Package ‘mau’

October 13, 2022

Type Package
Version 0.1.2
Title Decision Models with Multi Attribute Utility Theory
Encoding UTF-8
Date 2018-01-17
Description Provides functions for the creation, evaluation and test of decision models based in Multi Attribute Utility Theory (MAUT). Can process and evaluate local risk aversion utilities for a set of indexes, compute utilities and weights for the whole decision tree defining the decision model and simulate weights employing Dirichlet distributions under addition constraints in weights.
Maintainer Pedro Guarderas <pedro.felipe.guarderas@gmail.com>
License LGPL-3
URL https://github.com/pedroguarderas/mau
Depends R (>= 3.0)
Imports data.table, gtools, stringr, igraph, RColorBrewer, ggplot2, Rdpack
RoxygenNote 6.0.1
Suggests knitr, rmarkdown
VignetteBuilder knitr
RdMacros Rdpack
NeedsCompilation no
Author Felipe Aguirre [ctb], Julio Andrade [ctb], Pedro Guarderas [aut, cre], Daniel Lagos [ctb], Andrés Lopez [ctb], Nelson Recalde [ctb], Edison Salazar [ctb]
Repository CRAN
Date/Publication 2018-01-17 05:35:14 UTC
Description

Provides functions for the creation, evaluation and test of decision models based in Multi Attribute Utility Theory (MAUT).

Details

MAUT models are defined employing a decision tree where similarity relations between different index utilities are defined, this helps to group utilities following a criteria of similarity. Each final node has an utility and weight associated, the utility of any internal node in the decision tree is computed by adding the weighted sum of each of its final nodes. In a model with \( n \) indexes, a criteria is composed by \( C \subset \{1, \ldots, n\} \), the respective utility is given by:

\[
\sum_{i \in C} w_i u_i(x_i)
\]

Currently, each utility is defined like a piecewise risk aversion utility, those functions are of the following form:

\[
ax + b
\]

or

\[
ae^{cx} + b
\]

The current capabilities of \texttt{mau} are:
1. Read a list of risk aversion utilities defined in a standardized format.
2. Evaluate utilities of a table of indexes.
3. Load decision trees defined in column standard format.
4. Compute criteria utilities and weights for any internal node of the decision tree.
5. Simulate weights employing Dirichlet distributions under addition constraints in weights.

Author(s)

Maintainer: Pedro Guarderas <pedro.felipe.guarderas@gmail.com>

Other contributors:

• Felipe Aguirre [contributor]
• Julio Andrade [contributor]
• Daniel Lagos [contributor]
• Andrés Lopez [contributor]
• Nelson Recalde [contributor]
• Edison Salazar [contributor]

References


See Also

Useful links:

• https://github.com/pedroguarderas/mau

Examples

library(mau)
vignette(topic = 'Running_MAUT', package = 'mau')
Bar.Plot

Bar plot of utilities

Description
Create ggplot2 bar plots of the utilities at any level of the decision model.

Usage
Bar.Plot(model, deep, colors, title, xlab, ylab)

Arguments
- model: data.table obtained with Compute.Model
- deep: the deep to navigate the model object to select the utilities
- colors: a list of colors for the bars
- title: title for the bar plot
- xlab: label for horizontal axis
- ylab: label for vertical axis

Value
ggplot2 object.

Author(s)
Pedro Guarderas <pedro.felipe.guarderas@gmail.com>

Examples
vignette(topic = 'Running_MAUT', package = 'mau')

Compute.Model

Evaluation of decision tree nodes

Description
Evaluation of decision tree nodes. All the MAUT model is computed at every level the utilities are computed considering the given weights.

Usage
Compute.Model(tree, utilities, weights)
**Deep.Compute**

**Arguments**
- **tree**: initial tree structure with utilities in its leaves.
- **utilities**: data.table with ordered columns containing the values of utilities.
- **weights**: weights for the decision model.

