have an impact on the final assignment of at least one of the input alternatives.

Usage

LPDMRSortIdentifyUsedDictatorProfiles(performanceTable, assignments, categoriesRanks, criteriaMinMax, majorityThreshold, criteriaWeights, profilesPerformances, dictatorPerformances, vetoPerformances = NULL, majorityRule = "D", alternativesIDs = NULL, criteriaIDs = NULL)

Arguments

performanceTable
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).

assignments
A vector containing the category to which each alternative is assigned. The vector needs to be named using the alternatives IDs.

categoriesRanks
A vector containing the ranks of the categories (1 for the best, with higher values for increasingly less preferred categories). The vector needs to be named with the categories names, whereas the ranks need to be a range of values from 1 to the number of categories.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

majorityThreshold
The majority threshold needed to determine when a coalition of criteria is sufficient in order to validate an outranking relation.

criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.
profilesPerformances
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.
dictatorPerformances
Matrix containing in each row a vector defining the dictator values for the lower profile of the category. NA values mean that no dictator is defined. A dictator threshold for criterion i and category k represents the performance above which an alternative outranks the lower profile of category k regardless of the size of the coalition of criteria in favor of this statement. The rows are named according to the categories, whereas the columns are named according to the criteria.
vetoPerformances
Matrix containing in each row a vector defining the veto values for the lower profile of the category. NA values mean that no veto is defined. A veto threshold for criterion i and category k represents the performance below which an alternative is forbidden to outrank the lower profile of category k, and thus is forbidden to be assigned to the category k. The rows are named according to the categories, whereas the columns are named according to the criteria. By default no veto profiles are needed.
majorityRule
String denoting how the vetoes and dictators are combined in order to form the assignment rule. The values to choose from are "D", "v", "d", "dV", "Dv", "dv". "D" considers only the dictators, "v" is like "V" only that a dictator may invalidate a veto, "d" is like "D" only that a veto may invalidate a dictator, "dV" is like "V" only that if there is no veto we may then consider the dictator, "Dv" is like "D" only that when there is no dictator we may consider the vetoes, while finally "dv" is identical to using both dictator and vetoes only that when both are active they invalidate each other, so the majority rule is considered in that case.
alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.
criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

Value
The function returns a matrix containing TRUE/FALSE indicators for each evaluation of the veto profiles.

Examples
# the performance table

performanceTable <- rbind(
  c(1,27,1),
  c(6,20,1),
  c(2,20,0),
  c(6,40,0),
  c(30,10,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")

# lower profiles of the categories (best category in the first position of the list)
categoriesLowerProfiles <- rbind(c(3, 11, 3),c(7, 25, 2),c(NA,NA,NA))
colnames(categoriesLowerProfiles) <- colnames(performanceTable)
rownames(categoriesLowerProfiles) <- c("Good","Medium","Bad")

# the order of the categories, 1 being the best
categoriesRanks <- c(1, 2, 3)
names(categoriesRanks) <- c("Good","Medium","Bad")

# criteria to minimize or maximize
criteriaMinMax <- c("min","min","max")
names(criteriaMinMax) <- colnames(performanceTable)

dictators
criteriaDictators <- rbind(c(1, 1, -1),c(1, 20, 0),c(NA, NA, NA))
colnames(criteriaDictators) <- colnames(performanceTable)
rownames(criteriaDictators) <- c("Good","Medium","Bad")

# vetos
criteriaVetos <- rbind(c(9, 50, 5),c(50, 50, 5),c(NA, NA, NA))
colnames(criteriaVetos) <- colnames(performanceTable)
rownames(criteriaVetos) <- c("Good","Medium","Bad")

# weights
criteriaWeights <- c(1/6, 3/6, 2/6)
names(criteriaWeights) <- colnames(performanceTable)

# assignments
assignments <- c("Good","Medium","Bad","Bad","Bad")

# LPDMRSortIdentifyUsedDictatorProfiles
used <- LPDMRSortIdentifyUsedDictatorProfiles(performanceTable, assignments, categoriesRanks, criteriaMinMax,
LPDMRSortIdentifyUsedVetoProfiles

Identify veto profiles evaluations that have an impact on the final assignments of MRSort with large performance differences

Description

MRSort is a simplified ELECTRE-TRI approach which assigns alternatives to a set of ordered categories using delimiting profiles evaluations. In this case, we also take into account large performance differences. This method is used to identify which veto profiles evaluations have an impact on the final assignment of at least one of the input alternatives.

Usage

LPDMRSortIdentifyUsedVetoProfiles(performanceTable, assignments, categoriesRanks, criteriaMinMax, majorityThreshold, criteriaWeights, profilesPerformances, vetoPerformances, dictatorPerformances = NULL, majorityRule = "V", alternativesIDs = NULL, criteriaIDs = NULL)

Arguments

performanceTable
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).

assignments
A vector containing the category to which each alternative is assigned. The vector needs to be named using the alternatives IDs.

categoriesRanks
A vector containing the ranks of the categories (1 for the best, with higher values for increasingly less preferred categories). The vector needs to be named with the categories names, whereas the ranks need to be a range of values from 1 to the number of categories.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.
majorityThreshold
The majority threshold needed to determine when a coalition of criteria is sufficient in order to validate an outranking relation.

criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

profilesPerformances
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.

vetoPerformances
Matrix containing in each row a vector defining the veto values for the lower profile of the category. NA values mean that no veto is defined. A veto threshold for criterion i and category k represents the performance below which an alternative is forbidden to outrank the lower profile of category k, and thus is forbidden to be assigned to the category k. The rows are named according to the categories, whereas the columns are named according to the criteria.

dictatorPerformances
Matrix containing in each row a vector defining the dictator values for the lower profile of the category. NA values mean that no dictator is defined. A dictator threshold for criterion i and category k represents the performance above which an alternative outranks the lower profile of category k regardless of the size of the coalition of criteria in favor of this statement. The rows are named according to the categories, whereas the columns are named according to the criteria. By default no dictator profiles are needed for this method.

majorityRule
String denoting how the vetoes and dictators are combined in order to form the assignment rule. The values to choose from are "V", "v", "d", "dV", "Dv", "dv". "V" considers only the vetoes, "v" is like "V" only that a dictator may invalidate a veto, "d" is like "D" only that a veto may invalidate a dictator, "dV" is like "V" only that if there is no veto we may then consider the dictator, "Dv" is like "D" only that when there is no dictator we may consider the vetoes, while finally "dv" is identical to using both dictator and vetoes only that when both are active they invalidate each other, so the majority rule is considered in that case.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

Value
The function returns a matrix containing TRUE/FALSE indicators for each evaluation of the veto profiles.

Examples
# the performance table
performanceTable <- rbind(
  c(1,27,1),
  c(6,20,1),
  c(2,20,0),
  c(6,40,0),
  c(30,10,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")

# lower profiles of the categories (best category in the first position of the list)
categoriesLowerProfiles <- rbind(c(3, 11, 3),c(7, 25, 2),c(NA,NA,NA))
colnames(categoriesLowerProfiles) <- colnames(performanceTable)
rownames(categoriesLowerProfiles) <- c("Good","Medium","Bad")

# the order of the categories, 1 being the best
categoriesRanks <- c(1,2,3)
names(categoriesRanks) <- c("Good","Medium","Bad")

# criteria to minimize or maximize
criteriaMinMax <- c("min","min","max")
names(criteriaMinMax) <- colnames(performanceTable)

# dictators
criteriaDictators <- rbind(c(1, 1, -1),c(1, 20, 0),c(NA,NA,NA))
colnames(criteriaDictators) <- colnames(performanceTable)
rownames(criteriaDictators) <- c("Good","Medium","Bad")

# vetos
criteriaVetos <- rbind(c(9, 50, 5),c(50, 50, 5),c(NA,NA,NA))
colnames(criteriaVetos) <- colnames(performanceTable)
rownames(criteriaVetos) <- c("Good","Medium","Bad")

# weights
criteriaWeights <- c(1/6,3/6,2/6)
names(criteriaWeights) <- colnames(performanceTable)

# assignments
assignments <- c("Good","Medium","Bad","Bad","Bad")

# LPDMRSortIdentifyUsedVetoProfiles
used<-LPDMRSortIdentifyUsedVetoProfiles(performanceTable, assignments,
categoriesRanks, criteriaMinMax,
0.5, criteriaWeights,
categoriesLowerProfiles,
criteriaVetos,
criteriaDictators,
"dv")

LPDMRSortInferenceApprox

Identification of profiles, weights, majority threshold, veto and dictator thresholds for LPDMRSort using a genetic algorithm.

Description

MRSort is a simplified ElectreTRI method that uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. LPDMRSort considers both a binary discordance and a binary concordance conditions including several interactions between them. The identification of the profiles, weights, majority threshold and veto thresholds is done by taking into account assignment examples.

Usage

LPDMRSortInferenceApprox(performanceTable, criteriaMinMax, categoriesRanks, assignments,
majorityRules = c("M","V","D","v","d","Dv","dv"),
alternativesIDs = NULL, criteriaIDs = NULL,
timeLimit = 60, populationSize = 20, mutationProb = 0.1)

Arguments

- **performanceTable**: 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).
- **assignments**: Vector containing the assignments (IDs of the categories) of the alternatives to the categories. The elements are named according to the alternatives.
- **categoriesRanks**: Vector containing the ranks of the categories. The elements are named according to the IDs of the categories.
- **criteriaMinMax**: Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.
majorityRules A vector containing the different type of majority rules to be considered ("M", "V", "D", "v", "d", "dV", "Dv", "dv"). "M" corresponds to using only the majority rule without vetoes or dictators, "V" considers only the vetoes, "D" only the dictators, "v" is like "V" only that a dictator may invalidate a veto, "d" is like "D" only that a veto may invalidate a dictator, "dV" is like "V" only that if there is no veto we may then consider the dictator, "Dv" is like "D" only that when there is no dictator we may consider the vetoes, while finally "dv" is identical to using both dictator and vetoes only that when both are active they invalidate each other, so the majority rule is considered in that case.

alternativesIDs Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs Vector containing IDs of criteria, according to which the data should be filtered.

timeLimit Allows to fix a time limit of the execution, in seconds (default 60).

populationSize Allows to change the size of the population used by the genetic algorithm (default 20).

mutationProb Allows to change the mutation probability used by the genetic algorithm (default 0.1).

Value

The function returns a list containing:

majorityThreshold The inferred majority threshold (single numeric value).

criteriaWeights The inferred criteria weights (a vector named with the criteria IDs).

majorityRule A string corresponding to the inferred majority rule (one of "M", "V", "D", "v", "d", "dV", "Dv", "dv").

profilesPerformances The inferred category limits (a matrix with the column names given by the criteria IDs and the rownames given by the upper categories each profile delimits).

vetoPerformances The inferred vetoes (a matrix with the column names given by the criteria IDs and the rownames given by the categories to which each profile applies).

dictatorPerformances The inferred dictators (a matrix with the column names given by the criteria IDs and the rownames given by the categories to which each profile applies).

fitness The classification accuracy of the inferred model (from 0 to 1).

References


no reference yet for the algorithmic approach; one should become available in 2018
Examples

```r
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),c(10,9,7),
                          c(10,7,10),c(7,10,10),c(9,17,9),c(9,9,17),c(9,17,7),
                          c(7,10,17),c(10,17,7),c(7,17,10),c(17,10,7),c(17,10,7),
                          c(7,9,17),c(9,17,7),c(17,7,9),c(17,9,7),c(17,9,7))
rownames(performanceTable) <- c("a1", "a2", "a3", "a4", "a5", "a6", "a7", "a8", "a9", "a10", "a11",
             "a12", "a13", "a14", "a15", "a16", "a17", "a18", "a19", "a20",
             "a21", "a22", "a23", "a24")

colnames(performanceTable) <- c("c1", "c2", "c3")
                "F", "F", "F", "F")
names(assignments) <- rownames(performanceTable)
categoriesRanks <- c(1,2)
names(categoriesRanks) <- c("P", "F")
criteriaMinMax <- c("max", "max", "max")
names(criteriaMinMax) <- colnames(performanceTable)
set.seed(1)
x<-LPDMRSortInferenceApprox(performanceTable, criteriaMinMax, categoriesRanks, assignments,
majorityRules = c("dV","Dv","dv"),
timelimit = 180, populationSize = 30,
alternativesIDs = c("a1","a2","a3","a4","a5","a6","a7"))
```

Description

MRSort is a simplified ElectreTRI method that uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. LPDMRSort considers both a binary discordance and a binary concordance conditions including several interactions between them. The identification of the profiles, weights, majority threshold and veto and dictator thresholds are done by taking into account assignment examples.

Usage

```
LPDMRSortInferenceExact(performanceTable, assignments,
```
categoriesRanks, criteriaMinMax,
majorityRule = "M", readableWeights = FALSE,
readableProfiles = FALSE, minmaxLPD = FALSE,
alternativesIDs = NULL, criteriaIDs = NULL,
solver="glpk",cplexTimeLimit = NULL,
cplexIntegralityTolerance = NULL, cplexThreads = NULL)

Arguments

- **performanceTable**: 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).

- **assignments**: Vector containing the assignments (IDs of the categories) of the alternatives to the categories. The elements are named according to the alternatives.

- **categoriesRanks**: Vector containing the ranks of the categories. The elements are named according to the IDs of the categories.

- **criteriaMinMax**: Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **majorityRule**: String denoting how the vetoes and dictators are combined in order to form the assignment rule. The values to choose from are "M", "V", "D", "v", "d", "dV", "Dv", "dv". "M" corresponds to using only the majority rule without vetoes or dictators, "V" considers only the vetoes, "D" only the dictators, "v" is like "V" only that a dictator may invalidate a veto, "d" is like "D" only that a veto may invalidate a dictator, "dV" is like "V" only that if there is no veto we may then consider the dictator, "Dv" is like "D" only that when there is no dictator we may consider the vetoes, while finally "dv" is identical to using both dictator and vetoes only that when both are active they invalidate each other, so the majority rule is considered in that case.

- **readableWeights**: Boolean parameter indicating whether the weights are to be spaced more evenly or not.

- **readableProfiles**: Boolean parameter indicating whether the profiles are to be spaced more evenly or not.

- **minmaxLPD**: Boolean parameter indicating whether the veto thresholds are to be minimized (or maximized if lower criteria values are preferred) while the dictator thresholds are to be maximized (or minimized if lower criteria values are preferred).

- **alternativesIDs**: Vector containing IDs of alternatives, according to which the data should be filtered.

