Package 'mau'

May 8, 2026

Type Package

Version 0.4.0

Title Decision Models with Multi Attribute Utility Theory

Encoding UTF-8

Date 2026-02-11

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.
Also includes other rating analysis methods as for example the Colley, Offensive - Defensive ratings and the ranking aggregation with Borda count.

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 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.1.0)

VignetteBuilder knitr

RdMacros Rdpack

Config/testthat/edition 3

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 2026-02-13 09:40:13 UTC

Contents

`mau-package` *mau*

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 eaf 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:

$$\sum_{i \in C}^{n} w_i u_i(x_i)$$

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 **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

Bell D., Raiffa H., Tversky A. (1988). *Decision Making: Descriptive, normative and prescriptive interactions*. Cambridge University Press.

Clement R. (1991). *Marking Hard Decision: An introduction to decision analysis*. PWS-Kent Publishing Co.

Ward E. (1992). *Utility Theories: Measurements and Applications*. Kluwer Academic Publishers.

Barron FH, Barrett BE (1996). "Decision Quality Using Ranked Attribute Weights." *Manage. Sci.*, **42**(11), 1515–1523. ISSN 0025-1909, doi:10.1287/mnsc.42.11.1515.

Bodily SE (1992). "Introduction: The Practice of Decision and Risk Analysis." *Interfaces*, **22**(6), 1-4. doi:10.1287/inte.22.6.1.

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 a 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

```
library( mau )
vignette( topic = 'Running_MAUT', package = 'mau' )
```

borda_count *Borda count*

Description

Rank aggregation with the Borda count method

Usage

```
borda_count(R, v = NULL)
```

Arguments

R	matrix with rankings
v	vector of votes for each ranking

Value

Vector with aggregated ranking

Author(s)

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

Examples

```
m <- 10
n <- 5
R <- matrix( runif( m * n ), m, n )
v <- sample( 50:100, n )
r <- borda_count( R, v )
```

colley_rating *Colley method*

Description

Rank computation using the Cooley method with ties if required

Usage

```
colley_rating(n, w, l, t = NULL)
```

Arguments

n	symmetric matrix of number of times each player faced another, zero diagonal
w	accumulated vector of wins for each player
l	accumulated vector of losses for each player
t	symmetric matrix of ties, with zero diagonal

Value

Vector with ratings

Author(s)

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

Examples

```
d <- 10
n <- matrix( sample( x = 0:5, size = d * d, replace = TRUE ), d, d )
n <- n + t( n )
diag( n ) <- 0
g <- rowSums( n )
# Number of win matches for each team
w <- sapply( 1:d, FUN = function( i ) sample( x = 1:g[i], size = 1, replace = TRUE ) )
# Number of lost matches for ech team
l <- rowSums( n ) - w
r <- colley_rating( n, w, l )
```

---

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)
```

Arguments

tree	initial tree structure with utilities in its leafs.
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 a any level and represented in a table format.

Value

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

Author(s)

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

See Also

stand_string, read_utilities, eval_utilities, read_tree, make_decision_tree, sim_const_weights.

Examples

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

---

deep_compute *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

```
deep_compute(tree)
```

Arguments

tree	igraph object representing the tree

Value

igraph object updated

Author(s)

Pedro Guarderas, Andrés Lopez

See Also

read_tree

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

```
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

```
eval_utilities(index, columns, functions)
```

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

```
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.

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

```
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 )
```

---

od_rating *Offensive - Defensive rating method*

Description

Computes rating using the offensive-defensive method

Usage

```
od_rating(A, iter = 1000, rer = 1e-12)
```

Arguments

A	matrix with scores
iter	number of iterations
rer	relative error to stop computation

Value

list with rating, offensive score, defensive score and number of iterations.

Author(s)

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

Examples

```
A <- matrix( c( 0, 7, 21, 7, 0,
52, 0, 34, 25, 27,
24, 16, 0, 7, 3,
38, 17, 5, 0, 14,
45, 7, 30, 52, 0 ), nrow = 5, ncol = 5 )
r <- od_rating( A )
```

`plot_sim_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

```
plot_sim_weight(
  S,
  title = "Simulations",
  xlab = "ID",
  ylab = "Utility",
  lines.cols = "blue",
  box.col = "gold",
  box.outlier.col = "darkred",
  utility.col = "darkgreen",
  utility.point.col = "darkgreen",
  text.col = "black"
)
```

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.
`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

---

read_tree *Evaluate utilities*

Description

Read a csv file where the decision tree is defined.

Usage

```
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

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

`read_utilities` *Read utilities*

Description

Builds utility functions from definition standard.

