Package: utility (via r-universe)

October 18, 2026

Type Package

Title Construct, Evaluate and Plot Value and Utility Functions

Version 1.4.7

Date 2025-11-20

Description Construct and plot objective hierarchies and associated value and utility functions. Evaluate the values and utilities and visualize the results as colored objective hierarchies or tables. Visualize uncertainty by plotting median and quantile intervals within the nodes of objective hierarchies. Get numerical results of the evaluations in standard R data types for further processing.

License GPL-3

NeedsCompilation no

Author Peter Reichert [cre, aut], Nele Schuwirth [aut]

Maintainer Peter Reichert <peter.reichert@emeriti.eawag.ch>

Repository https://peterreichert.r-universe.dev

Date/Publication 2025-11-21 06:48:37 UTC

RemoteUrl https://github.com/cran/utility

RemoteRef HEAD

RemoteSha 3f3f7ce8f127361172a9d36b5e498c8ae998a970

Contents

utility-package 3
evaluate 11
evaluate.utility.aggregation 13
evaluate.utility.conversion.intpol 14
evaluate.utility.conversion.parfun 16
evaluate.utility.endnode.classcounts 17
evaluate.utility.endnode.cond 19
evaluate.utility.endnode.discrete 20

evaluate.utility.endnode.firstavail . . . 22
evaluate.utility.endnode.intpol1d . . . 23
evaluate.utility.endnode.intpol2d . . . 25
evaluate.utility.endnode.parfun1d . . . 26
plot.utility.aggregation . . . 28
plot.utility.conversion.intpol . . . 31
plot.utility.conversion.parfun . . . 34
plot.utility.endnode.classcounts . . . 37
plot.utility.endnode.cond . . . 39
plot.utility.endnode.discrete . . . 41
plot.utility.endnode.firstavail . . . 42
plot.utility.endnode.intpol1d . . . 44
plot.utility.endnode.intpol2d . . . 46
plot.utility.endnode.parfun1d . . . 48
print.utility.aggregation . . . 49
print.utility.conversion.intpol . . . 50
print.utility.conversion.parfun . . . 51
print.utility.endnode.classcounts . . . 52
print.utility.endnode.cond . . . 53
print.utility.endnode.discrete . . . 54
print.utility.endnode.firstavail . . . 55
print.utility.endnode.intpol1d . . . 56
print.utility.endnode.intpol2d . . . 57
print.utility.endnode.parfun1d . . . 58
summary.utility.aggregation . . . 59
summary.utility.conversion.intpol . . . 60
summary.utility.conversion.parfun . . . 61
summary.utility.endnode.classcounts . . . 62
summary.utility.endnode.cond . . . 63
summary.utility.endnode.discrete . . . 64
summary.utility.endnode.firstavail . . . 65
summary.utility.endnode.intpol1d . . . 66
summary.utility.endnode.intpol2d . . . 67
summary.utility.endnode.parfun1d . . . 68
updatepar . . . 69
updatepar.utility.aggregation . . . 71
updatepar.utility.conversion.intpol . . . 72
updatepar.utility.conversion.parfun . . . 73
updatepar.utility.endnode.classcounts . . . 74
updatepar.utility.endnode.cond . . . 75
updatepar.utility.endnode.discrete . . . 76
updatepar.utility.endnode.firstavail . . . 78
updatepar.utility.endnode.intpol1d . . . 79
updatepar.utility.endnode.intpol2d . . . 80
updatepar.utility.endnode.parfun1d . . . 81
utility.aggregate.add . . . 82
utility.aggregate.addmin . . . 85
utility.aggregate.addpower . . . 88