- **criteriaIDs**: Vector containing IDs of criteria, according to which the data should be filtered.

- **solver**: String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.
cplexTimeLimit  If the cplex solver is used, allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to the default value of cplex).

cplexIntegralityTolerance  If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).

cplexThreads  If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

Value

The function returns a list structured as follows:

lambda  The majority threshold.

weights  A vector containing the weights of the criteria. The elements are named according to the criteria IDs.

profilesPerformances  A matrix containing the lower profiles of the categories. The columns are named according to the criteria, whereas the rows are named according to the categories. The lower profile of the lower category can be considered as a dummy profile.

vetoPerformances  A matrix containing the veto profiles of the categories. The columns are named according to the criteria, whereas the rows are named according to the categories. The veto profile of the lower category can be considered as a dummy profile.

solverStatus  The solver status as given by glpk or cplex.

References


Examples

# the performance table

performanceTable <- rbind(c(10,10,9), c(10,9,10), c(9,10,10), c(9,9,10),
                         c(9,10,9), c(10,9,9), c(10,10,7), c(10,7,10),
                         c(7,10,10), c(9,9,17), c(9,17,9), c(17,9,9),
                         c(7,10,17), c(10,17,7), c(17,7,10), c(7,17,10),
                         c(17,10,7), c(10,7,17), c(7,9,17), c(9,17,7),
                         c(17,7,9), c(7,17,9), c(17,9,7), c(9,17,7))

rownames(performanceTable) <- c("a1", "a2", "a3", "a4", "a5", "a6", "a7", "a8", "a9", "a10", "a11", "a12", "a13", "a14", "a15", "a16", "a17", "a18", "a19", "a20")
colnames(performanceTable) <- c("a1", "a2", "a3", "a4")
categoriesRanks <- c(1,2)
names(categoriesRanks) <- c("P", "F")
criteriaMinMax <- c("max", "max", "max")
names(criteriaMinMax) <- colnames(performanceTable)


colnames(assignments) <- rownames(performanceTable)
majorityRules <- c("V", "D", "v", "d", "dV", "Dv", "dv")

for(i in 1:1) # change to 7 in order to perform all tests {
  x <- LPDMRSortInferenceExact(performanceTable, assignments[i,,]
                            categoriesRanks, criteriaMinMax,
                            majorityRule = majorityRules[i],
                            readableWeights = TRUE,
                            readableProfiles = TRUE,
                            minmaxLPD = TRUE)

  ElectreAssignments <- LPDMRSort(performanceTable, x$profilesPerformances,
                                   categoriesRanks,
                                   x$weights, criteriaMinMax, x$lambda,
                                   criteriaVeto=x$vetoPerformances,
                                   criteriaDictator=x$dictatorPerformances,
                                   majorityRule = majorityRules[i])

  print(x)
  print(all(ElectreAssignments == assignments[i,,]))
}
MARE

Multi-Attribute Range Evaluations (MARE)

Description

MARE is a multi-criteria decision analysis method which was originally developed by Hodgett et al. in 2014.

Usage

MARE(performanceTableMin, performanceTable, performanceTableMax, criteriaWeights, criteriaMinMax, alternativesIDs = NULL, criteriaIDs = NULL)

Arguments

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

performanceTable
Matrix or data frame containing the most likely performance table. Each column corresponds to an alternative, and each row to a criterion. Columns (resp. rows) must be named according to the IDs of the alternatives (resp. criteria).

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

criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

Value

The function returns an element of type mare which contains the MARE scores for each alternative.
MRSort

References


Examples

```r
performanceTableMin <- t(matrix(c(78, 79, 87, 86, 74, 8, 90, 89, 74.5, 9, 20, 81, 30),
                                nrow=3, ncol=5, byrow=TRUE))
performanceTable <- t(matrix(c(80, 87, 86, 19, 8, 70, 74, 10, 90, 89, 75, 9, 33, 82, 30),
                              nrow=3, ncol=5, byrow=TRUE))
performanceTableMax <- t(matrix(c(81, 87, 95, 19, 8, 72, 74, 15, 90, 89, 75.5, 9, 36, 84, 30),
                              nrow=3, ncol=5, byrow=TRUE))
row.names(performanceTable) <- c("Yield", "Toxicity", "Cost", "Separation", "Odour")
colnames(performanceTable) <- c("Route One", "Route Two", "Route Three")
row.names(performanceTableMin) <- row.names(performanceTable)
colnames(performanceTableMin) <- colnames(performanceTable)
row.names(performanceTableMax) <- row.names(performanceTable)
colnames(performanceTableMax) <- colnames(performanceTable)
weights <- c(0.339, 0.077, 0.434, 0.127, 0.023)
names(weights) <- row.names(performanceTable)
criteriaMinMax <- c("max", "max", "max", "max", "max")
names(criteriaMinMax) <- row.names(performanceTable)
overall1 <- MARE(performanceTableMin,
                 performanceTable,
                 performanceTableMax,
                 weights,
                 criteriaMinMax)
overall2 <- MARE(performanceTableMin,
                 performanceTable,
                 performanceTableMax,
                 weights,
                 criteriaMinMax,
                 alternativesIDs = c("Route Two", "Route Three"),
                 criteriaIDs = c("Yield", "Toxicity", "Cost", "Separation"))
```

Description

This simplification of the Electre TRI method uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. Only a binary discordance condition is considered, i.e. a veto forbids an outranking in any possible concordance situation, or not.
Usage

```r
MRSort(performanceTable, categoriesLowerProfiles, categoriesRanks, criteriaWeights, criteriaMinMax, majorityThreshold, criteriaVetos = NULL, alternativesIDs = NULL, criteriaIDs = NULL, categoriesIDs = NULL)
```

Arguments

- **performanceTable**: 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).

- **categoriesLowerProfiles**: 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.

- **categoriesRanks**: A vector containing the ranks of the categories (1 for the best, with higher values for increasingly less preferred categories). The vector needs to be named with the categories names, whereas the ranks need to be a range of values from 1 to the number of categories.

- **criteriaWeights**: Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

- **criteriaMinMax**: Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **majorityThreshold**: The cut threshold for the concordance condition. Should be at least half of the sum of the weights.

- **criteriaVetos**: Matrix containing in each row a vector defining the veto values for the lower profile of the category. NA values mean that no veto is defined. A veto threshold for criterion i and category k represents the performance below which an alternative is forbidden to outrank the lower profile of category k, and thus is forbidden to be assigned to the category k. The rows are named according to the categories, whereas the columns are named according to the criteria.

- **alternativesIDs**: Vector containing IDs of alternatives, according to which the data should be filtered.

- **criteriaIDs**: Vector containing IDs of criteria, according to which the data should be filtered.

- **categoriesIDs**: Vector containing IDs of categories, according to which the data should be filtered.
Value

The function returns a vector containing the assignments of the alternatives to the categories.

References


Examples

# the performance table
performanceTable <- rbind(
  c(1,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")

# lower profiles of the categories
# (best category in the first position of the list)
categoriesLowerProfiles <- rbind(c(3, 11, 3),c(7, 25, 2),c(NA,NA,NA))
colnames(categoriesLowerProfiles) <- colnames(performanceTable)
rownames(categoriesLowerProfiles)<-c("Good","Medium","Bad")

# the order of the categories, 1 being the best
categoriesRanks <-c(1,2,3)
names(categoriesRanks) <- c("Good","Medium","Bad")

# criteria to minimize or maximize
criteriaMinMax <- c("min","min","max")
names(criteriaMinMax) <- colnames(performanceTable)

# vetos
criteriaVetos <- rbind(c(10, NA, NA),c(NA, NA, 1),c(NA,NA,NA))
colnames(criteriaVetos) <- colnames(performanceTable)
rownames(criteriaVetos) <- c("Good","Medium","Bad")
MRSortIdentifyIncompatibleAssignments

Identifies all sets of assignment examples which are incompatible with the MRSort method.

Description

This MRSort method, which is a simplification of the Electre TRI method, uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. Only a binary discordance condition is considered, i.e. a veto forbids an outranking in any possible concordance situation, or not. This function outputs for all (or a fixed number of) sets of incompatible assignment examples ranging in size from the minimal size and up to a given threshold. The retrieved sets are also not contained in each other.

Usage

MRSortIdentifyIncompatibleAssignments(performanceTable, assignments, categoriesRanks, criteriaMinMax, veto = FALSE, incompatibleSetsLimit = 100,
largerIncompatibleSetsMargin = 0,
alternativesIDs = NULL,
criteriaIDs = NULL,
solver="glpk",
cplexIntegralityTolerance = NULL,
cplexThreads = NULL)

Arguments

performanceTable
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).

assignments
Vector containing the assignments (IDs of the categories) of the alternatives to
the categories. The elements are named according to the alternatives.

categoriesRanks
Vector containing the ranks of the categories. The elements are named according
to the IDs of the categories.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp.
"max") indicates that the criterion has to be minimized (maximized). The ele-
ments are named according to the IDs of the criteria.

veto
Boolean parameter indicating whether veto profiles are being used by the model
or not.

incompatibleSetsLimit
Positive integer denoting the upper limit of the number of sets to be retrieved.

largerIncompatibleSetsMargin
Positive integer denoting whether sets larger than the minimal size should be
retrieved, and by what margin. For example, if this is 0 then only sets of the
minimal size will be retrieved, if this is 1 then sets also larger by 1 element will
be retrieved.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be fil-
tered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

solver
String specifying if the glpk solver (glpk) should be used, or the cplex (cplex)
solver. By default glpk. The cplex solver requires to install the cplex binary and
the cplex C API, as well as the cplexAPI R package.

cplexIntegralityTolerance
If the cplex solver is used, allows to fix a tolerance for integrality. By default
NULL (which corresponds to the default value of cplex).

cplexThreads
If the cplex solver is used, allows to the number of threads for the calculation.
By default NULL (which corresponds to the default value of cplex).

Value

The function returns NULL if there is a problem, or a list containing a list of incompatible sets of
alternatives as vectors and the status of the execution.
References


Examples

```r
performanceTable <- rbind(c(10,10,9), c(10,9,10), c(9,10,10), c(9,9,10),
                          c(9,10,9), c(10,9,9), c(10,10,7), c(10,7,10),
                          c(7,10,10), c(9,9,17), c(9,17,9), c(17,9,9),
                          c(7,10,17), c(10,17,7), c(17,7,10), c(7,17,10),
                          c(17,10,7), c(10,7,17), c(7,9,17), c(9,17,7),
                          c(17,7,9), c(7,17,9), c(17,9,7), c(9,7,17))
rownames(performanceTable) <- c("a1", "a2", "a3", "a4", "a5", "a6", "a7",
                      "a8", "a9", "a10", "a11", "a12", "a13",
                      "a14", "a15", "a16", "a17", "a18", "a19",
                      "a20", "a21", "a22", "a23", "a24")

colnames(performanceTable) <- c("c1","c2","c3")
                "F", "F", "F", "F", "F", "F", "F", "F",
                "F")
names(assignments) <- rownames(performanceTable)
categoriesRanks <-c(1,2)
names(categoriesRanks) <- c("P","F")
criteriaMinMax <- c("max","max","max")
names(criteriaMinMax) <- colnames(performanceTable)
incompatibleAssignmentsSets<-MRSortIdentifyIncompatibleAssignments(
  performanceTable, assignments,
  categoriesRanks, criteriaMinMax,
  veto = TRUE,
  alternativesIDs = c("a1","a2","a3","a4",
                      "a5","a6","a7","a8","a9","a10"))

print(incompatibleAssignmentsSets)

filteredAlternativesIDs <- setdiff(c("a1","a2","a3","a4",
  "a5","a6","a7","a8","a9"),
  incompatibleAssignmentsSets[[1]][[1]])

print(filteredAlternativesIDs)
x<-MRSortInferenceExact(performanceTable, assignments, categoriesRanks,
  criteriaMinMax, veto = TRUE,
```
MRSortIdentifyUsedVetoProfiles

Identify veto profiles evaluations that have an impact on the final assignments of MRSort

Description

MRSort is a simplified ELECTRE-TRI approach which assigns alternatives to a set of ordered categories using delimiting profiles evaluations. In addition, veto profiles may also be used in order to circumvent a sufficient majority coalition in favor of an alternative being assigned to a certain category. This method is used to identify which veto profiles evaluations have an impact on the final assignment of at least one of the input alternatives.

Usage

MRSortIdentifyUsedVetoProfiles(performanceTable, assignments, categoriesRanks, criteriaMinMax, majorityThreshold, criteriaWeights, profilesPerformances, vetoPerformances, alternativesIDs = NULL, criteriaIDs = NULL)

Arguments

- **performanceTable**: 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).
- **assignments**: A vector containing the category to which each alternative is assigned. The vector needs to be named using the alternatives IDs.
- **categoriesRanks**: A vector containing the ranks of the categories (1 for the best, with higher values for increasingly less preferred categories). The vector needs to be named with the categories names, whereas the ranks need to be a range of values from 1 to the number of categories.
criteriaMinMax  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

majorityThreshold

The majority threshold needed to determine when a coalition of criteria is sufficient in order to validate an outranking relation.

criteriaWeights

Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

profilesPerformances

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.

vetoPerformances

Matrix containing in each row a vector defining the veto values for the lower profile of the category. NA values mean that no veto is defined. A veto threshold for criterion i and category k represents the performance below which an alternative is forbidden to outrank the lower profile of category k, and thus is forbidden to be assigned to the category k. The rows are named according to the categories, whereas the columns are named according to the criteria.

alternativesIDs

Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs

Vector containing IDs of criteria, according to which the data should be filtered.

Value

The function returns a matrix containing TRUE/FALSE indicators for each evaluation of the veto profiles.