Usage

```
read_utilities(file, lines, skip = 2, encoding = "utf-8", envir = .GlobalEnv)
```

Arguments

`file`	standardize file with definitions.
`lines`	number lines to read in `file`.
`skip`	to read the `file` it had to `skip` a given number of lines.
`encoding`	file encoding.
`envir`	environment where utility functions will be created.

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

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

Examples

```
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_sim_weight( S$simulation, title = 'Simulations',
                           xlab = 'ID', ylab = 'Utility' )
plot( plot.S )
```

`sim_weights` *Simulation of weights*

Description

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

Usage

```
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

```
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 )
```

---

spider_plot *Spider plot*

Description

Generates an spider plot for a decision model

Usage

```
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.font,
  title.color,
  title.size,
  label.font,
  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.font	font type for title
title.color	plot title color
title.size	plot title size
label.font	font type for labels
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 angel
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 )
library( mau )
n <- 27
m <- 4
cols <- sample( colors()[ grepl('(purple|blue|olive)', colors() ) ], m, replace = TRUE )

axis <- seq( 0.1, 1, 0.1 )
dat <- data.table( grp = paste( 'A', sort( rep( 1:m, n ) ), sep = '' ),
                   val = qlnorm( runif( m * n ) * plnorm( 1, 3, 4 ), 3, 4 ) )

dat <- dat[ order( grp, val ) ]
dat[ , cri := factor( rep( paste( 'c', n:1, sep = '' ), m ),
             levels = paste( 'c', n:1, sep = '' ), ordered = TRUE ) ]
dat <- as.data.frame( dat )

parameters <- list( data = dat,
                    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,

                    group = as.name( 'grp' ),
                    criteria = as.name( 'cri' ),
                    valor = as.name( 'val' ),

                    # Spider plot parameters
                    title = 'Spider',
                    title.font = 'New Times Roman',
                    title.color = 'red3',
                    title.size = 20,

                    label.font = 'New Times Roman',
                    label.size = rep( 3, n ),
                    label.color = rep( 'steelblue4', n ),
                    label.angle = rep( 0, n ),
                    label.position = rep( 1.05, n ),

                    theta = pi/3,

                    grid = seq( 0.1, 1, 0.1 ),
                    grid.color = 'grey75',
                    grid.radius.color = 'grey75',
                    grid.linetype = 'dashed',
                    grid.size = 0.5,
                    grid.radius.linetype = 'solid',
```

```
                        grid.radius.size = 0.5,

                        axis = axis, # Same as grid
                        axis.label = paste( 100 * axis, '%', sep = '' ),
                        axis.color = 'black',
                        axis.size = 0.7,
                        axis.linetype = 'solid',
                        axis.angle = 0.4*pi,
                        axis.label.color = 'darkgreen',
                        axis.label.size = 3,
                        axis.label.displace = -0.07,
                        axis.label.angle = 0,

                        legend.position = c(0.9, 0.9),
                        legend.size = 0.5,
                        legend.text.color = 'black',

                        plot.margin = unit( c( 1.0, 1.0, 1.0, 1.0 ),"cm") )


    p <- do.call( spider_plot, parameters )

    plot(p)
```

`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/\u00ac\u00b0r&\u00eca *_the#-rot",
      "ju%LI\u00d6 a P\u00e9rs",
      "(S)tev\n\u00e9n\t los cat%$" )
y <- sapply( x, FUN = stand_string )
names( y ) <- NULL
```

`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