utility.aggregate.addsplitpower . . . 91
utility.aggregate.bonusmalus . . . 94
utility.aggregate.cobbdouglas . . . 97
utility.aggregate.geo . . . 100
utility.aggregate.geooff . . . 103
utility.aggregate.harmo . . . 106
utility.aggregate.harmooff . . . 109
utility.aggregate.max . . . 112
utility.aggregate.min . . . 114
utility.aggregate.mix . . . 116
utility.aggregate.mult . . . 118
utility.aggregate.revaddpower . . . 122
utility.aggregate.revaddsplitpower . . . 125
utility.aggregate.revgeo . . . 128
utility.aggregate.revgeooff . . . 131
utility.aggregate.revharmo . . . 134
utility.aggregate.revharmooff . . . 137
utility.aggregation.create . . . 140
utility.calc.colors . . . 147
utility.conversion.intpol.create . . . 148
utility.conversion.parfun.create . . . 150
utility.endnode.classcounts.create . . . 152
utility.endnode.cond.create . . . 154
utility.endnode.discrete.create . . . 156
utility.endnode.firstavail.create . . . 159
utility.endnode.intpol1d.create . . . 160
utility.endnode.intpol2d.create . . . 163
utility.endnode.parfun1d.create . . . 165
utility.fun.exp . . . 167
utility.get.attrib.names . . . 169
utility.get.colors . . . 170
utility.structure . . . 171

Index **172**

`utility-package` *Construct, Evaluate and Plot Value and Utility Functions*

Description

Construct and plot objective hierarchies and associated value and utility functions. Evaluate the values and utilities and visualize the results as colored objective hierarchies or tables. Visualize uncertainty by plotting median and quantile intervals within the nodes of the objective hierarchy. Get numerical results of the evaluations in standard R data types for further processing.

Details

Package:	utility
Type:	Package
Version:	1.4.7
Date:	2025-11-20
License:	GPL-3

An objectives hierarchy and an associated value or utility function is constructed by constructing the nodes of the hierarchy starting from the end nodes and proceeding to the higher hierarchies. Five types of end nodes are distinguished: End nodes of the class `utility.endnode.discrete` define a value or utility function for an attribute that has a finite number of discrete numeric or non-numeric levels. End nodes of the classes `utility.endnode.intpol1d` and `utility.endnode.parfun1d` implement single-attribute value or utility functions that accept a continuous argument. The first of these functions allows the user to specify attribute-value pairs and performs linear interpolation between these points. The second function allows the user to specify any parameteric function that is implemented as a function in R. End nodes of the class `utility.endnode.intpol2d` implement interpolated value or utility functions that are based on two attributes. End nodes of the class `utility.endnode.cond` implement value or utility functions that assign different value or utility functions to a finite set of attribute combinations. End nodes of the class `utility.endnode.firstavail` implement value or utility functions that try to evaluate a list of nodes and return the value of the first node that could be evaluated based on the provided attribute data. Finally, end nodes of the class `utility.endnode.classcounts` implement value or utility functions that value counts e.g. of species of different classes by assigning a basic value for the occurrence of at least one species of the best class and incrementing this value by multiplicities of species of this class and of the next lower class. These end nodes can be implemented by using the following constructors.

`utility.endnode.discrete.create`
`utility.endnode.intpol1d.create`
`utility.endnode.parfun1d.create`
`utility.endnode.intpol2d.create`
`utility.endnode.cond.create`
`utility.endnode.firstavail.create`
`utility.endnode.classcounts.create`

To advance to higher hierarchical levels, values or utilities at lower levels must be aggregated to the next higher level. This is done ab aggregation nodes of the class `utility.aggregation`. Such nodes can be implemented by using the following constructor:

`utility.aggregation.create`

Finally, to provide decision support under uncertainty, values at an adequate level of the objectives hierarchy must be converted to utilities by accounting for the risk attitude of the decision maker. Similar to the single-attribute value or utility functions, this can either be done by linear interpolation with a node of the class `utility.conversion.intpol` or by using a parametric funciton in a node of the class `utility.conversion.parfun`. These conversion nodes can be implemented by the constructors:

`utility.conversion.intpol.create`

utility.conversion.parfun.create

The definition of the objective hierarchy and the associated value and utility function can then be listed or visualized by using the generic functions

```
print
summary
plot
```

which automaticall call the implementation corresponding to the node specified as the first argument:

```
print.utility.endnode.discrete
print.utility.endnode.intpol1d
print.utility.endnode.parfun1d
print.utility.endnode.intpol2d
print.utility.endnode.cond
print.utility.endnode.firstavail
print.utility.endnode.classcounts
print.utility.aggregation
print.utility.conversion.intpol
print.utility.conversion.parfun

summary.utility.endnode.discrete
summary.utility.endnode.intpol1d
summary.utility.endnode.parfun1d
summary.utility.endnode.intpol2d
summary.utility.endnode.cond
summary.utility.endnode.firstavail
summary.utility.endnode.classcounts
summary.utility.aggregation
summary.utility.conversion.intpol
summary.utility.conversion.parfun

plot.utility.endnode.discrete
plot.utility.endnode.intpol1d
plot.utility.endnode.parfun1d
plot.utility.endnode.intpol2d
plot.utility.endnode.cond
plot.utility.endnode.firstavail
plot.utility.endnode.classcounts
plot.utility.aggregation
plot.utility.conversion.intpol
plot.utility.conversion.parfun
```

The value or utility function can then be evaluated by applying the generic function

```
evaluate
```

that again calls automatically the corresponding class-specific function

```
evaluate.utility.endnode.discrete
evaluate.utility.endnode.intpol1d
evaluate.utility.endnode.parfun1d
evaluate.utility.endnode.intpol2d
evaluate.utility.endnode.cond
evaluate.utility.endnode.firstavail
evaluate.utility.endnode.classcounts
evaluate.utility.aggregation
evaluate.utility.conversion.intpol
evaluate.utility.conversion.parfun
```

This function requires the provision of observed or predicted attributes of the valued system and returns the corresponding values or utilities of all nodes of the hierarchy. These results can then be visualized by providing them to the generic function

```
plot
```

in addition to the definition of the objective hierarchy stored in the variable corresponding to the highest node of the hierarchy. Again, this function automatically calls the correct class-specific implementation (the root of the hierarchy will be an aggregation or a conversion node, not an end node):

```
plot.utility.aggregation
plot.utility.conversion.intpol
plot.utility.conversion.parfun
```

This proceedure guarantees easy handling with the simple commands `print`, `summary`, `evaluate`, and `plot` and the specific function descriptions provided above are only required to check advanced attributes.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch> with contributions by Nele Schuwirth <nele.schuwirth@eawag.ch>