Examples

# the performance table

```
performanceTable <- rbind(
  c(1,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,10,3))

rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")

colnames(performanceTable) <- c("Price","Time","Comfort")
```

# lower profiles of the categories (best category in the first position of the list)

categoriesLowerProfiles <- rbind(c(3, 11, 3),c(7, 25, 2),c(NA,NA,NA))
colnames(categoriesLowerProfiles) <- colnames(performanceTable)
rownames(categoriesLowerProfiles) <- c("Good", "Medium", "Bad")

# the order of the categories, 1 being the best
categoriesRanks <- c(1, 2, 3)
names(categoriesRanks) <- c("Good", "Medium", "Bad")

# criteria to minimize or maximize
criteriaMinMax <- c("min", "min", "max")
names(criteriaMinMax) <- colnames(performanceTable)

# vetos
criteriaVetos <- rbind(c(9, 50, -1), c(50, 50, 0), c(NA, NA, NA))
colnames(criteriaVetos) <- colnames(performanceTable)
rownames(criteriaVetos) <- c("Good", "Medium", "Bad")

# weights
criteriaWeights <- c(1/6, 3/6, 2/6)
names(criteriaWeights) <- colnames(performanceTable)

# assignments
assignments <- c("Good", "Medium", "Bad", "Bad", "Bad")

# MRSortIdentifyUsedVetoProfiles
used <- MRSortIdentifyUsedVetoProfiles(performanceTable, assignments,
categoriesRanks, criteriaMinMax, 0.5, criteriaWeights,
categoriesLowerProfiles, criteriaVetos)

---

**MRSortInferenceApprox**  
*Identification of profiles, weights, majority threshold and veto thresholds for MRSort using a genetic algorithm.*

**Description**

MRSort is a simplification of the Electre TRI method that uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. Only a binary discordance condition is considered, i.e. a veto forbids an outranking in any possible concordance situation, or not.
The identification of the profiles, weights, majority threshold and veto thresholds are done by taking into account assignment examples.

Usage

```r
MRSortInferenceApprox(performanceTable, assignments, categoriesRanks, criteriaMinMax,
                  veto = FALSE, alternativesIDs = NULL, criteriaIDs = NULL,
                  timeLimit = 60, populationSize = 20, mutationProb = 0.1)
```

Arguments

- `performanceTable`: 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).
- `assignments`: Vector containing the assignments (IDs of the categories) of the alternatives to the categories. The elements are named according to the alternatives.
- `categoriesRanks`: Vector containing the ranks of the categories. The elements are named according to the IDs of the categories.
- `criteriaMinMax`: Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.
- `veto`: Boolean parameter indicating whether veto profiles are to be used or not.
- `alternativesIDs`: Vector containing IDs of alternatives, according to which the data should be filtered.
- `criteriaIDs`: Vector containing IDs of criteria, according to which the data should be filtered.
- `timeLimit`: Allows to fix a time limit of the execution, in seconds (default 60).
- `populationSize`: Allows to change the size of the population used by the genetic algorithm (default 20).
- `mutationProb`: Allows to change the mutation probability used by the genetic algorithm (default 0.1).

Value

The function returns a list containing:

- `majorityThreshold`: The inferred majority threshold (single numeric value).
- `criteriaWeights`: The inferred criteria weights (a vector named with the criteria IDs).
- `profilesPerformances`: The inferred category limits (a matrix with the column names given by the criteria IDs and the rownames given by the upper categories each profile delimits).
vetoPerformances

The inferred vetoes (a matrix with the column names given by the criteria IDs and the rownames given by the categories to which each profile applies).

fitness

The classification accuracy of the inferred model (from 0 to 1).

References


no reference yet for the algorithmic approach; one should become available in 2018

Examples

```r
performanceTable <- rbind(c(10,10,9), c(10,9,10), c(9,9,10), c(9,9,10), c(9,10,9), c(10,9,9),
                         c(10,10,7), c(10,7,10), c(9,9,17), c(9,9,17), c(7,9,17), c(7,9,17), c(7,9,17), c(7,9,17), c(9,7,17))
rownames(performanceTable) <- c("a1", "a2", "a3", "a4", "a5", "a6", "a7", "a8", "a9", "a10", "a11",
                         "a12", "a13", "a14", "a15", "a16", "a17", "a18", "a19", "a20",
                         "a21", "a22", "a23", "a24")
colnames(performanceTable) <- c("c1","c2","c3")
                 "F", "F", "F", "F", "F", "F", "F")
names(assignments) <- rownames(performanceTable)
categoriesRanks <- c(1,2)
names(categoriesRanks) <- c("P","F")
criteriaMinMax <- c("max","max","max")
names(criteriaMinMax) <- colnames(performanceTable)
set.seed(1)
x<-MRSortInferenceApprox(performanceTable, assignments, categoriesRanks,
criteriaMinMax, veto = TRUE,
alternativesIDs = c("a1","a2","a3","a4","a5","a6","a7"))
```

**MRSortInferenceExact**

Identification of profiles, weights and majority threshold for the MR-Sort sorting method using an exact approach.
**Description**

The MRSort method, a simplification of the Electre TRI method, uses the pessimistic assignment rule, without indifference or preference thresholds attached to criteria. Only a binary discordance condition is considered, i.e. a veto forbids an outranking in any possible concordance situation, or not. The identification of the profiles, weights and majority threshold are done by taking into account assignment examples.

**Usage**

```r
MRSortInferenceExact(performanceTable, assignments,
categoriesRanks, criteriaMinMax,
veto = FALSE, readableWeights = FALSE,
readableProfiles = FALSE,
alternativesIDs = NULL, criteriaIDs = NULL,
solver = "glpk",
cplexTimeLimit = NULL, cplexIntegralityTolerance = NULL, cplexThreads = NULL)
```

**Arguments**

- **performanceTable**
  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).

- **assignments**
  Vector containing the assignments (IDs of the categories) of the alternatives to the categories. The elements are named according to the alternatives.

- **categoriesRanks**
  Vector containing the ranks of the categories. The elements are named according to the IDs of the categories.

- **criteriaMinMax**
  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **veto**
  Boolean parameter indicating whether veto profiles are being used or not.

- **readableWeights**
  Boolean parameter indicating whether the weights are to be spaced more evenly or not.

- **readableProfiles**
  Boolean parameter indicating whether the profiles are to be spaced more evenly or not.

- **alternativesIDs**
  Vector containing IDs of alternatives, according to which the data should be filtered.

- **criteriaIDs**
  Vector containing IDs of criteria, according to which the data should be filtered.

- **solver**
  String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.

- **cplexTimeLimit**
  If the cplex solver is used, allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to the default value of cplex).
cplexIntegralityTolerance
If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).

cplexThreads
If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

Value
The function returns a list structured as follows:

lambda
The majority threshold.

weights
A vector containing the weights of the criteria. The elements are named according to the criteria IDs.

profilesPerformances
A matrix containing the lower profiles of the categories. The columns are named according to the criteria, whereas the rows are named according to the categories. The lower profile of the lower category can be considered as a dummy profile.

vetoPerformances
A matrix containing the veto profiles of the categories. The columns are named according to the criteria, whereas the rows are named according to the categories. The veto profile of the lower category can be considered as a dummy profile.

solverStatus
The solver status as given by glpk or cplex.

References

Examples

```r
performanceTable <- rbind(c(10,10,9), c(10,9,10), c(9,10,10), c(9,9,10),
                           c(9,10,9), c(10,9,10), c(10,10,7), c(10,7,10),
                           c(10,10,10), c(9,9,17), c(17,9,9),
                           c(7,10,17), c(10,17,7), c(17,7,10), c(7,17,10),
                           c(17,10,7), c(10,7,17), c(7,9,17), c(9,17,7),
                           c(17,7,9), c(7,17,9), c(17,9,7), c(9,7,17))
rownames(performanceTable) <- c("a1", "a2", "a3", "a4", "a5", "a6", "a7",
                                 "a8", "a9", "a10", "a11", "a12", "a13",
                                 "a14", "a15", "a16", "a17", "a18", "a19",
                                 "a20", "a21", "a22", "a23", "a24")
colnames(performanceTable) <- c("c1","c2","c3")
```

names(assignments) <- rownames(performanceTable)
categoriesRanks <- c(1,2)
names(categoriesRanks) <- c("P","F")
criteriaMinMax <- c("max","max","max")
names(criteriaMinMax) <- colnames(performanceTable)
x <- MRSortInferenceExact(performanceTable, assignments, categoriesRanks,
criteriaMinMax, veto = TRUE, readableWeights = TRUE,
readableProfiles = TRUE,
alternativesIDs = c("a1","a2","a3","a4","a5","a6","a7"))

ElectreAssignments <- MRSort(performanceTable, x$profilesPerformances,
categoriesRanks,
x$weights, criteriaMinMax, x$lambda,
criteriaVetos=x$vetoPerformances,
alternativesIDs = c("a1","a2","a3","a4","a5","a6","a7"))

---

**MRSortInterval**

**MRSort with imprecise evaluations**

**Description**

This method is an extension of the classical MRSort, that allows the handling of problems where the decision alternatives contain imprecise or even missing evaluations. Unlike MRSort, where an alternative is assigned to one category, MRSortInterval offers the possibility of assigning an alternative to one or more neighboring categories.

**Usage**

```r
MRSortInterval(performanceTable, categoriesLowerProfiles,
categoriesRanks, criteriaRanks, criteriaMinMax,
majorityThresholdPes, majorityThresholdOpt)
```

**Arguments**

- `performanceTable`
  Two-dimensionnal list 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). This list may contain imprecise performances of alternatives on the criteria, represented by interval evaluations, as well as missing performances.

- `categoriesLowerProfiles`
  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 except of the last one.
categoriesRanks
A vector containing the ranks of the categories (1 for the best, with higher values for increasingly less preferred categories). The vector needs to be named with the categories names, whereas the ranks need to be a range of values from 1 to the number of categories.

criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

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

majorityThresholdPes
The cut threshold for the pessimistic concordance relation.

majorityThresholdOpt
The cut threshold for the optimistic concordance relation.

Value
The function returns a list containing the assignments of the alternatives to all possible categories.

Examples

```
# the performance table

performanceTable <- as.list(numeric(6*5))
dim(performanceTable)=c(6,5)
performanceTable[[1,1]]<0
performanceTable[[1,2]]<0
performanceTable[[1,3]]<0
performanceTable[[1,4]]<0
performanceTable[[1,5]]<0
performanceTable[[2,1]]<0
performanceTable[[2,2]]<0
performanceTable[[2,3]]<-1
performanceTable[[2,4]]<0
performanceTable[[2,5]]<0
performanceTable[[3,1]]<0
performanceTable[[3,2]]<0
performanceTable[[3,3]]<-2
performanceTable[[3,4]]<0
performanceTable[[3,5]]<0
performanceTable[[4,1]]<0
performanceTable[[4,2]]<0
performanceTable[[4,3]]<-0:1
performanceTable[[4,4]]<0
performanceTable[[4,5]]<0
performanceTable[[5,1]]<0
performanceTable[[5,2]]<0
performanceTable[[5,3]]<NA
performanceTable[[5,4]]<0
performanceTable[[5,5]]<0
performanceTable[[6,1]]<0
```
normalizePerformanceTable

Function to normalize (or rescale) the columns (or criteria) of a performance table.

Description

Standardizes the range of the criteria according to a few methods: percentage of max, scale between 0 and 1, scale to 0 mean and 1 standard deviation, scale to euclidian unit length.
normalizePerformanceTable

Usage

normalizePerformanceTable(performaceTable, 
  normalizationTypes, 
  alternativesIDs = NULL, 
  criteriaIDs = NULL)

Arguments

performaceTable
A matrix containing the performance table to be plotted. The columns are la-
belled according to the criteria IDs, and the rows according to the alternatives
IDs.

normalizationTypes
Vector indicating the type of normalization that should be applied to each of the
criteria. Possible values: "percentageOfMax", "rescaling" (minimum becomes 
0, maximum becomes 1), "standardization" (rescale to a mean of 0 and a stan-
dard deviation of 1), "scaleToUnitLength" (scale the criteria values such that 
the column has euclidian length 1). Any other value (like "none") will result
in no data transformation. The elements are named according to the IDs of the
criteria.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be
filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

Examples

library(MCDA)

performaceTable <- matrix(runif(5*9), ncol=5)

row.names(performaceTable) <- c("x1","x2","x3","x4","x5","x6","x7","x8","x9")

colnames(performaceTable) <- c("g1","g2","g3","g4","g5")

normalizationTypes <- c("percentageOfMax","rescaling", 
  "standardization","scaleToUnitLength", "none")

names(normalizationTypes) <- c("g1","g2","g3","g4","g5")

normalizedPerformaceTable <- normalizePerformanceTable(performaceTable, 
  normalizationTypes)
pairwiseConsistencyMeasures

Consistency Measures for Pairwise Comparison Matrices

Description

This function calculates four pairwise consistency checks: Consistency Ratio (CR) from Saaty (1980), Koczkodaj’s Measure from Koczkodaj (1993) and Congruence / Dissonance Measures from Siraj et al. (2015).

Usage

pairwiseConsistencyMeasures(matrix)

Arguments

matrix A reciprocal matrix containing pairwise judgements

Value

The function returns a list of outputs for the four pairwise consistency checks

References


Examples

eexamplematrix <- t(matrix(c(1,0.25,4,1/6,4,1,4,0.25,0.25,0.25,1,0.2,6,4,5,1),nrow=4,ncol=4))
pairwiseConsistencyMeasures(examplematrix)

plotAlternativesValuesPreorder

Function to plot a preorder of alternatives, based on some score or ranking.

Description

Plots a preorder of alternatives as a graph, representing the ranking of the alternatives, w.r.t. some scores or ranks. A decreasing order or increasing order can be specified, w.r.t. to these scores or ranks.
Usage

plotAlternativesValuesPreorder(alternativesValues,
    decreasing = TRUE,
    alternativesIDs = NULL)

Arguments

alternativesValues
  A vector containing some values related to alternatives, as scores or ranks. The elements of the vector are named according to the IDs of the alternatives.

decreasing  A boolean to indicate if the alternatives are to be sorted increasingly (FALSE) or decreasingly (TRUE) w.r.t. the alternativesValues.

alternativesIDs  Vector containing IDs of alternatives, according to which the data should be filtered.