**Details**
The whole decision model can be computed at any level and represented in a table format.

**Value**
data.table structure containing the utilities of the model for every level of the decision tree.

**Author(s)**
Pedro Guarderas, Andrés Lopez <pedro.felipe.guarderas@gmail.com>

**See Also**

**Examples**
```r
vignette(topic = 'Running_MAUT', package = 'mau')
```

---

**Compute the deep position of every node**

**Description**
For the computation of the complete decision model is necessary to establish the deep position of every node.

**Usage**
```r
Deep.Compute(tree)
```

**Arguments**
- **tree**: igraph object representing the tree

**Value**
igraph object updated

**Author(s)**
Pedro Guarderas, Andrés Lopez
Divide.Weights  

*Divide weights of internal nodes*

**Description**

After the addition of weights for internal nodes the final weights have to be computed dividing by the total weight of each parent.

**Usage**

```r
divide.weights(tree)
```

**Arguments**

- `tree` igraph object representing the tree

**Value**

igraph object updated

**Author(s)**

Pedro Guarderas, Andrés Lopez

**See Also**

- `read.tree`

---

Eval.Utilities  

*Evaluate utilities*

**Description**

Evaluation of utilities for a data.table of indexes, the utilities functions are computed over every index represented by each column of the input table.

**Usage**

```r
eval.utilities(index, columns, functions)
```
Eval.Utilities

Arguments

index data.table of indexes.
columns columns with indexes where the utilities will be computed.
functions vector of characters with name of functions.

Details

Every index has associated an utility function, inside mau is possible to employ any functions, the only special requirement is that the utility has to be normalized, this means that the utility is bounded between 0 and 1.

Also is possible to consider utilities with constant risk aversion CRA, in the sense of Arrow, for such case there is only two types of functions $u(x) = ax + b$ or $u(x) = ae^{bx} + c$, to determine these functions, it is only necessary to specify the parameters $a$, $b$ and $c$. For a decision model only elaborated with CRA utilities, mau could read a text file where every utility is piecewise defined.

The format for the text file containing the definition of utility functions is given by is:

```
[Header]
[Function name]
[min1 max1 a1 b1 c1]
[min2 max2 a2 b2 c2]
[min3 max3 a3 b3 c3]
...
[Function name]
[min1 max1 a1 b1 c1]
[min2 max2 a2 b2 c2]
[min3 max3 a3 b3 c3]
...
```

If the coefficient $c$ is non zero the function is interpreted as an exponential type.

Value
data.table with utilities evaluated for every index.

Author(s)

Pedro Guarderas, <pedro.felipe.guarderas@gmail.com>, Andrés Lopez.

See Also

Read.Utilities, Stand.String

Examples

```r
library( mau )
vignette( topic = 'Running_MAUT', package = 'mau' )
```
Index.Weights

Compute leaves weights

Description
The computation of weights could be determined in an inverse processes given the internal weights.

Usage
Index.Weights(tree)

Arguments

| tree | igraph object representing the tree |

Value
igraph object updated

Author(s)
Pedro Guarderas, Andrés Lopez

See Also
Read.Tree

Make.Decision.Tree

Evaluate utilities

Description
Create decision tree for MAUT models exporting to an igraph object.

Usage
Make.Decision.Tree(tree.data)

Arguments

| tree.data | data.table with decision tree information. |

Details
With the tree information loaded by the Read.Tree the decision tree could be represented like an igraph object.
Plot.Simulation.Weight

Value

igraph object containing the graph of the decision tree.

Author(s)

Pedro Guarderas, Andrés Lopez <pedro.felipe.guarderas@gmail.com>

See Also

Read.Tree

Examples

```r
library( data.table )
library( igraph )
file<-system.file("extdata", "tree.csv", package = "mau" )
tree.data<-Read.Tree( file, skip = 0, nrows = 8 )
tree<-Make.Decision.Tree( tree.data )
plot( tree )
```

---

Plot.Simulation.Weight

Plot decision MAUT model with weights simulations

Description

Spider plot for the decision model considering the weights simulated with a Dirichlet distributions, every simulation is represented with lines, a box plot is included to account the behavior of every global utility.