Maintainer: Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

Examples

```
# define discrete end node for width variability
# (attribute "widthvariability_class" with levels "high",
# "moderate" and "none")

widthvar <-
  utility.endnode.discrete.create(
    name.node     = "width variability",
    attrib.levels = data.frame(widthvariability_class=
      c("high","moderate","none")),
    u             = c(1,0.4125,0),
    names.u       = c("u.high","u_moderate","u.none"),
    required      = FALSE,
    utility       = FALSE)

# define 1d interpolation end node for bed modification with
# riprap
# (attribute "bedmodfract_percent" with levels from 0 to 100)

bedmod_riprap <-
  utility.endnode.intpol1d.create(
    name.node   = "bed modification riprap",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,100),
    u           = c(1,0.775,0.5625,0.24),
    required    = FALSE,
    utility     = FALSE)

# define 1d interpolation end node for bed modification with
# other material
# (attribute "bedmodfract_percent" with levels from 0 to 100)

bedmod_other <-
  utility.endnode.intpol1d.create(
    name.node   = "bed modification other",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,100),
    u           = c(1,0.775,0.5625,0),
    required    = FALSE,
    utility     = FALSE)

# define combination end node for bed modification
# (attributes "bedmodtype_class" and "bedmodfract_percent")

bedmod <-
  utility.endnode.cond.create(
    name.node     = "bed modification",
```

```
    attrib.levels = data.frame(bedmodtype_class=
      c("riprap","other")),
    nodes         = list(bedmod_riprap,bedmod_other),
    required      = FALSE,
    utility       = FALSE)

# define 1d interpolation end node for bank modification with
# permeable material
# (attribute "bankmodfract_percent" with levels from 0 to 100)

bankmod_perm <-
  utility.endnode.intpol1d.create(
    name.node   = "bank modification perm",
    name.attrib = "bankmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,60,100),
    u           = c(1,0.8667,0.675,0.4125,0.24),
    required    = FALSE,
    utility     = FALSE)

# define 1d interpolation end node for bank modification with
# impermeable material
# (attribute "bankmodfract_percent" with levels from 0 to 100)

bankmod_imperm <-
  utility.endnode.intpol1d.create(
    name.node   = "bank modification imperm",
    name.attrib = "bankmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,60,100),
    u           = c(1,0.775,0.5625,0.24,0),
    required    = FALSE,
    utility     = FALSE)

# define combination end node for bank modification
# (attributes "bankmodtype_class" and "bankmodfract_percent")

bankmod <-
  utility.endnode.cond.create(
    name.node     = "bank modification",
    attrib.levels = data.frame(bankmodtype_class=
      c("perm","imperm")),
    nodes         = list(bankmod_perm,bankmod_imperm),
    required      = FALSE,
    utility       = FALSE)

# define 2d interpolation end node for riparian zone width
# (attributes "riparianzonewidth_m" and "riparianzonewidth_m")

riparzone_width <-
  utility.endnode.intpol2d.create(
    name.node   = "riparian zone width",
    name.attrib = c("riverbedwidth_m","riparianzonewidth_m"),
```

```
    ranges      = list(c(0,16),c(0,30)),
    isolines    = list(list(x=c(0,16),y=c(0,0)),
                       list(x=c(0,2,10,16),y=c(5,5,15,15)),
                       list(x=c(0,16),y=c(15,15)),
                       list(x=c(0,16),y=c(30,30))),
    u           = c(0.0,0.6,1.0,1.0),
    lead         = 1,
    utility      = FALSE)

# define discrete end node for riparian zone vegetation
# (attriute "riparianzoneveg_class" with levels "natural",
# "seminatural" and "artificial")

riparzone_veg <-
  utility.endnode.discrete.create(
    name.node     = "riparian zone veg.",
    attrib.levels = data.frame(riparianzoneveg_class=
      c("natural","seminatural","artificial")),
    u             = c(1,0.5625,0),
    required      = FALSE,
    utility       = FALSE)

# define aggregation node for riparian zone

riparzone <-
  utility.aggregation.create(
    name.node = "riparian zone",
    nodes     = list(riparzone_width,riparzone_veg),
    name.fun  = "utility.aggregate.cobbdouglas",
    par       = c(1,1),
    required  = FALSE)

# define aggregation node for ecomorphological state

morphol <-
  utility.aggregation.create(
    name.node = "ecomorphology",
    nodes     = list(widthvar,bedmod,bankmod,riparzone),
    name.fun  = "utility.aggregate.mix",
    par       = c(0.25,0.25,0.25,0.25,0,0,1),
    names.par = c("w_widthvar","w_bedmod","w_bankmod","w_riparzone",
                  "w_add","w_min","w_cobbdouglas"),
    required  = TRUE)