Examples

library(MCDA)

alternativesValues <- c(10,1,8,3,8,3,4,4,8,5)
names(alternativesValues) <- c("x10","x1","x9","x2","x8",
    "x3","x7","x4","x6","x5")

plotAlternativesValuesPreorder(alternativesValues,
    decreasing=TRUE,
    alternativesIDs=c("x10","x3","x7",
    "x4","x6","x5"))
Examples

```r
performanceTableMin <- t(matrix(c(78,87,79,19,8,68,74,8,90,89,74.5,9,20,81,30),
nrow=3,ncol=5, byrow=TRUE))
performanceTable <- t(matrix(c(80,87,86,19,8,70,74,10,90,89,75,9,33,82,30),
nrow=3,ncol=5, byrow=TRUE))
performanceTableMax <- t(matrix(c(81,87,95,19,8,72,74,15,90,89,75.5,9,36,84,30),
nrow=3,ncol=5, byrow=TRUE))

row.names(performanceTable) <- c("Yield","Toxicity","Cost","Separation","Odour")
colnames(performanceTable) <- c("Route One","Route Two","Route Three")
row.names(performanceTableMin) <- row.names(performanceTable)
colnames(performanceTableMin) <- colnames(performanceTable)
row.names(performanceTableMax) <- row.names(performanceTable)
colnames(performanceTableMax) <- colnames(performanceTable)

weights <- c(0.339,0.077,0.434,0.127,0.023)
names(weights) <- row.names(performanceTable)
criteriaMinMax <- c("max", "max", "max", "max", "max")
names(criteriaMinMax) <- row.names(performanceTable)

overall1 <- MARE(performanceTableMin, performanceTable, performanceTableMax,
                 weights, criteriaMinMax)
plotMARE(overall1)

overall2 <- MARE(performanceTableMin, performanceTable, performanceTableMax,
                 weights, criteriaMinMax,
                 alternativesIDs = c("Route Two","Route Three"),
                 criteriaIDs = c("Yield","Toxicity","Cost","Separation"))
plotMARE(overall2)
```

Description

The profiles shown are the separation profiles between the classes. They are stored as the lower profiles of the categories.

Usage

```r
plotMRSortSortingProblem(performanceTable, categoriesLowerProfiles,
categoriesRanks, assignments, criteriaMinMax, criteriaUBs, criteriaLBs,
```
plotMRSortSortingProblem

categoriesDictators = NULL, categoriesVetoes = NULL,
majorityRule = NULL, criteriaWeights = NULL,
majorityThreshold = NULL, alternativesIDs = NULL,
criteriaIDs = NULL, legendRatio = 0.2)

Arguments

performanceTable

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).

categoriesLowerProfiles

Matrix containing, in each row, the lower profiles of the categories (the separation
profiles in fact). 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.

categoriesRanks

A vector containing the ranks of the categories (1 for the best, with higher values
for increasingly less preferred categories). The vector needs to be named with
the categories names, whereas the ranks need to be a range of values from 1 to
the number of categories.

assignments

Vector containing the assignments (IDs of the categories) of the alternatives to
the categories. The elements are named according to the alternatives.

criteriaMinMax

Vector containing the preference direction on each of the criteria. "min" (resp.
"max") indicates that the criterion has to be minimized (maximized). The ele-
ments are named according to the IDs of the criteria.

criteriaLBs

Vector containing the lower bounds of the criteria to be considered for the plot-
ting. The elements are named according to the IDs of the criteria.

criteriaUBs

Vector containing the upper bounds of the criteria to be considered for the plot-
ting. The elements are named according to the IDs of the criteria.

categoriesDictators

Matrix containing, in each row, the lower dictator 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.

categoriesVetoes

Matrix containing, in each row, the lower veto 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.

majorityRule

A string containing one of the following values: 'V', 'D', 'v', 'd', 'dV', 'Dv',
dv'. This indicates the type of majority rule that will be used by the MRSort
model. 'V' stands for MRSort with vetoes, 'D' stands for MRSort with dicta-
tors, 'v' stands for MRSort with vetoes weakened by dictators, 'd' stands for
MRSort with dictators weakened by vetoes, 'dV' stands for MRSort with vetoes
dominating dictators, 'Dv' stands for MRSort with dictators dominating vetoes,
while 'dv' stands for MRSort with conflicting vetoes and dictators.
criteriaWeights
Vector containing the criteria weights. The elements are named according to the IDs of the criteria.

majorityThreshold
A value corresponding to the majority threshold. Along with the criteria weights, this value is used to determine when a coalition of criteria is sufficient in order to assert that an alternative is at least as good as a category profile.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

legendRatio
The ratio between the legend and plot heights. By default 0.2.

Examples

# the performance table

performanceTable <- rbind(
  c(1,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")

# lower profiles of the categories
# (best category in the first position of the list)

categoriesLowerProfiles <- rbind(c(3, 11, 3),c(7, 25, 2),c(30,30,0))
colnames(categoriesLowerProfiles) <- colnames(performanceTable)
rownames(categoriesLowerProfiles)<-c("Good","Medium","Bad")
categoriesRanks <-c(1,2,3)
names(categoriesRanks) <- c("Good","Medium","Bad")

# criteria to minimize or maximize

criteriaMinMax <- c("min","min","max")

names(criteriaMinMax) <- colnames(performanceTable)

# lower bounds of the criteria for the determination of value functions
criteriaLBs=cbind(0,5,0)
plotPiecewiseLinearValueFunctions

Function to plot piecewise linear value functions.

Description

Plots piecewise linear value function.

Usage

plotPiecewiseLinearValueFunctions(valueFunctions,
                                   criteriaIDs = NULL)

Arguments

valueFunctions A list containing, for each criterion, the piecewise linear value functions defined by the coordinates of the break points. Each value function is defined by a matrix of breakpoints, where the first row corresponds to the abscissa (row labelled "x") and where the second row corresponds to the ordinate (row labelled "y").

criteriaIDs Vector containing IDs of criteria, according to which the data should be filtered.
plotRadarPerformanceTable

Function to plot radar plots of alternatives of a performance table.

Description

Plots radar plots of alternatives contained in a performance table, either in one radar plot, or on multiple radar plots. For a given alternative, the plot shows how far above/below average (the thick black line) each of the criteria performances values are (average taken w.r.t. to the filtered performance table).

Usage

plotRadarPerformanceTable(performanceTable,
                          criteriaMinMax=NULL,
                          alternativesIDs = NULL,
                          criteriaIDs = NULL,
                          overlay=FALSE,
                          bw=FALSE,
                          lwd=2)

Arguments

performanceTable
   A matrix containing the performance table to be plotted. The columns are labelled according to the criteria IDs, and the rows according to the alternatives IDs.

criteriaMinMax
   Vector indicating whether criteria should be minimized or maximized. If it is given, a "higher" value in the radar plot corresponds to a more preferred value according to the decision maker. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

Examples

v<-list(
        Price = array(c(30, 0, 16, 0, 2, 0.0875),
                      dim=c(2,3), dimnames = list(c("x", "y"), NULL)),
        Time = array(c(40, 0, 30, 0, 20, 0.025, 10, 0.9),
                     dim = c(2, 4), dimnames = list(c("x", "y"), NULL)),
        Comfort = array(c(0, 0, 1, 0, 2, 0.0125, 3, 0.0125),
                        dim = c(2, 4), dimnames = list(c("x", "y"), NULL)))

# plot the value functions
plotPiecewiseLinearValueFunctions(v)
alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.
criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.
overlay
Boolean value indicating if the plots should be overlayed on one plot (TRUE), or not (FALSE)
bw
Boolean value indicating if the plots should be in black/white (TRUE) or color (FALSE)
lwd
Value indicating the line width of the plot.

Examples

library(MCDA)

performanceTable <- matrix(runif(6*9), ncol=6)

row.names(performanceTable) <- c("x1","x2","x3","x4","x5","x6","x7","x8","x9")

colnames(performanceTable) <- c("g1","g2","g3","g4","g5","g6")

criteriaMinMax <- c("min","max","min","max","min","max")

names(criteriaMinMax) <- c("g1","g2","g3","g4","g5","g6")

# plotRadarPerformanceTable(performanceTable, criteriaMinMax, overlay=TRUE)

plotRadarPerformanceTable(performanceTable, criteriaMinMax, alternativesIDs = c("x1","x2","x3","x4"), criteriaIDs = c("g1","g3","g4","g5","g6"), overlay=FALSE, bw=FALSE)

# plotRadarPerformanceTable(performanceTable, criteriaMinMax, alternativesIDs = c("x1","x2"), criteriaIDs = c("g1","g3","g4","g5","g6"), overlay=FALSE)

PROMETHEEI

Description

The PROMETHEE I constructs preference indices from the criteria evaluations of alternatives and outputs three preference relations (P - preference, I - indifference, R - incomparability) based on the outranking flows between the alternatives.
Usage

PROMETHEEI(performanceTable, preferenceFunction, preferenceThreshold, indifferenceThreshold, gaussParameter, criteriaWeights, criteriaMinMax)

Arguments

performanceTable
Matrix containing the evaluation 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).

preferenceFunction
A vector with preference functions. preferenceFunction should be equal to Usual, U-shape, V-shape, Level, V-shape-Indiff or Gaussian. The elements are named according to the IDs of the criteria.

preferenceThreshold
A vector containing threshold of strict preference. The elements are named according to the IDs of the criteria.

indifferenceThreshold
A vector containing threshold of indifference. The elements are named according to the IDs of the criteria.

gaussParameter
A vector containing parameter of the Gaussian preference function. The elements are named according to the IDs of the criteria.

criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

Value

The function returns three matrices: The first one contains the preference relations between the alternatives, the second one contains the indifference relations between the alternatives and the third one contains the incomparability relations between the alternatives.

Examples

# The evaluation table

performanceTable <- rbind(
  c(1,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")
The preference functions

```r
preferenceFunction <- c("Gaussian", "Level", "V-shape-Indiff")
```

# Preference threshold
```
preferenceThreshold <- c(5, 15, 3)
```
```
names(preferenceThreshold) <- colnames(performanceTable)
```

# Indifference threshold
```
indifferenceThreshold <- c(3, 11, 1)
```
```
names(indifferenceThreshold) <- colnames(performanceTable)
```

# Parameter of the Gaussian preference function
```
 gaussParameter <- c(4, 0, 0)
```
```
names(gaussParameter) <- colnames(performanceTable)
```

# Weights
```
criteriaWeights <- c(0.2, 0.3, 0.5)
```
```
names(criteriaWeights) <- colnames(performanceTable)
```

# Criteria to minimize or maximize
```
criteriaMinMax <- c("min", "min", "max")
```
```
names(criteriaMinMax) <- colnames(performanceTable)
```

PROMETHEEII(performanceTable, preferenceFunction, preferenceThreshold, indifferenceThreshold, gaussParameter, criteriaWeights, criteriaMinMax)
preferenceFunction
A vector with preference functions. preferenceFunction should be equal to Usual, U-shape, V-shape, Level, V-shape-Indiff or Gaussian. The elements are named according to the IDs of the criteria.

preferenceThreshold
A vector containing threshold of strict preference. The elements are named according to the IDs of the criteria.

indifferenceThreshold
A vector containing threshold of indifference. The elements are named according to the IDs of the criteria.

gaussParameter
A vector containing parameter of the Gaussian preference function. The elements are named according to the IDs of the criteria.

criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

Value
The function returns a list containing the alternatives IDs in decreasing order of preference. Each elements of the list can be a vector of alternatives IDs.

Examples

# The evaluation table

```r
performanceTable <- rbind(
  c(1,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")
```

# The preference functions

```r
preferenceFunction<-c("Gaussian","Level","V-shape-Indiff")
```

# Preference threshold

```r
preferenceThreshold<-c(5,15,3)
names(preferenceThreshold)<-colnames(performanceTable)
```

# Indifference threshold

```r
indifferenceThreshold<-c(3,11,1)
names(indifferenceThreshold)<-colnames(performanceTable)
```

# Parameter of the Gaussian preference function

gaussParameter<-c(4,0,0)
names(gaussParameter) <- colnames(performanceTable)

# weights
criteriaWeights <- c(0.2, 0.3, 0.5)
names(criteriaWeights) <- colnames(performanceTable)

# criteria to minimize or maximize
criteriaMinMax <- c("min", "min", "max")
names(criteriaMinMax) <- colnames(performanceTable)

PROMETHEEII(performanceTable, preferenceFunction, preferenceThreshold,
             indifferenceThreshold, gaussParameter, criteriaWeights,
             criteriaMinMax)

---

**PROMETHEEOutrankingFlows**

*Outranking flows for the PROMETHEE methods*

**Description**

This function computes the positive and negative outranking flows for the PROMETHEE methods. It takes as input a performance table and converts the evaluations to preference indices based on the given function types and parameters for each criterion.

**Usage**

PROMETHEEOutrankingFlows(performanceTable, preferenceFunction, preferenceThreshold, indifferenceThreshold, gaussParameter, criteriaWeights, criteriaMinMax)

**Arguments**

- **performanceTable**
  Matrix containing the evaluation 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).

- **preferenceFunction**
  A vector with preference functions. preferenceFunction should be equal to Usual, U-shape, V-shape, Level, V-shape-Indiff or Gaussian. The elements are named according to the IDs of the criteria.

- **preferenceThreshold**
  A vector containing threshold of strict preference. The elements are named according to the IDs of the criteria.

- **indifferenceThreshold**
  A vector containing threshold of indifference. The elements are named according to the IDs of the criteria.
PROMETHEEOutrankingFlows

gaussParameter A vector containing parameter of the Gaussian preference function. The elements are named according to the IDs of the criteria.

criteriaWeights Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

criteriaMinMax Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

Value

The function returns two vectors: The first one contains the positive outranking flows and the second one contains the negative outranking flows.

Examples

# The evaluation table

```r
performanceTable <- rbind(
  c(1,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")
```

# The preference functions

```r
preferenceFunction<-c("Gaussian","Level","V-shape-Indiff")
```

#Preference threshold

```r
preferenceThreshold<-c(5,15,3)
names(preferenceThreshold)<-colnames(performanceTable)
```

#Indifference threshold

```r
indifferenceThreshold<-c(3,11,1)
names(indifferenceThreshold)<-colnames(performanceTable)
```

#Parameter of the Gaussian preference function

```r
gaussParameter<-c(4,0,0)
names(gaussParameter)<-colnames(performanceTable)
```

#weights

```r
criteriaWeights<-c(0.2,0.3,0.5)
names(criteriaWeights)<-colnames(performanceTable)
```

# criteria to minimize or maximize

```r
criteriaMinMax<-c("min","min","max")
names(criteriaMinMax)<-colnames(performanceTable)
```
# Outranking flows

outrankingFlows<-PROMETHEEOutrankingFlows(performanceTable, preferenceFunction, 
  preferenceThreshold, indifferenceThreshold, 
  gaussParameter, criteriaWeights, 
  criteriaMinMax)

## Description

This function computes the preference indices from a performance table based on the given function types and parameters for each criterion.

## Usage

PROMETHEEPreferenceIndices(performanceTable, preferenceFunction, 
  preferenceThreshold, indifferenceThreshold, 
  gaussParameter, criteriaWeights, criteriaMinMax)

## Arguments

- **performanceTable**: Matrix 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).
- **preferenceFunction**: A vector containing the names of the preference functions to be used. preferenceFunction should be equal to Usual, U-shape, V-shape, Level, V-shape-Indiff or Gaussian. The elements of the vector are named according to the IDs of the criteria.
- **preferenceThreshold**: A vector containing thresholds of strict preference. The elements are named according to the IDs of the criteria.
- **indifferenceThreshold**: A vector containing thresholds of indifference. The elements are named according to the IDs of the criteria.
- **gaussParameter**: A vector containing parameters of the Gaussian preference function. The elements are named according to the IDs of the criteria.
- **criteriaWeights**: Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.
PROMETHEEPreferenceIndices

criteriaMinMax  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

Value

The function returns a matrix containing all the aggregated preference indices.