Usage

```r
```

Arguments

- `S`: first element of the simulation list produced by the function `Sim.Weights, Sim.Const.Weights`
- `title`: text for the title plot.
- `xlab`: text for x-axis label.
- `ylab`: text for y-axis label.
- `lines.cols`: the spectrum of colors for the simulation is selected randomly from a base color.
- `box.col`: color for the boxes.
Read.Tree

- `box.outlier.col`: color for the outlier points representing the extreme observations in the boxplot.
- `utility.col`: the main utility value is also plotted with this specific color.
- `utility.point.col`: the line of main utilities is plotted with points represented with this color.
- `text.col`: color for the text values plotted for each utility.

**Value**

- ggplot object with the plot of simulations.

**Author(s)**

- Pedro Guarderas

**See Also**

- `Sim.Const.Weights`
- `Sim.Weights`

---

**Description**

Read a csv file where the decision tree is defined.

**Usage**

```r
Read.Tree(file, skip, nrows)
```

**Arguments**

- `file`: input csv file containing the tree.
- `skip`: starting row for read.
- `nrows`: number of rows to read.

**Value**

- data.table with utilities.

**Author(s)**

- Pedro Guarderas, Andrés Lopez

**See Also**

- `Read.Utilities`
- `Make.Decision.Tree`
**Examples**

```r
library( data.table )
library( igraph )
file<-system.file("extdata", "tree.csv", package = "mau" )
sheetIndex<-1
tree.data<-Read.Tree( file, skip = 0, nrows = 8 )
```

---

**Description**

Builds utility functions from definition standard.

**Usage**

```r
Read.Utilities(file, script, lines, skip = 2, encoding = "utf-8")
```

**Arguments**

- `file`: standardize file with definitions.
- `script`: output script where the utility functions are defined automatically.
- `lines`: number lines to read in file.
- `skip`: to read the file it had to skip a given number of lines.
- `encoding`: file encoding.

**Details**

The basic MAUT models are built with functions of constant absolute risk aversion, this functions could be defined with simple parameters, only is necessary a function name and the domain of definition of every function and more important is necessary no more than three coefficients for the function definition.

**Value**

Returns data table with definition of utility functions by range.

**Author(s)**

Pedro Guarderas, Andrés Lopez

**See Also**

`Stand.String`
Examples

```r
library( data.table )
file<-system.file("extdata", "utilities.txt", package = "mau" )
script<-'utilities.R'
lines<-17
skip<-2
encoding<-'utf-8'
functions<-ReadUtilities( file, script, lines, skip, encoding )
```

Sim.Const.Weights

**Simulation of constrained weights**

**Description**

Simulation of weights employing the Dirichlet distribution. The concentration parameters for the Dirichlet distribution are tentative weights, additionally constraints over partial sums of weights are introduced by a list ordered structure.

**Usage**

```
Sim.Const.Weights(n, utilities, alpha, constraints)
```

**Arguments**

- `n`: number of simulations
- `utilities`: utility dataframe, first column is the identifier
- `alpha`: concentration parameter for the Dirichlet distribution
- `constraints`: list of sum constraints

**Details**

Employing the properties of the Dirichlet distribution, weights could be simulated with a given concentration, additionally this simulation can be carry out by subsets of weights only to meet specific constraints.

**Value**

List with data.frames {simulation, weights} with total utilities and simulated weights

**Author(s)**

Pedro Guarderas <pedro.felipe.guarderas@gmail.com>

**See Also**

`Eval.Utilities`
Sim.Weights

Examples

```r
library( data.table )
N<-10
utilities<-data.table( id = 1:N,
  u1 = runif( N, 0, 1 ),
  u2 = runif( N, 0, 1 ),
  u3 = runif( N, 0, 1 ),
  u4 = runif( N, 0, 1 ) )
n<-100
alpha<-c( 0.2, 0.5, 0.1, 0.2 )
constraints<-list( list( c(1,2), 0.7 ),
  list( c(3,4), 0.3 ) )
S<-Sim.Const.Weights( n, utilities, alpha, constraints )
plot.S<-Plot.Simulation.Weight( S$simulation, title = 'Simulations',
  xlab = 'ID', ylab = 'Utility' )
plot( plot.S )
```

Sim.Weights

Description

Simulation of weights employing the Dirichlet distribution. The concentration parameters for the Dirichlet distribution are tentative weights.