# print individual definitions

print(widthvar)
print(bedmod)

# print all definitions

print(morphol)
```

```
# plot objectives hierarchy with attributes

plot(morphol)

# plot individual nodes:

plot(widthvar)
plot(widthvar,par=c(u_moderate=0.2))
plot(bedmod_other)
plot(bankmod)
#plot(riparzone_width)   # too slow for package installation

# plot selected node definitions of a hierarchy

plot(morphol,type="nodes",nodes=c("width variability",
                                  "bed modification other",
                                  "bank modification"))

# evaluate value function for data sets and plot colored hierarchies
# and table

attrib_channelized <- data.frame(widthvariability_class = "none",
                                 bedmodtype_class       = "riprap",
                                 bedmodfract_percent    = 50,
                                 bankmodtype_class      = "imperm",
                                 bankmodfract_percent   = 70,
                                 riverbedwidth_m        = 10,
                                 riparianzonewidth_m    = 5,
                                 riparianzoneveg_class  = "seminatural")
attrib_rehab       <- data.frame(widthvariability_class = "high",
                                 bedmodtype_class       = "riprap",
                                 bedmodfract_percent    = 50,
                                 bankmodtype_class      = "imperm",
                                 bankmodfract_percent   = 20,
                                 riverbedwidth_m        = 15,
                                 riparianzonewidth_m    = 15,
                                 riparianzoneveg_class  = "natural")

res_channelized     <- evaluate(morphol,attrib=attrib_channelized)
res_channelized_add <- evaluate(morphol,attrib=attrib_channelized,
                                par=c(w_add=1,w_min=0,w_cobbdouglas=0))
res_rehab           <- evaluate(morphol,attrib=attrib_rehab)
res_both            <- rbind(res_channelized,res_rehab)
rownames(res_both)  <- c("channelized","rehabilitated")

plot(morphol,u=res_channelized)
plot(morphol,u=res_channelized_add)
plot(morphol,u=res_rehab)
plot(morphol,u=res_rehab,uref=res_channelized)
plot(morphol,u=res_both,type="table",plot.val=FALSE)
plot(morphol,u=res_both,type="table",plot.val=TRUE,print.val=FALSE)
plot(morphol,u=res_both,uref=res_channelized,type="table",plot.val=FALSE)
```

```
# consideration of uncertain attribute levels
# (Higher uncertainty for predicted state after rehabilitation than for
# observed channelized state.
# Note that the normal distributions lead to a small probability of attribute
# levels beyond the range for which the value function is defined. This could
# be corrected for by truncating or choosing another distribution. We keep
# those values to demonstrate that this leads to warnings when evaluating the
# value function for these attribute levels,):

sampsize <- 1000

attrib_channelized_unc <- data.frame(
  widthvariability_class = rep("high",sampsize),
  bedmodtype_class       = rep("riprap",sampsize),
  bedmodfract_percent    = rnorm(sampsize,mean=50,sd=5),
  bankmodtype_class      = rep("imperm",sampsize),
  bankmodfract_percent   = rnorm(sampsize,mean=70,sd=5),
  riverbedwidth_m        = rep(10,sampsize),
  riparianzonewidth_m    = rep(5,sampsize),
  riparianzoneveg_class  = c("seminatural","artificial")[rbinom(sampsize,1,0.5)+1])

attrib_rehab_unc <- data.frame(
  widthvariability_class = c("moderate","high")[rbinom(sampsize,1,0.5)+1],
  bedmodtype_class       = rep("riprap",sampsize),
  bedmodfract_percent    = rnorm(sampsize,mean=50,sd=15),
  bankmodtype_class      = rep("imperm",sampsize),
  bankmodfract_percent   = rnorm(sampsize,mean=20,sd=5),
  riverbedwidth_m        = rnorm(sampsize,mean=10,sd=2),
  riparianzonewidth_m    = rnorm(sampsize,mean=10,sd=2),
  riparianzoneveg_class  = c("natural","seminatural")[rbinom(sampsize,1,0.5)+1])

res_channelized_unc <- evaluate(morphol,attrib=attrib_channelized_unc)
res_rehab_unc       <- evaluate(morphol,attrib=attrib_rehab_unc)

plot(morphol,u=res_channelized_unc)
#plot(morphol,u=res_rehab_unc)
plot(morphol,u=res_rehab_unc,uref=res_channelized_unc)
plot(morphol,u=list(channelized=res_channelized_unc,rehabilitated=res_rehab_unc),
     type="table")
plot(morphol,u=list(channelized=res_channelized_unc,rehabilitated=res_rehab_unc),
     type="table",nodes=c("ecomorphology","riparian zone"))
plot(morphol,u=list(channelized=res_channelized_unc,rehabilitated=res_rehab_unc),
     type="table",levels=2)
plot(morphol,u=list(channelized=res_channelized_unc,rehabilitated=res_rehab_unc),
     uref=res_channelized_unc,
     type="table")
```