Examples

# The evaluation table

```
performanceTable <- rbind(
  c(1,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")
```

# The preference functions

```
preferenceFunction<-c("Gaussian","Level","V-shape-Indiff")
```

#Preference threshold

```
preferenceThreshold<-c(5,15,3)
names(preferenceThreshold)<-colnames(performanceTable)
```

#Indifference threshold

```
indifferenceThreshold<-c(3,11,1)
names(indifferenceThreshold)<-colnames(performanceTable)
```

#Parameter of the Gaussian preference function

```
gaussParameter<-c(4,0,0)
names(gaussParameter)<-colnames(performanceTable)
```

#weights

```
criteriaWeights<-c(0.2,0.3,0.5)
names(criteriaWeights)<-colnames(performanceTable)
```

# criteria to minimize or maximize

```
criteriaMinMax<-c("min","min","max")
names(criteriaMinMax)<-colnames(performanceTable)
```

#Preference indices

```
preferenceTable<-PROMETHEEPreferenceIndices(performanceTable, preferenceFunction, preferenceThreshold, indifferenceThreshold, gaussParameter, criteriaWeights,
```
SRMP

SRMP: a simple ranking method using reference profiles

Description

SRMP is a ranking method that uses dominating reference profiles, in a given lexicographic ordering, in order to output a total preorder of a set of alternatives.

Usage

SRMP(performanceTable, referenceProfiles, lexicographicOrder, criteriaWeights, criteriaMinMax, alternativesIDs = NULL, criteriaIDs = NULL)

Arguments

performanceTable
  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).

referenceProfiles
  Matrix containing, in each row, the reference profiles. The columns are named according to the criteria.

lexicographicOrder
  A vector containing the indexes of the reference profiles in a given order. This vector needs to be of the same length as the number of rows in referenceProfiles and it has to contain a permutation of the indices of these rows.

criteriaWeights
  Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

criteriaMinMax
  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

alternativesIDs
  Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
  Vector containing IDs of criteria, according to which the data should be filtered.

Value

The function returns a vector containing the ranks of the alternatives (the higher the better).

References

Examples

```r
# the performance table
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),
c(10,10,7),c(10,7,10),c(7,10,10),c(9,9,17),c(9,17,9),
c(7,17,7),c(17,7,7),c(17,17,10),c(7,17,10),c(10,7,17),
c(7,9,17),c(9,17,9),c(17,7,9),c(17,9,7),c(9,7,17))

referenceProfiles <- rbind(c(5,5,5),c(10,10,10),c(15,15,15))

lexicographicOrder <- c(2,1,3)
weights <- c(0.2,0.44,0.36)
criteriaMinMax <- c("max","max","max")

rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11","a12",
"a13","a14","a15","a16","a17","a18","a19","a20","a21","a22",
"a23","a24")

colnames(performanceTable) <- c("c1","c2","c3")
colnames(referenceProfiles) <- c("c1","c2","c3")

names(weights) <- c("c1","c2","c3")
names(criteriaMinMax) <- colnames(performanceTable)

expectedpreorder <- list("a16","a13","a14","a17","a18","a19","a20",
"a10","a11","a22",
"a4","a2","a8","a20","a22",
"a5","a10","a19","a24",
"a4","a12","a21","a23","a6")

preorder<--SRMP(performanceTable, referenceProfiles, lexicographicOrder, weights, criteriaMinMax)
```

---

**SRMPInference**

*Exact inference of an SRMP model given a maximum number of reference profiles*

**Description**

Exact inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method outputs an SRMP model that is as consistent as possible with the provided pairwise comparisons (i.e. the model - the number of profiles and their lexicographic order - that maximizes the number of fulfilled pairwise comparisons). The method will search for a model with the minimum possible number of profiles up to a given maximum value.

**Usage**

```r
SRMPInference(performanceTable, criteriaMinMax, maxProfilesNumber, preferencePairs, weights, criteriaMinMax)
```
indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, solver="glpk", timeLimit = NULL, cplexIntegralityTolerance = NULL, cplexThreads = NULL)

Arguments

performanceTable
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).

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

maxProfilesNumber
A strictly positive numerical value which gives the highest number of reference profiles the sought SRMP model should have.

preferencePairs
A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

indifferencePairs
A two column matrix containing on each row a pair of alternative names the two alternatives are considered to be indifferent with respect to each other.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

solver
String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.

timeLimit
Allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to no time limit).

cplexIntegralityTolerance
If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).

cplexThreads
If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

Value

The function returns a list containing:

criteriaWeights
The inferred criteria weights.

referenceProfilesNumber
The inferred reference profiles number.

referenceProfiles
The inferred reference profiles.
lexicographicOrder
The inferred lexicographic order of the profiles.

fitness
The percentage (0 to 1) of fulfilled pair-wise relations.

solverStatus
The solver status as given by glpk or cplex.

humanReadableStatus
A description of the solver status.

References


Examples

```r
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),c(10,9,9),
c(10,10,7),c(10,7,10),c(7,10,10),c(9,9,17),c(9,17,9),c(17,9,9),
c(7,10,17),c(10,17,7),c(17,7,10),c(17,10,7),c(10,7,17),
c(7,9,17),c(9,17,7),c(17,7,9),c(17,9,7),c(9,7,17))
criteriaMinMax <- c("max","max","max")
rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11","a12",
"a13","a14","a15","a16","a17","a18","a19","a20","a21","a22",
"a23","a24")
colnames(performanceTable) <- c("c1","c2","c3")
names(criteriaMinMax) <- colnames(performanceTable)
preferencePairs <- matrix(c("a16","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5",
"a10","a4","a12","a13","a2","a14","a17","a18","a15","a2",
"a11","a5","a10","a4","a12","a6"),14,2)
indifferencePairs <- matrix(c("a3","a1","a2","a11","a11","a20","a10","a10","a19","a12","a12",
"a21","a9","a7","a8","a20","a22","a19","a24","a24","a21",
"a23","a23"),12,2)
result<-SRMPInference(performanceTable, criteriaMinMax, 3, preferencePairs, indifferencePairs,
alternativesIDs = c("a1","a3","a7","a9","a13","a14","a15","a16","a17","a18"))
```

SRMPInferenceApprox
Approximative inference of an SRMP model

Description

Approximative inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method outputs an SRMP model that fulfills as many pairwise comparisons as possible. Neither the number of reference profiles, nor the lexicographic order are fixed beforehand, however a maximum value for the number of reference profiles needs to be provided.
Usage

SRMPInferenceApprox(performanceTable, criteriaMinMax, maxProfilesNumber, preferencePairs,
    indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL,
    timeLimit = 60, populationSize = 20, mutationProb = 0.1)

Arguments

performanceTable
    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).

criteriaMinMax
    Vector containing the preference direction on each of the criteria. "min" (resp.
    "max") indicates that the criterion has to be minimized (maximized). The ele-
    ments are named according to the IDs of the criteria.

maxProfilesNumber
    The maximum number of reference profiles of the SRMP model.

preferencePairs
    A two column matrix containing on each row a pair of alternative names where
the first alternative is considered to be strictly preferred to the second.

indifferencePairs
    A two column matrix containing on each row a pair of alternative names the two
alternatives are considered to indifferent with respect to each other.

alternativesIDs
    Vector containing IDs of alternatives, according to which the datashould be fil-
tered.

criteriaIDs
    Vector containing IDs of criteria, according to which the data should be filtered.

timeLimit
    Allows to fix a time limit of the execution, in seconds (default 60).

populationSize
    Allows to change the size of the population used by the genetic algorithm (de-
    fault 20).

mutationProb
    Allows to change the mutation probability used by the genetic algorithm (default
    0.1).

Value

The function returns a list containing:

criteriaWeights
    The inferred criteria weights.

referenceProfilesNumber
    The number of inferred reference profiles.

referenceProfiles
    The inferred reference profiles.

lexicographicOrder
    The inferred lexicographic order of the reference profiles.

fitness
    The percentage of fulfilled pair-wise relations.
References


Examples

```r
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),c(10,9,9),
c(10,10,7),c(10,7,10),c(7,10,10),c(9,9,17),c(9,17,9),c(17,9,9),
c(7,10,17),c(10,17,7),c(17,7,10),c(7,17,10),c(17,10,7),c(10,7,17),
c(7,9,17),c(9,17,7),c(17,7,9),c(7,17,9),c(17,9,7),c(9,7,17))
criteriaMinMax <- c("max","max","max")
rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11","a12","a13","a14","a15","a16","a17","a18","a19","a20","a21","a22","a23","a24")
colnames(performanceTable) <- c("c1","c2","c3")
names(criteriaMinMax) <- colnames(performanceTable)
# expected result for the tests below
expectedpreorder <- list("a16","a13",c("a3","a9"),"a14","a17",c("a1","a7"),"a18","a15")
# test - preferences and indifferences
preferencePairs <- matrix(c("a16","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2")
indifferencePairs <- matrix(c("a3","a1","a2","a11","a11","a20","a10","a10","a19","a12","a12","a21","a9","a7","a8","a20","a22","a22","a19","a24","a24","a24","a23","a23"),12,2)
set.seed(1)
result<-SRMPInferenceApprox(performanceTable, criteriaMinMax, 3, preferencePairs,
indifferencePairs, alternativesIDs = c("a1","a3","a7","a9","a13","a14","a15","a16","a18"))
```

SRMPInferenceApproxFixedLexicographicOrder

Approximative inference of an SRMP model given the lexicographic order of the profiles
SRMPInferenceApproxFixedLexicographicOrder

Description

Approximative inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method outputs an SRMP model that fulfills as many pairwise comparisons as possible. The number of reference profiles and their lexicographic order is fixed beforehand.

Usage

SRMPInferenceApproxFixedLexicographicOrder(performanceTable, criteriaMinMax, lexicographicOrder, preferencePairs, indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, timeLimit = 60, populationSize = 20, mutationProb = 0.1)

Arguments

performanceTable
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).

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

lexicographicOrder
A vector containing the indexes of the reference profiles in a given order. The number of reference profiles to be used is derived implicitly from the size of this vector. The elements of this vector need to be a permutation of the indices from 1 to its size.

preferencePairs
A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

indifferencePairs
A two column matrix containing on each row a pair of alternative names the two alternatives are considered to indifferent with respect to each other.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

timeLimit
Allows to fix a time limit of the execution, in seconds (default 60).

populationSize
Allows to change the size of the population used by the genetic algorithm (default 20).

mutationProb
Allows to change the mutation probability used by the genetic algorithm (default 0.1).
Value
The function returns a list containing:

- **criteriaWeights**: The inferred criteria weights.
- **referenceProfiles**: The inferred reference profiles.
- **lexicographicOrder**: The lexicographic order of the reference profiles, in this case the one that was originally given as input.
- **fitness**: The percentage of fulfilled pair-wise relations.

References

Examples

```r
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),c(10,9,9),
c(10,10,7),c(10,7,10),c(7,10,10),c(9,9,17),c(9,17,9),c(17,9,9),
c(7,10,17),c(10,17,7),c(17,7,10),c(7,17,10),c(17,10,7),c(10,7,17),
c(7,9,17),c(9,17,7),c(17,7,9),c(7,17,9),c(17,9,7),c(9,7,17))

lexicographicOrder <- c(1,2,3)
criteriaMinMax <- c("max","max","max")
rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11",
    "a12","a13","a14","a15","a16","a17","a18","a19","a20",
    "a21","a22","a23","a24")
colnames(performanceTable) <- c("c1","c2","c3")
names(criteriaMinMax) <- colnames(performanceTable)

# expected result for the tests below
expectedpreorder <- list("a16","a13",c("a3","a9"),"a14","a17",c("a1","a7"),"a18","a15")

# test - preferences and indifferences
preferencePairs <- matrix(c("a16","a13","a3","a14","a17","a1","a18","a15","a2","a11",
    "a5","a10","a4","a12","a13","a3","a14","a17","a1","a18",
    "a15","a2","a11","a5","a10","a4","a12","a16"),14,2)
indifferencePairs <- matrix(c("a3","a1","a2","a11","a11","a20","a10","a10","a19","a12",
    "a12","a21","a9","a7","a8","a20","a22","a22","a19","a24",
    "a24","a21","a23","a23"),12,2)

set.seed(1)
```
SRMPInferenceApproxFixedProfilesNumber

Approximative inference of an SRMP model given the number of reference profiles

Description

Approximative inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method outputs an SRMP model that fulfills as many pairwise comparisons as possible. The number of reference profiles is fixed beforehand, however the algorithm will explore any lexicographic order between them.

Usage

SRMPInferenceApproxFixedProfilesNumber(performanceTable, criteriaMinMax, profilesNumber, preferencePairs, indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, timeLimit = 60, populationSize = 20, mutationProb = 0.1)

Arguments

- **performanceTable**: 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).

- **criteriaMinMax**: Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **profilesNumber**: The number of reference profiles of the SRMP model.

- **preferencePairs**: A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

- **indifferencePairs**: A two column matrix containing on each row a pair of alternative names the two alternatives are considered to indifferent with respect to each other.

- **alternativesIDs**: Vector containing IDs of alternatives, according to which the data should be filtered.
criteriaIDs Vector containing IDs of criteria, according to which the data should be filtered.
timeLimit Allows to fix a time limit of the execution, in seconds (default 60).
populationSize Allows to change the size of the population used by the genetic algorithm (default 20).
mutationProb Allows to change the mutation probability used by the genetic algorithm (default 0.1).

Value

The function returns a list containing:
criteriaWeights The inferred criteria weights.
referenceProfiles The inferred reference profiles.
lexicographicOrder The inferred lexicographic order of the reference profiles.
fitness The percentage of fulfilled pair-wise relations.

