UTASTAR

From MCDA v0.0.7
by Patrick Meyer

UTASTAR Method To Elicit Value Functions.

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

Keywords methods

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 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 alternativesI

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 alternativesPrefer

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 datashould 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 :

optimumThe value of the objective function.

valueFunctionsA 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").

overallValuesA vector of the overall values of the input alternatives.

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

KendallKendall's tau between the input ranking and the one obtained via the elicited value functions.

errorsA list containing the errors (sigmaPlus and sigmaMinus) which have to be substracted and added to the overall values of the alternatives in order to

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

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

References

Siskos, Y. and D. Yannacopoulos, UTASTAR: An ordinal regression method for building additive value functions, Investigacao Operacional , 5 (1), 39--53, 1985.

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<-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,
  criteriaMinMax)

# 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(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<-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,
      criteriaMinMax)

# 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<-UTASTAR(performanceTable, criteriaMinMax,
        criteriaNumberOfBreakPoints, epsilon,
```

```
# plot the value functions obtained

plotPiecewiseLinearValueFunctions(x$averageValueFunctionsPO)

# apply the value functions on the original performance table

transformedPerformanceTable <- applyPiecewiseLinearValueFunctionsOnPerformanceTable(
      x$averageValueFunctionsPO,
      performanceTable,
      criteriaMinMax)

# 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,
  criteriaMinMax,
  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,
        criteriaNumberOfBreakPoints, epsilon = 0.1,
        alternativesPreferences = alternativesPreferences,
        alternativesIndifferences = alternativesIndifferences,
        criteriaLBs = criteriaLBs, criteriaUBs = criteriaUBs
        )
```


Community examples

Looks like there are no examples yet.

New example
Use markdown to format your example

R code blocks are runnable and interactive:
```r
a <- 2
print(a)
```

You can also display normal code blocks
```
var a = b
```

Submit your example

API documentation

Created by DataCamp.com