---

evaluate *Evaluate Node and Associated Hierarchy*

Description

Generic function to calculate values or utilities at all nodes of a hierarchy for given levels of the attributes.

Usage

```
evaluate(x, ...)
```

Arguments

x	node to be evaluated.
...	attribute levels have to be provided as an additional argument `attrib`; parameter values can optionally be provided as an additional argument `par`.

Value

Data frame with results of values or utilities at all nodes of the hierarchy for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`,
`utility.aggregation.create`,
`utility.conversion.intpol.create`,
`utility.conversion.parfun.create`

to create the nodes to be evaluated.

Examples

```
# see
help(utility)
# for examples.
```

`evaluate.utility.aggregation`
Evaluate Node and Associated Hierarchy

Description

Calculate values or utilities at all nodes of a hierarchy for given levels of the attributes.

Usage

```
## S3 method for class 'utility.aggregation'
evaluate(x, attrib, par = NA, ...)
```

Arguments

`x`	node to be evaluated.
`attrib`	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
`par`	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
`...`	currently no other arguments are implemented or passed further.

Value

Data frame with results of values or utilities at all nodes of the hierarchy for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

`utility.aggregation.create` to create the node,
`print.utility.aggregation` or `summary.utility.aggregation` to print its definition, and
`plot.utility.aggregation` to plot the node

and

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`,
`utility.endnode.firstavail.create`,
`utility.conversion.intpol.create`,
`utility.conversion.parfun.create`

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

`evaluate.utility.conversion.intpol`

Evaluate Node and Associated Hierarchy

Description

Calculate values or utilities at all nodes of a hierarchy for given levels of the attributes.

Usage

```
## S3 method for class 'utility.conversion.intpol'
evaluate(x, attrib, par = NA, ...)
```

Arguments

x	node to be evaluated.
attrib	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
...	currently no other arguments are implemented or passed further.

Value

Data frame with results of values or utilities at all nodes of the hierarchy for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

utility.conversion.intpol.create to create the node,
print.utility.conversion.intpol or summary.utility.conversion.intpol to print its definition, and
plot.utility.conversion.intpol to plot the node

and

utility.endnode.discrete.create,
utility.endnode.intpol1d.create,
utility.endnode.parfun1d.create,

utility.endnode.intpol2d.create,
utility.endnode.cond.create.
utility.endnode.firstavail.create.
utility.aggregation.create,
utility.conversion.parfun.create

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

---

evaluate.utility.conversion.parfun
Evaluate Node and Associated Hierarchy

Description

Calculate values or utilities at all nodes of a hierarchy for given levels of the attributes.

Usage

```
## S3 method for class 'utility.conversion.parfun'
evaluate(x, attrib, par = NA, ...)
```

Arguments

x	node to be evaluated.
attrib	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
...	currently no other arguments are implemented or passed further.

Value

Data frame with results of values or utilities at all nodes of the hierarchy for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

`utility.aggregation.create` to create the node,
`print.utility.aggregation` or `summary.utility.aggregation` to print its definition, and
`plot.utility.aggregation` to plot the node

and

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`,
`utility.endnode.firstavail.create`,
`utility.aggregation.create`,
`utility.conversion.intpol.create`

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

`evaluate.utility.endnode.classcounts`
Evaluate Node and Associated Hierarchy

Description

Calculate values or utilities at the node for given levels of the attributes.

Usage

```
## S3 method for class 'utility.endnode.classcounts'
evaluate(x, attrib, par = NA, ...)
```

Arguments

`x`	node to be evaluated.
`attrib`	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
`par`	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
`...`	currently no other arguments are implemented or passed further.