References


Examples

# the performance table
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,9,10),c(9,10,9),c(10,9,9),
  c(10,10,7),c(10,7,10),c(7,10,10),c(9,9,17),c(9,17,9),c(17,9,9),
  c(7,10,7),c(10,10,7),c(17,7,10),c(17,10,7),c(17,10,7),c(17,7,17),
  c(7,9,17),c(9,17,7),c(17,7,9),c(7,17,9),c(17,9,7),c(9,7,17))
criteriaMinMax <- c("max","max","max")
rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11",
  "a12","a13","a14","a15","a16","a17","a18","a19","a20",
  "a21","a22","a23","a24")
colnames(performanceTable) <- c("c1","c2","c3")
names(criteriaMinMax) <- colnames(performanceTable)

# expected result for the tests below
expectedpreorder <- list("a16","a13",c("a3","a9"),"a14",c("a1","a7"),"a15")

# test - preferences and indifferences
SRMPInferenceFixedLexicographicOrder <- matrix(c("a16","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a14","a17","a1","a18","a15","a2","a11","a5","a10","a4","a12","a13","a14","a17","a1","a18"),14,2)

indifferencePairs <- matrix(c("a3","a1","a2","a11","a11","a20","a10","a10","a19","a12","a12","a21","a9","a7","a8","a20","a22","a22","a19","a24","a24","a21","a23","a23"),12,2)

set.seed(1)

result<-SRMPInferenceApproxFixedProfilesNumber(performanceTable, criteriaMinMax, 3, preferencePairs, indifferencePairs, alternativesIDs = c("a1","a3","a7","a9","a13","a14","a15","a16"))

---

**SRMPInferenceFixedLexicographicOrder**

*Exact inference of an SRMP model given the lexicographic order of the profiles*

---

**Description**

Exact inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method outputs an SRMP model that maximizes the number of fulfilled pairwise comparisons. The number of reference profiles and their lexicographic order is fixed.

**Usage**

```r
SRMPInferenceFixedLexicographicOrder(performanceTable, criteriaMinMax, lexicographicOrder, preferencePairs, indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, solver="glpk", timeLimit = NULL, cplexIntegralityTolerance = NULL, cplexThreads = NULL)
```

**Arguments**

- **performanceTable**
  - 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).

- **criteriaMinMax**
  - Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **lexicographicOrder**
  - A vector containing the indexes of the reference profiles in a given order. The number of reference profiles to be used is derived implicitly from the size of this vector. The elements of this vector need to be a permutation of the indices from 1 to its size.
preferencePairs
A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

indifferencePairs
A two column matrix containing on each row a pair of alternative names the two alternatives are considered to indifferent with respect to each other.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

solver
String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.

timeLimit
Allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to no time limit).

cplexIntegralityTolerance
If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).

cplexThreads
If the cplex solver is used, allows to set the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex - 1).

Value
The function returns a list containing:

criteriaWeights
The inferred criteria weights.

referenceProfiles
The inferred reference profiles.

fitness
The percentage (0 to 1) of fulfilled pair-wise relations.

solverStatus
The solver status as given by glpk or cplex.

humanReadableStatus
A description of the solver status.

References

Examples

# the performance table

performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,9,10),c(9,10,9),c(10,9,9),
c(10,10,7),c(10,7,10),c(7,10,10),c(9,9,17),c(9,17,9),c(17,9,9),
c(7,10,17),c(10,17,7),c(17,7,10),c(7,17,10),c(17,10,7),c(10,7,17),
SRMPInferenceFixedProfilesNumber

Exact inference of an SRMP model given the number of reference profiles

Description

Exact inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method outputs an SRMP model that is as consistent as possible with the provided pairwise comparisons (i.e. the model - and the lexicographic order of the reference profiles - that maximizes the number of fulfilled pairwise comparisons). The number of reference profiles is fixed and needs to be provided.

Usage

SRMPInferenceFixedProfilesNumber(performanceTable, criteriaMinMax, profilesNumber, preferencePairs, indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, solver="glpk", timeLimit = NULL, cplexIntegralityTolerance = NULL, cplexThreads = NULL)
**Arguments**

- **performanceTable**
  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).

- **criteriaMinMax**
  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **profilesNumber**
  A strictly positive numerical value which gives the number of reference profiles in the sought SRMP model.

- **preferencePairs**
  A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

- **indifferencePairs**
  A two column matrix containing on each row a pair of alternative names the two alternatives are considered to indifferent with respect to each other.

- **alternativesIDs**
  Vector containing IDs of alternatives, according to which the data should be filtered.

- **criteriaIDs**
  Vector containing IDs of criteria, according to which the data should be filtered.

- **solver**
  String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.

- **timeLimit**
  Allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to no time limit).

- **cplexIntegralityTolerance**
  If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).

- **cplexThreads**
  If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

**Value**

The function returns a list containing:

- **criteriaWeights**
  The inferred criteria weights.

- **referenceProfiles**
  The inferred reference profiles.

- **lexicographicOrder**
  The inferred lexicographic order of the profiles.

- **fitness**
  The percentage (0 to 1) of fulfilled pair-wise relations.

- **solverStatus**
  The solver status as given by glpk or cplex.

- **humanReadableStatus**
  A description of the solver status.
SRMPInferenceNoInconsist

References


Examples

```r
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),c(10,9,9),
c(10,10,7),c(10,7,10),c(7,10,10),c(9,17,9),c(17,9,9),c(17,10,7),
c(7,10,7),c(10,7,10),c(7,10,10),c(7,17,9),c(17,9,7),c(17,7,9),
c(10,7,7),c(10,7,10),c(7,17,9),c(17,7,9),c(7,17,7),c(9,7,7))
criteriaMinMax <- c("max","max","max")
rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11","a12",
"a13","a14","a15","a16","a17","a18","a19","a20","a21","a22",
"a23","a24")
colnames(performanceTable) <- c("c1","c2","c3")
names(criteriaMinMax) <- colnames(performanceTable)

preferencePairs <- matrix(c("a16","a13","a4","a5","a6","a7","a8","a9","a10","a11","a12",
"a13","a14","a15","a16","a17","a18","a19","a20","a21","a22",
"a23","a24"),14,2)
indifferencePairs <- matrix(c("a3","a1","a2","a11","a11","a20","a10","a10","a19","a12","a12",
"a21","a9","a7","a8","a20","a22","a19","a24","a24","a21",
"a23","a23"),12,2)

result<-SRMPInferenceFixedProfilesNumber(performanceTable,criteriaMinMax, 3, preferencePairs,
indifferencePairs, alternativesIDs = c("a1","a3",
"a7","a9","a13","a14","a15","a16","a17","a18"))
```

---

SRMPInferenceNoInconsist

*Exact inference of an SRMP model given a maximum number of reference profiles - no inconsistencies*

Description

Exact inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method only outputs a result when an SRMP model consistent with the provided pairwise comparisons exists. The method will search for a model with the minimum possible number of profiles up to a given maximum value. If such a model exists, this method is significantly faster than the one which handles inconsistencies.
Usage

SRMPInferenceNoInconsist(performanceTable, criteriaMinMax, maxProfilesNumber, preferencePairs, indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, solver="glpk", timeLimit = NULL, cplexIntegerTolerance = NULL, cplexThreads = NULL)

Arguments

performanceTable
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).

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

maxProfilesNumber
A strictly positive numerical value which gives the highest number of reference profiles the sought SRMP model should have.

preferencePairs
A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

indifferencePairs
A two column matrix containing on each row a pair of alternative names the two alternatives are considered to indifferent with respect to each other.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs
Vector containing IDs of criteria, according to which the data should be filtered.

solver
String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.

timeLimit
Allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to no time limit).

cplexIntegerTolerance
If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).

cplexThreads
If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

Value

The function returns a list containing:

criteriaWeights
The inferred criteria weights.
referenceProfilesNumber
   The inferred reference profiles number.

referenceProfiles
   The inferred reference profiles.

lexicographicOrder
   The inferred lexicographic order of the profiles.

solverStatus
   The solver status as given by glpk or cplex.

humanReadableStatus
   A description of the solver status.

References


Examples

```r
performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),c(10,9,9),
                         c(10,10,7),c(10,7,10),c(7,10,10),c(7,9,10),c(9,9,17),c(9,17,9),
                         c(7,10,17),c(10,17,7),c(17,7,10),c(17,10,7),c(10,17,7),
                         c(9,9,17),c(9,17,7),c(17,7,9),c(17,9,7),c(9,7,17))
criteriaMinMax <- c("max","max","max")
rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11","a12",
   "a13","a14","a15","a16","a17","a18","a19","a20","a21","a22",
   "a23","a24")
colnames(performanceTable) <- c("c1","c2","c3")
names(criteriaMinMax) <- colnames(performanceTable)

preferencePairs <- matrix(c("a16","a13","a3","a14","a17","a1","a18","a15","a2","a11","a5",
   "a10","a4","a12","a13","a3","a14","a17","a1","a18","a15","a2",
   "a11","a5","a10","a4","a12","a6"),14,2)
indifferencePairs <- matrix(c("a3","a1","a2","a11","a11","a20","a10","a10","a19","a12","a12",
   "a21","a9","a7","a8","a20","a22","a22","a19","a24","a24","a21",
   "a23","a23"),12,2)

result<-SRMPInferenceNoInconsist(performanceTable, criteriaMinMax, 3, preferencePairs,
   indifferencePairs, alternativesIDs = c("a1","a2","a3","a4",
   "a5","a6","a7","a8","a10","a11","a12","a14","a16","a17","a18",
   "a19","a20","a21","a23","a24"))
```
SRMPInferenceNoInconsistFixedLexicographicOrder

Exact inference of an SRMP model given the lexicographic order of the profiles - no inconsistencies

Description

Exact inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method only outputs a result when an SRMP model consistent with the provided pairwise comparisons exists. The number of reference profiles and their lexicographic order is fixed. If such a model exists, this method is significantly faster than the one which handles inconsistencies.

Usage

SRMPInferenceNoInconsistFixedLexicographicOrder(performanceTable, criteriaMinMax, lexicographicOrder, preferencePairs, indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, solver="glpk", timeLimit = NULL, cplexIntegralityTolerance = NULL, cplexThreads = NULL)

Arguments

performanceTable
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).

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

lexicographicOrder
A vector containing the indexes of the reference profiles in a given order. The number of reference profiles to be used is derived implicitly from the size of this vector. The elements of this vector need to be a permutation of the indices from 1 to its size.

preferencePairs
A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

indifferencePairs
A two column matrix containing on each row a pair of alternative names the two alternatives are considered to indifferent with respect to each other.

alternativesIDs
Vector containing IDs of alternatives, according to which the data should be filtered.
criteriaIDs Vector containing IDs of criteria, according to which the data should be filtered.
solver String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.
timeLimit Allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to no time limit).
cplexIntegralityTolerance If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).
cplexThreads If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

Value

The function returns a list containing:

criteriaWeights The inferred criteria weights.
referenceProfiles The inferred reference profiles.
solverStatus The solver status as given by glpk or cplex.
humanReadableStatus A description of the solver status.

References


Examples

# the performance table

performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,10,9),c(10,9,9),
c(10,10,7),c(10,7,10),c(9,10,9),c(9,17,9),c(17,9,9),c(17,9,7),
c(7,10,17),c(10,17,7),c(17,7,10),c(17,10,7),c(10,7,17),
c(9,17,7),c(9,17,7),c(17,7,9),c(7,17,9),c(9,7,17))

lexicographicOrder <- c(2,1,3)

criteriaMinMax <- c("max","max","max")

rownames(performanceTable) <- c("a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11","a12","a13","a14","a15","a16","a17","a18","a19","a20","a21","a22","a23","a24")

colnames(performanceTable) <- c("c1","c2","c3")
SRMPInferenceNoInconsistFixedProfilesNumber

Exact inference of an SRMP model given the number of reference profiles - no inconsistencies

Description

Exact inference approach from pairwise comparisons of alternatives for the SRMP ranking model. This method only outputs a result when an SRMP model consistent with the provided pairwise comparisons exists. The number of reference profiles is fixed and need to be provided. If such a model exists, this method is significantly faster than the one which handles inconsistencies.

Usage

SRMPInferenceNoInconsistFixedProfilesNumber(performanceTable, criteriaMinMax, profilesNumber, preferencePairs, indifferencePairs = NULL, alternativesIDs = NULL, criteriaIDs = NULL, solver="glpk", timeLimit = NULL, cplexIntegralityTolerance = NULL, cplexThreads = NULL)

Arguments

performanceTable
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).

criteriaMinMax
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.
profilesNumber  A strictly positive numerical value which gives the number of reference profiles in the sought SRMP model.

preferencePairs  A two column matrix containing on each row a pair of alternative names where the first alternative is considered to be strictly preferred to the second.

indifferencePairs  A two column matrix containing on each row a pair of alternative names the two alternatives are considered to indifferent with respect to each other.

alternativesIDs  Vector containing IDs of alternatives, according to which the data should be filtered.

criteriaIDs  Vector containing IDs of criteria, according to which the data should be filtered.

solver  String specifying if the glpk solver (glpk) should be used, or the cplex (cplex) solver. By default glpk. The cplex solver requires to install the cplex binary and the cplex C API, as well as the cplexAPI R package.

timeLimit  Allows to fix a time limit of the execution, in seconds. By default NULL (which corresponds to no time limit).

cplexIntegralityTolerance  If the cplex solver is used, allows to fix a tolerance for integrality. By default NULL (which corresponds to the default value of cplex).

cplexThreads  If the cplex solver is used, allows to the number of threads for the calculation. By default NULL (which corresponds to the default value of cplex).

Value

The function returns a list containing:

criteriaWeights  The inferred criteria weights.

referenceProfiles  The inferred reference profiles.

lexicographicOrder  The inferred lexicographic order of the profiles.

solverStatus  The solver status as given by glpk or cplex.

humanReadableStatus  A description of the solver status.

References


Examples

performanceTable <- rbind(c(10,10,9),c(10,9,10),c(9,10,10),c(9,9,10),c(9,10,9),c(10,9,9),
c(10,10,7),c(10,7,10),c(7,10,10),c(9,9,17),c(9,17,9),c(17,9,9),...
TOPSIS

Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method

Description

TOPSIS is a multi-criteria decision analysis method which was originally developed by Hwang and Yoon in 1981.

Usage

```r
TOPSIS(performanceTable, criteriaWeights, criteriaMinMax, positiveIdealSolutions = NULL, negativeIdealSolutions = NULL, alternativesIDs = NULL, criteriaIDs = NULL)
```
Arguments

- **performanceTable**
  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).

- **criteriaWeights**
  Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.

- **criteriaMinMax**
  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **positiveIdealSolutions**
  Vector containing the positive ideal solutions for each criteria. The elements are named according to the IDs of the criteria.

- **negativeIdealSolutions**
  Vector containing the negative ideal solutions for each criteria. The elements are named according to the IDs of the criteria.