Usage

```r
Sim.Weights(n, utilities, alpha)
```

Arguments

- `n` number of simulations
- `utilities` utility dataframe, first column is the identifier
- `alpha` concentration parameter for the Dirichlet distribution

Details

Taking advantage of the Dirichlet distribution properties, the weights could be simulated with a concentration around given weights.

Value

List with data.frames {simulation, weights} with total utilities and simulated weights

Author(s)

Pedro Guarderas <pedro.felipe.guarderas@gmail.com>
See Also

`Eval.Utilities`

Examples

```r
library( data.table )
N<-10
utilities<-data.table( id = 1:N,
                         u1 = runif( N, 0, 1 ),
                         u2 = runif( N, 0, 1 ),
                         u3 = runif( N, 0, 1 ),
                         u4 = runif( N, 0, 1 ) )

n<-100
alpha<-c( 0.2, 0.5, 0.1, 0.2 )
S<-Sim.Weights( n, utilities, alpha )
```

Description

Generates a spider plot for a decision model

Usage

```r
Spider.Plot(data, data.label, data.fill, data.color, data.linetype, data.alpha, 
data.size, data.label.color, data.label.size, group, criteria, valor, title, 
title.color, title.size, label.size, label.color, label.angle, label.position, 
theta, grid, grid.color, grid.radius.color, grid.linetype, grid.size, 
grid.radius.linetype, grid.radius.size, axis, axis.label, axis.color, 
axis.size, axis.linetype, axis.angle, axis.label.color, axis.label.size, 
axis.label.displace, axis.label.angle, legend.position, legend.size, 
legend.text.color, plot.margin)
```

Arguments

- `data`  
  data.table with the utilities of a decision model
- `data.label`  
  data label
- `data.fill`  
  data fill color
- `data.color`  
  data color
- `data.linetype`  
  line type for data
- `data.alpha`  
  alpha scale for data
- `data.size`  
  line size for data
- `data.label.color`  
  label color for data
data.label.size  
  label size for data  
group  
  name for the column of groups  
criteria  
  column name for criteria  
valor  
  column name for utilities  
title  
  plot title  
title.color  
  plot title color  
title.size  
  plot title size  
label.size  
  labels size  
label.color  
  labels color  
label.angle  
  labels angle  
label.position  
  labels position  
theta  
  plot rotation angle  
grid  
  grid for plot  
grid.color  
  grid color  
grid.radius.color  
  grid radius color  
grid.linetype  
  grid line type  
grid.size  
  grid line size  
grid.radius.linetype  
  grid radius line type  
grid.radius.size  
  grid radius line size  
axis  
  axis  
axis.label  
  axis label  
axis.color  
  axis color  
axis.size  
  axis size  
axis.linetype  
  axis line type  
axis.angle  
  axis angle  
axis.label.color  
  axis label color  
axis.label.size  
  axis label size  
axis.label.displace  
  axis label displacement  
axis.label.angle  
  axis label angle  
legend.position  
  label position  
legend.size  
  legend size  
legend.text.color  
  legend text color  
plot.margin  
  plot margin
Value

ggplot2 object with the spider plot

Author(s)

Pedro Guarderas, Andrés Lopez <pedro.felipe.guarderas@gmail.com>

Examples

# Preparing data
library( data.table )
library( ggplot2 )
n<-10
m<-7
cols<-sample( colors()[ grepl('red|blue|olive|darkgreen', colors() ) ], m, replace = TRUE )
data<-data.frame( grp = paste( 'A', sort( rep( 1:m, n ) ), sep = '' ),
cri = factor( rep( paste( 'c', 1:n, sep = '' ), m ),
levels = paste( 'c', 1:n, sep = '' ), ordered = TRUE ),
val = runif( m * n ) )
data.label<-paste( 'A', 1:m, ' class', sep = '' )
data.fill<-cols
data.color<-cols
data.linetype<-rep( 'solid', m )
data.alpha<-rep( 0.05, m )
data.size<-rep( 0.7, m )
data.label.color<-'black'
data.label.size<-15