- **alternativesIDs**
  Vector containing IDs of alternatives, according to which the data should be filtered.

- **criteriaIDs**
  Vector containing IDs of criteria, according to which the data should be filtered.

Value

The function returns a vector containing the TOPSIS score for each alternative.

References


Examples

```r
performanceTable <- matrix(c(5490,51.4,8.5,285,6500,70.6,7, 288,6489,54.3,7.5,290), nrow=3, ncol=4, byrow=TRUE)
row.names(performanceTable) <- c("Corsa","Clio","Fiesta")
colnames(performanceTable) <- c("Purchase Price","Economy", "Aesthetics","Boot Capacity")
weights <- c(0.35,0.25,0.25,0.15)
criteriaMinMax <- c("min","max","max","max")
positiveIdealSolutions <- c(0.179573776, 0.171636015, 0.159499658, 0.087302767)
```
negativeIdealSolutions <- c(0.212610118, 0.124958799, 0.131352659, 0.085797547)

names(weights) <- colnames(performanceTable)
names(criteriaMinMax) <- colnames(performanceTable)
names(positiveIdealSolutions) <- colnames(performanceTable)
names(negativeIdealSolutions) <- colnames(performanceTable)

overall1 <- TOPSIS(performanceTable, weights, criteriaMinMax)
overall2 <- TOPSIS(performanceTable, weights, criteriaMinMax, positiveIdealSolutions, negativeIdealSolutions)
overall3 <- TOPSIS(performanceTable, weights, criteriaMinMax, alternativesIDs = c("Corsa","Clio"), criteriaIDs = c("Purchase Price","Economy","Aesthetics"))
overall4 <- TOPSIS(performanceTable, weights, criteriaMinMax, positiveIdealSolutions, negativeIdealSolutions, alternativesIDs = c("Corsa","Clio"), criteriaIDs = c("Purchase Price","Economy","Aesthetics"))

---

**UTA**

**UTA method to elicit value functions.**

**Description**

Elicits value functions from a ranking of alternatives, according to the UTA method.

**Usage**

```r
UTA(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, epsilon, alternativesRanks = NULL, alternativesPreferences = NULL, alternativesIndifferences = NULL, criteriaLBs=NULL, criteriaUBs=NULL, alternativesIDs = NULL, criteriaIDs = NULL, kPostOptimality = NULL)
```
Arguments

**performanceTable**
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).

**criteriaMinMax**
Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

**criteriaNumberOfBreakPoints**
Vector containing the number of breakpoints of the piecewise linear value functions to be determined. Minimum 2. The elements are named according to the IDs of the criteria.

**epsilon**
Numeric value containing the minimal difference in value between two consecutive alternatives in the final ranking.

**alternativesRanks**
Optional vector containing the ranks of the alternatives. The elements are named according to the IDs of the alternatives. If not present, then at least one of alternativesPreferences or alternativesIndifferences should be given.

**alternativesPreferences**
Optional matrix containing the preference constraints on the alternatives. Each line of the matrix corresponds to a constraint of the type alternative a is strictly preferred to alternative b. If not present, then either alternativesRanks or alternativesIndifferences should be given.

**alternativesIndifferences**
Optional matrix containing the indifference constraints on the alternatives. Each line of the matrix corresponds to a constraint of the type alternative a is indifferent to alternative b. If not present, then either alternativesRanks or alternativesIndifferences should be given.

**criteriaLBs**
Vector containing the lower bounds of the criteria to be considered for the elicitation of the value functions. If not specified, the lower bounds present in the performance table are taken.

**criteriaUBs**
Vector containing the upper bounds of the criteria to be considered for the elicitation of the value functions. If not specified, the upper bounds present in the performance table are taken.

**alternativesIDs**
Vector containing IDs of alternatives, according to which the data should be filtered.

**criteriaIDs**
Vector containing IDs of criteria, according to which the data should be filtered.

**kPostOptimality**
A small positive threshold used during the postoptimality analysis (see article on UTA by Siskos and Lagreze in EJOR, 1982). If not specified, no postoptimality analysis is performed.

Value

The function returns a list structured as follows:
optimum The value of the objective function.

valueFunctions A list containing the value functions which have been determined. Each value function is defined by a matrix of breakpoints, where the first row corresponds to the abscissa (row labelled "x") and where the second row corresponds to the ordinate (row labelled "y").

overallValues A vector of the overall values of the input alternatives.

ranks A vector of the ranks of the alternatives obtained via the elicited value functions. Ties method = "min".

Kendall Kendall’s tau between the input ranking and the one obtained via the elicited value functions. NULL if no input ranking is given but alternativesPreferences or alternativesIndifferences.

errors A vector of the errors (sigma) which have to be added to the overall values of the alternatives in order to respect the input ranking.

minimumWeightsPO In case a post-optimality analysis is performed, the minimal weight of each criterion, else NULL.

maximumWeightsPO In case a post-optimality analysis is performed, the maximal weight of each criterion, else NULL.

averageValueFunctionsPO In case a post-optimality analysis is performed, average value functions respecting the input ranking, else NULL.

References


Examples

# the separation threshold
epsilon <- 0.05

# the performance table
performanceTable <- rbind(
  c(3,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))

rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")

colnames(performanceTable) <- c("Price","Time","Comfort")
# ranks of the alternatives
alternativesRanks <- c(1,2,2,3,4)
names(alternativesRanks) <- row.names(performanceTable)

# criteria to minimize or maximize
criteriaMinMax <- c("min","min","max")
names(criteriaMinMax) <- colnames(performanceTable)

# number of break points for each criterion
criteriaNumberOfBreakPoints <- c(3,4,4)
names(criteriaNumberOfBreakPoints) <- colnames(performanceTable)

x<-UTA(performanceTable, criteriaMinMax,
       criteriaNumberOfBreakPoints, epsilon,
       alternativesRanks = alternativesRanks)

# plot the value functions obtained
plotPiecewiseLinearValueFunctions(x$valueFunctions)

# apply the value functions on the original performance table
transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(
    x$valueFunctions,
    performanceTable)

# calculate the overall score of each alternative
weightedSum(transformedPerformanceTable,c(1,1,1))

# ranking some cars (from original article on UTA by Siskos and Lagreze, 1982)

# the separation threshold
epsilon <-0.01

# the performance table
performanceTable <- rbind(
    c(173, 11.4, 10.01, 10, 7.88, 49500),
    c(176, 12.3, 10.48, 11, 7.96, 46700),
    c(142, 8.2, 7.30, 5, 5.65, 32100),
    c(148, 10.5, 9.61, 7, 6.15, 39150),
    c(178, 14.5, 11.05, 13, 8.06, 64700),
    c(180, 13.6, 10.40, 13, 8.47, 75700),
    c(182, 12.7, 12.26, 11, 7.81, 68593),
c(145, 14.3, 12.95, 11, 8.38, 55000),
c(161, 8.6, 8.42, 7, 5.11, 35200),
c(117, 7.2, 6.75, 3, 5.81, 24800)
)
rownames(performanceTable) <- c(
  "Peugeot 505 GR",
  "Opel Record 2000 LS",
  "Citroen Visa Super E",
  "VW Golf 1300 GLS",
  "Citroen CX 2400 Pallas",
  "Mercedes 230",
  "BMW 520",
  "Volvo 244 DL",
  "Peugeot 104 ZS",
  "Citroen Dyane"
)

colnames(performanceTable) <- c(
  "MaximalSpeed",
  "ConsumptionTown",
  "Consumption120kmh",
  "HP",
  "Space",
  "Price"
)

# ranks of the alternatives
alternativesRanks <- c(1,2,3,4,5,6,7,8,9,10)
names(alternativesRanks) <- row.names(performanceTable)

# criteria to minimize or maximize
criteriaMinMax <- c("max","min","min","max","max","min")
names(criteriaMinMax) <- colnames(performanceTable)

# number of break points for each criterion
criteriaNumberOfBreakPoints <- c(5,4,4,5,4,5)
names(criteriaNumberOfBreakPoints) <- colnames(performanceTable)

# lower bounds of the criteria for the determination of value functions
criteriaLBs=c(110,7,6,3,5,20000)
names(criteriaLBs) <- colnames(performanceTable)

# upper bounds of the criteria for the determination of value functions
criteriaUBs=c(190,15,13,13,9,80000)
names(criteriaUBs) <- colnames(performanceTable)

x<-UTA(performanceTable, criteriaMinMax,
       criteriaNumberOfBreakPoints, epsilon,
       alternativesRanks = alternativesRanks,
       criteriaLBs = criteriaLBs, criteriaUBs = criteriaUBs)

# plot the value functions obtained
plotPiecewiseLinearValueFunctions(x$valueFunctions)

# apply the value functions on the original performance table
transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(
    x$valueFunctions,
    performanceTable)

# calculate the overall score of each alternative
weights<-c(1,1,1,1,1,1)
names(weights)<-colnames(performanceTable)
weightedSum(transformedPerformanceTable,c(1,1,1,1,1,1))

# the same analysis with less extreme value functions
# from the post-optimality analysis
x<-UTA(performanceTable, criteriaMinMax,
       criteriaNumberOfBreakPoints, epsilon,
       alternativesRanks = alternativesRanks,
       criteriaLBs = criteriaLBs,
       criteriaUBs = criteriaUBs,
       kPostOptimality = 0.01)

# plot the value functions obtained
plotPiecewiseLinearValueFunctions(x$averageValueFunctionsPO)

# apply the value functions on the original performance table
transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(
    x$averageValueFunctionsPO,
    performanceTable)

# calculate the overall score of each alternative
weights<-c(1,1,1,1,1,1)
names(weights)<-colnames(performanceTable)
weightedSum(transformedPerformanceTable,c(1,1,1,1,1,1))
# Let us consider only 2 criteria : Price and MaximalSpeed. What happens ?

```r
# x<-UTA(performanceTable, criteriaMinMax, 
# criteriaNumberOfBreakPoints, epsilon, 
# alternativesRanks = alternativesRanks, 
# criteriaLBs = criteriaLBs, criteriaUBs = criteriaUBs, 
# criteriaIDs = c("MaximalSpeed","Price"))
```

# plot the value functions obtained
```r
# plotPiecewiseLinearValueFunctions(x$valueFunctions, 
# criteriaIDs = c("MaximalSpeed","Price"))
```

# apply the value functions on the original performance table
```r
# transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(
# x$valueFunctions, 
# performanceTable, 
# criteriaIDs = c("MaximalSpeed","Price")
# )
```

# calculate the overall score of each alternative
```r
# weights<-c(1,1,1,1,1,1)
# names(weights)<-colnames(performanceTable)
# weightedSum(transformedPerformanceTable, 
# weights, criteriaIDs = c("MaximalSpeed","Price"))
```

# An example without alternativesRanks, but with alternativesPreferences 
# and alternativesIndifferences
```r
alternativesPreferences <- rbind(c("Peugeot 505 GR","Opel Record 2000 LS"), 
                                 c("Opel Record 2000 LS","Citroen Visa Super E"))
```

```r
alternativesIndifferences <- rbind(c("Peugeot 104 ZS","Citroen Dyane"))
```

```r
x<-UTA(performanceTable, criteriaMinMax, 
criteriaNumberOfBreakPoints, epsilon = 0.1, 
alternativesPreferences = alternativesPreferences, 
alternativesIndifferences = alternativesIndifferences, 
criteriaLBs = criteriaLBs, criteriaUBs = criteriaUBs)
```
**UTADIS**

**UTADIS method to elicit value functions in view of sorting alternatives in ordered categories**

**Description**

Elicits value functions from assignment examples, according to the UTADIS method.

**Usage**

```r
UTADIS(performanceTable, criteriaMinMax, 
criteriaNumberOfBreakPoints, 
alternativesAssignments, categoriesRanks, epsilon, 
criteriaLBs=NULL, criteriaUBs=NULL, 
alternativesIDs = NULL, criteriaIDs = NULL, 
categoriesIDs = NULL)
```

**Arguments**

- **performanceTable**
  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).

- **criteriaMinMax**
  Vector containing the preference direction on each of the criteria. "min" (resp. "max") indicates that the criterion has to be minimized (maximized). The elements are named according to the IDs of the criteria.

- **criteriaNumberOfBreakPoints**
  Vector containing the number of breakpoints of the piecewise linear value functions to be determined. Minimum 2. The elements are named according to the IDs of the criteria.

- **alternativesAssignments**
  Vector containing the assignments of the alternatives to categories. Minimum 2 categories. The elements of the vector are named according to the IDs of the alternatives.

- **categoriesRanks**
  Vector containing the ranks of the categories. Minimum 2 categories. The elements of the vector are named according to the IDs of the categories.

- **epsilon**
  Numeric value containing the minimal difference in value between the upper bound of a category and an alternative of that category.

- **criteriaLBs**
  Vector containing the lower bounds of the criteria to be considered for the elicitation of the value functions. If not specified, the lower bounds present in the performance table are taken.

- **criteriaUBs**
  Vector containing the upper bounds of the criteria to be considered for the elicitation of the value functions. If not specified, the upper bounds present in the performance table are taken.
The function returns a list structured as follows:

- **optimum**: The value of the objective function.
- **valueFunctions**: A list containing the value functions which have been determined. Each value function is defined by a matrix of breakpoints, where the first row corresponds to the abscissa (row labelled "x") and where the second row corresponds to the ordinate (row labelled "y").
- **overallValues**: A vector of the overall values of the input alternatives.
- **categoriesLBs**: A vector containing the lower bounds of the considered categories.
- **errors**: A list containing the errors (sigmaPlus and sigmaMinus) which have to be subtracted and added to the overall values of the alternatives in order to respect the input ranking.

References


Examples

```r
# the separation threshold
epsilon <- 0.05

# the performance table
performanceTable <- rbind(
  c(3,10,1),
  c(4,20,2),
  c(2,20,0),
  c(6,40,0),
  c(30,30,3))
rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")
colnames(performanceTable) <- c("Price","Time","Comfort")

# ranks of the alternatives
alternativesAssignments <- c("good","medium","medium","bad","bad")
```
names(alternativesAssignments) <- row.names(performanceTable)

# criteria to minimize or maximize
criteriaMinMax <- c("min","min","max")

names(criteriaMinMax) <- colnames(performanceTable)

# number of break points for each criterion
criteriaNumberOfBreakPoints <- c(3,4,4)

names(criteriaNumberOfBreakPoints) <- colnames(performanceTable)

# ranks of the categories
categoriesRanks <- c(1,2,3)

names(categoriesRanks) <- c("good","medium","bad")

x<-UTADIS(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, alternativesAssignments, categoriesRanks,0.1)

# filtering out category "good" and assignment examples "RER" and "TAXI"
y<-UTADIS(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, alternativesAssignments, categoriesRanks,0.1, categoriesIDs=c("medium","bad"), alternativesIDs=c("METRO1","METRO2","BUS"))

# working furthermore on only 2 criteria : "Comfort" and "Time"
z<-UTADIS(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, alternativesAssignments, categoriesRanks,0.1, criteriaIDs=c("Comfort","Time"))

---

**UTASTAR**

**UTASTAR method to elicit value functions.**

**Description**

Elicits value functions from a ranking of alternatives, according to the UTASTAR method.

**Usage**

UTASTAR(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, epsilon, alternativesRanks = NULL, alternativesPreferences = NULL,
alternativesIndifferences = NULL,
criteriaLBs=NULL, criteriaUBs=NULL,
alternativesIDs = NULL, criteriaIDs = NULL,
kPostOptimality = NULL)

Arguments

performanceTable
  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).

criteriaMinMax
  Vector containing the preference direction on each of the criteria. "min" (resp.
  "max") indicates that the criterion has to be minimized (maximized). The ele-
ments are named according to the IDs of the criteria.

criteriaNumberOfBreakPoints
  Vector containing the number of breakpoints of the piecewise linear value func-
tions to be determined. Minimum 2. The elements are named according to the
IDs of the criteria.

epsilon
  Numeric value containing the minimal difference in value between two consec-
utive alternatives in the final ranking.

alternativesRanks
  Optional vector containing the ranks of the alternatives. The elements are named
according to the IDs of the alternatives. If not present, then at least one of
alternativesPreferences or alternativesIndifferences should be given.

alternativesPreferences
  Optional matrix containing the preference constraints on the alternatives. Each
line of the matrix corresponds to a constraint of the type alternative a is strictly
preferred to alternative b. If not present, then either alternativesRanks or alter-
nativesIndifferences should be given.

alternativesIndifferences
  Optional matrix containing the indifference constraints on the alternatives. Each
line of the matrix corresponds to a constraint of the type alternative a is indifferent
to alternative b. If not present, then either alternativesRanks or alternativesPreferences should be given.

criteriaLBs
  Vector containing the lower bounds of the criteria to be considered for the elic-
tiation of the value functions. If not specified, the lower bounds present in the
performance table are taken.

criteriaUBs
  Vector containing the upper bounds of the criteria to be considered for the elic-
tiation of the value functions. If not specified, the upper bounds present in the
performance table are taken.

alternativesIDs
  Vector containing IDs of alternatives, according to which the datashould be fil-
tered.

criteriaIDs
  Vector containing IDs of criteria, according to which the data should be filtered.
kPostOptimality
A small positive threshold used during the postoptimality analysis (see article on UTA by Siskos and Lagreze in EJOR, 1982). If not specified, no postoptimality analysis is performed.

Value
The function returns a list structured as follows:

- **optimum**: The value of the objective function.
- **valueFunctions**: A list containing the value functions which have been determined. Each value function is defined by a matrix of breakpoints, where the first row corresponds to the abscissa (row labelled "x") and where the second row corresponds to the ordinate (row labelled "y").
- **overallValues**: A vector of the overall values of the input alternatives.
- **ranks**: A vector of the ranks of the alternatives obtained via the elicited value functions. Ties method = "min".
- **Kendall**: Kendall’s tau between the input ranking and the one obtained via the elicited value functions.
- **errors**: A list containing the errors (sigmaPlus and sigmaMinus) which have to be subtracted and added to the overall values of the alternatives in order to respect the input ranking.
- **minimumWeightsPO**: In case a post-optimality analysis is performed, the minimal weight of each criterion, else NULL.
- **maximumWeightsPO**: In case a post-optimality analysis is performed, the maximal weight of each criterion, else NULL.
- **averageValueFunctionsPO**: In case a post-optimality analysis is performed, average value functions respecting the input ranking, else NULL.

References

Examples
```
# the separation threshold
epsilon <- 0.05

# the performance table
# the performance table
performanceTable <- rbind(
  c(3, 10, 1),
  c(4, 20, 2),
```
c(220, 0),
c(640, 0),
c(3030, 3))

rownames(performanceTable) <- c("RER","METRO1","METRO2","BUS","TAXI")

colnames(performanceTable) <- c("Price","Time","Comfort")

# ranks of the alternatives
alternativesRanks <- c(1,2,2,3,4)
names(alternativesRanks) <- row.names(performanceTable)

# criteria to minimize or maximize
criteriaMinMax <- c("min","min","max")
names(criteriaMinMax) <- colnames(performanceTable)

# number of break points for each criterion
criteriaNumberOfBreakPoints <- c(3,4,4)
names(criteriaNumberOfBreakPoints) <- colnames(performanceTable)

x<-UTASTAR(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, epsilon, alternativesRanks = alternativesRanks)

# plot the value functions obtained
plotPiecewiseLinearValueFunctions(x$valueFunctions)

# apply the value functions on the original performance table
transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(x$valueFunctions, performanceTable)

# calculate the overall score of each alternative
weightedSum(transformedPerformanceTable,c(1,1,1))

# ----------------------------------------
# ranking some cars (from original article on UTA by Siskos and Lagreze, 1982)
# the separation threshold
epsilon <-0.01

# the performance table
```r
performanceTable <- rbind(
  c(173, 11.4, 10.01, 10, 7.88, 49500),
  c(176, 12.3, 10.48, 11, 7.96, 46700),
  c(142, 8.2, 7.30, 5, 5.65, 32100),
  c(148, 10.5, 9.61, 7, 6.15, 39150),
  c(178, 14.5, 11.05, 13, 8.06, 64700),
  c(180, 13.6, 10.40, 13, 8.47, 75700),
  c(182, 12.7, 12.26, 11, 7.81, 68593),
  c(145, 14.3, 12.95, 11, 8.38, 55000),
  c(161, 8.6, 8.42, 7, 5.11, 35200),
  c(117, 7.2, 6.75, 3, 5.81, 24800)
)
rownames(performanceTable) <- c(
  "Peugeot 505 GR",
  "Opel Record 2000 LS",
  "Citroen Visa Super E",
  "VW Golf 1300 GLS",
  "Citroen CX 2400 Pallas",
  "Mercedes 230",
  "BMW 520",
  "Volvo 244 DL",
  "Peugeot 104 ZS",
  "Citroen Dyane"
)
colnames(performanceTable) <- c(
  "MaximalSpeed",
  "ConsumptionTown",
  "Consumption120kmh",
  "HP",
  "Space",
  "Price"
)

# ranks of the alternatives
alternativesRanks <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

# criteria to minimize or maximize
criteriaMinMax <- c("max", "min", "min", "max", "max", "min")

# number of break points for each criterion
criteriaNumberOfBreakPoints <- c(5, 4, 4, 5, 4, 5)

# lower bounds of the criteria for the determination of value functions
```
criteriaLBs <- c(110, 7, 6, 3, 5, 20000)

names(criteriaLBs) <- colnames(performanceTable)

# upper bounds of the criteria for the determination of value functions
criteriaUBs <- c(190, 15, 13, 13, 9, 80000)

names(criteriaUBs) <- colnames(performanceTable)
x <- UTASTAR(performanceTable, criteriaMinMax,
criteriaNumberOfBreakPoints, epsilon,
alternativesRanks = alternativesRanks,
criteriaLBs = criteriaLBs, criteriaUBs = criteriaUBs)

# plot the value functions obtained
plotPiecewiseLinearValueFunctions(x$valueFunctions)

# apply the value functions on the original performance table
transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(
x$valueFunctions,
performanceTable)

# calculate the overall score of each alternative
weights <- c(1, 1, 1, 1, 1)

names(weights) <- colnames(performanceTable)

weightedSum(transformedPerformanceTable, c(1, 1, 1, 1, 1))

# the same analysis with less extreme value functions
# from the post-optimality analysis
x <- UTASTAR(performanceTable, criteriaMinMax,
criteriaNumberOfBreakPoints, epsilon,
alternativesRanks = alternativesRanks,
criteriaLBs = criteriaLBs,
criteriaUBs = criteriaUBs,
kPostOptimality = 0.01)

# plot the value functions obtained
plotPiecewiseLinearValueFunctions(x$averageValueFunctionsPO)

# apply the value functions on the original performance table
transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(
x$averageValueFunctionsPO,
performanceTable)
# calculate the overall score of each alternative

weights<-c(1,1,1,1,1,1)

names(weights)<-colnames(performanceTable)

weightedSum(transformedPerformanceTable,c(1,1,1,1,1,1))

# ----------------------------------------
# Let us consider only 2 criteria : Price and MaximalSpeed. What happens ?

x<-UTASTAR(performanceTable, criteriaMinMax, criteriaNumberOfBreakPoints, epsilon, alternativesRanks = alternativesRanks, criteriaLBs = criteriaLBs, criteriaUBs = criteriaUBs, criteriaIDs = c("MaximalSpeed","Price"))

# plot the value functions obtained

plotPiecewiseLinearValueFunctions(x$valueFunctions, criteriaIDs = c("MaximalSpeed","Price"))

# apply the value functions on the original performance table

transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(x$valueFunctions, performanceTable, criteriaIDs = c("MaximalSpeed","Price"))

# calculate the overall score of each alternative

weights<-c(1,1,1,1,1,1)

names(weights)<-colnames(performanceTable)

weightedSum(transformedPerformanceTable, weights, criteriaIDs = c("MaximalSpeed","Price"))

# ----------------------------------------
# An example without alternativesRanks, but with alternativesPreferences # and alternativesIndifferences

alternativesPreferences <- rbind(c("Peugeot 505 GR","Opel Record 2000 LS"), c("Opel Record 2000 LS","Citroen Visa Super E"))

alternativesIndifferences <- rbind(c("Peugeot 104 ZS","Citroen Dyane"))

x<-UTASTAR(performanceTable, criteriaMinMax,
weightedSum

Weighted sum of evaluations of alternatives.

Description
Computes the weighted sum of the evaluations of alternatives, stored in a performance table, with respect to a vector of criteria weights.

Usage
weightedSum(performanceTable, criteriaWeights, 
alternativesIDs = NULL, criteriaIDs = NULL)

Arguments
performanceTable
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).
criteriaWeights
Vector containing the weights of the criteria. The elements are named according to the IDs of the criteria.
alternativesIDs
Vector containing IDs of alternatives, according to which the performance table should be filtered.
criteriaIDs
Vector containing IDs of criteria, according to which the performance table should be filtered.

Value
The function returns a vector containing the weighted sum of the alternatives with respect to the criteria weights.

Examples
performanceTable <- matrix(runif(3*4), ncol=3)
row.names(performanceTable) <- c("x1","x2","x3","x4")
colnames(performanceTable) <- c("g1","g2","g3")
weights <- c(1,2,3)

names(weights) <- c("g1","g2","g3")

overall1 <- weightedSum(performanceTable, weights)

overall2 <- weightedSum(performanceTable, weights,
   alternativesIDs <- c("x2","x3"), criteriaIDs <- c("g2","g3"))
Index

* methods
  additiveValueFunctionElicitation, 2
  applyPiecewiseLinearValueFunctionsOnPerformanceTable, 6
  assignAlternativesToCategoriesByThresholds, 8
  LPDMRSort, 11
  LPDMRSortIdentifyIncompatibleAssignments, 14
  LPDMRSortIdentifyUsedDictatorProfiles, 18
  LPDMRSortIdentifyUsedVetoProfiles, 21
  LPDMRSortInferenceApprox, 24
  LPDMRSortInferenceExact, 26
  MRSort, 31
  MRSortIdentifyIncompatibleAssignments, 34
  MRSortIdentifyUsedVetoProfiles, 37
  MRSortInferenceApprox, 39
  MRSortInferenceExact, 41
  normalizePerformanceTable, 46
  plotAlternativesValuesPreorder, 48
  plotMRSortSortingProblem, 50
  plotPiecewiseLinearValueFunctions, 53
  plotRadarPerformanceTable, 54
  SRMP, 63
  SRMPInference, 64
  SRMPInferenceApprox, 66
  SRMPInferenceApproxFixedLexicographicOrder, 68
  SRMPInferenceApproxFixedProfilesNumber, 71
  SRMPInferenceFixedLexicographicOrder, 73
  SRMPInferenceFixedProfilesNumber, 75
  SRMPInferenceNoInconsist, 77
  SRMPInferenceNoInconsistFixedLexicographicOrder, 80
  SRMPInferenceNoInconsistFixedProfilesNumber, 82
  UTA, 86
  UTADIS, 93
  UTASTAR, 95
  weightedSum, 102

additiveValueFunctionElicitation, 2
applyPiecewiseLinearValueFunctionsOnPerformanceTable, 6
assignAlternativesToCategoriesByThresholds, 8
ELECTREIIIIdistillation, 10
LPDMRSort, 11
LPDMRSortIdentifyIncompatibleAssignments, 14
LPDMRSortIdentifyUsedDictatorProfiles, 17
LPDMRSortIdentifyUsedVetoProfiles, 21
LPDMRSortInferenceApprox, 24
LPDMRSortInferenceExact, 26
MARE, 30
MRSort, 31
MRSortIdentifyIncompatibleAssignments, 34
MRSortIdentifyUsedVetoProfiles, 37
MRSortInferenceApprox, 39
MRSortInferenceExact, 41
MRSortInterval, 44
normalizePerformanceTable, 46
pairwiseConsistencyMeasures, 48
plotAlternativesValuesPreorder, 48
plotMARE, 49
plotMRSortSortingProblem, 50
plotPiecewiseLinearValueFunctions, 53
plotRadarPerformanceTable, 54
PROMETHEE I, 55
PROMETHEE II, 57
PROMETHEE Outranking Flows, 59
PROMETHEE Preference Indices, 61

SRMP, 63
SRMP Inference, 64
SRMP Inference Approx, 66
SRMP Inference Approx Fixed Lexicographic Order, 68
SRMP Inference Approx Fixed Profiles Number, 71
SRMP Inference Fixed Lexicographic Order, 73
SRMP Inference Fixed Profiles Number, 75
SRMP Inference No Inconsist, 77
SRMP Inference No Inconsist Fixed Lexicographic Order, 80
SRMP Inference No Inconsist Fixed Profiles Number, 82

TOPSIS, 84

UTA, 86
UTADIS, 93
UTASTAR, 95

weightedSum, 102