# Spider plot parameters
title<-'Spider'
title.color<-'red3'
title.size<-20

label.size<-rep( 8, n )
label.color<-rep( 'steelblue4', n )
label.angle<-rep( 0, n )
label.position<-rep( 1.1, n )
theta<-pi/2

grid<-sort( c( 0.1, 0.25, 0.5, 0.75, 1.0 ) )
grid.color<-'grey'
grid.radius.color<-'dodgerblue3'
grid.linetype<-'dashed'
grid.size<-0.5
grid.radius.linetype<-'solid'
grid.radius.size<-0.5

axis<-grid # Same as grid
axis.label<-paste( 100 * axis, '%', sep = '' )
axis.color<-'black'
axis.size<-.7
axis.linetype<-'solid'
axis.angles<-.4*pi
axis.label.color<-'darkgreen'
axis.label.size<-.5
axis.label.displace<-.07
axis.label.angle<-.0
legend.position<-c(0.9, 0.9)
legend.size<-.5
legend.text.color<-'black'
plot.margin<-unit(c(1.0, 1.0, 1.0, 1.0),"cm")

p<-Spider.Plot(data,
data.label,
data.fill,
data.color,
data.linetype,
data.alpha,
data.size,
data.label.color,
data.label.size,
grp,
cri,
val,
title,
title.color,
title.size,
label.size,
label.color,
label.angle,
label.position,
theta,
grid,
grid.color,
grid.radius.color,
grid.linetype,
grid.size,
grid.radius.linetype,
grid.radius.size,
axis,
axis.label,
axis.color,
axis.size,
axis.linetype,
axis.angle,
axis.label.color,
axis.label.size,
axis.label.displace,
axis.label.angle,
Stand.String

Stand.String

Stand.String

Standardize strings

Description

Function to correct and standardize names, designed to eliminate special characters, spaces and other characters.

Usage

Stand.String(x, chr = NULL, rep = NULL)

Arguments

x text to be formatted
chr character vector of replace characters
rep character vector of replacement characters

Value

Returns data table with definition of utility functions by range

Author(s)

Julio Andrade, Pedro Guarderas, Andrés Lopez <pedro.felipe.guarderas@gmail.com>

Examples

x<-c("H?\u00da\u00e0n with C@1_ad1", "M\u00a1a/¬\u00b0r&\u00eca _the#-rot", "ju%LI\u00d6 a P\u00e9rs", "(S)tev
\u00e9n	 los cat%$")
y<-sapply( x, FUN = Stand.String )
names( y )<-NULL
**Sum.Weights**

| Sum.Weights | Sum weights for internal nodes |

**Description**

The weights of the internal nodes has to be computed first is necessary to add each weights of the leaves.

**Usage**

`Sum.Weights(tree)`

**Arguments**

- tree: igraph object representing the tree

**Value**

igraph object updated

**Author(s)**

Pedro Guarderas, Andrés Lopez

**See Also**

`Read.Tree`
Index

Bar.Plot, 4
Compute.Model, 4, 4
Deep.Compute, 5
Divide.Weights, 6
Eval.Utilities, 5, 6, 12, 14
Index.Weights, 8
Make.Decision.Tree, 5, 8, 10
mau (mau-package), 2
mau-package, 2
Plot.Simulation.Weight, 9
Read.Tree, 5, 6, 8, 9, 10, 19
Read.Utilities, 5, 7, 10, 11
Sim(Const.Weights, 5, 9, 10, 12
Sim.Weights, 9, 10, 13
Spider.Plot, 14
Stand.String, 5, 7, 11, 18
Sum.Weights, 19