Value

Numeric vector of results of values or utilities at the node for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

`utility.endnode.classcounts.create` to create the node,
`print.utility.endnode.classcounts` or `summary.utility.endnode.classcounts` to print its definition, and
`plot.utility.endnode.classcounts` to plot the node

and

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,

utility.endnode.cond.create,
utility.endnode.firstavail.create,
utility.aggregation.create,
utility.conversion.intpol.create,
utility.conversion.parfun.create

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

---

evaluate.utility.endnode.cond
Evaluate Node and Associated Hierarchy

Description

Calculate values or utilities at the node for given levels of the attributes.

Usage

```
## S3 method for class 'utility.endnode.cond'
evaluate(x, attrib, par = NA, ...)
```

Arguments

x	node to be evaluated.
attrib	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
...	currently no other arguments are implemented or passed further.

Value

Numeric vector of results of values or utilities at the node for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

`utility.endnode.cond.create` to create the node,
`print.utility.endnode.cond` or `summary.utility.endnode.cond` to print its definition, and
`plot.utility.endnode.cond` to plot the node

and

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.firstavail.create`,
`utility.aggregation.create`,
`utility.conversion.intpol.create`,
`utility.conversion.parfun.create`

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

`evaluate.utility.endnode.discrete`
Evaluate Node

Description

Calculate values or utilities at the node for given levels of the attributes.

Usage

```
## S3 method for class 'utility.endnode.discrete'
evaluate(x, attrib, par = NA, ...)
```

Arguments

`x`	node to be evaluated.
`attrib`	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
`par`	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
`...`	currently no other arguments are implemented or passed further.

Value

Numeric vector of results of values or utilities at the node for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

`utility.aggregation.create` to create the node,
`print.utility.aggregation` or `summary.utility.aggregation` to print its definition, and
`plot.utility.aggregation` to plot the node

and

`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`,
`utility.endnode.firstavail.create`,

utility.aggregation.create,
utility.conversion.intpol.create,
utility.conversion.parfun.create

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

---

evaluate.utility.endnode.firstavail

Evaluate Node and Associated Hierarchy

Description

Calculate values or utilities at the node for given levels of the attributes.

Usage

```
## S3 method for class 'utility.endnode.firstavail'
evaluate(x, attrib, par = NA, ...)
```

Arguments

x	node to be evaluated.
attrib	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
...	currently no other arguments are implemented or passed further.

Value

Numeric vector of results of values or utilities at the node for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

`utility.endnode.firstavail.create` to create the node,
`print.utility.endnode.firstavail` or `summary.utility.endnode.firstavail` to print its definition, and
`plot.utility.endnode.firstavail` to plot the node

and

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`,
`utility.aggregation.create`,
`utility.conversion.intpol.create`,
`utility.conversion.parfun.create`

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

---

evaluate.utility.endnode.intpol1d *Evaluate Node*

Description

Calculate values or utilities at the node for given levels of the attributes.

Usage

```
## S3 method for class 'utility.endnode.intpol1d'
evaluate(x, attrib, par = NA, ...)
```

Arguments

x	node to be evaluated.
attrib	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
...	currently no other arguments are implemented or passed further.

Value

Numeric vector of results of values or utilities at the node for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

utility.aggregation.create to create the node,
print.utility.aggregation or
summary.utility.aggregation to print its definition, and
plot.utility.aggregation to plot the node

and

utility.endnode.discrete.create,
utility.endnode.parfun1d.create,
utility.endnode.intpol2d.create,
utility.endnode.cond.create,

utility.endnode.firstavail.create,
utility.aggregation.create,
utility.conversion.intpol.create,
utility.conversion.parfun.create

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

---

evaluate.utility.endnode.intpol2d
Evaluate Node

Description

Calculate values or utilities at the node for given levels of the attributes.

Usage

```
## S3 method for class 'utility.endnode.intpol2d'
evaluate(x, attrib, par = NA, ...)
```

Arguments

x	node to be evaluated.
attrib	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
...	currently no other arguments are implemented or passed further.

Value

Numeric vector of results of values or utilities at the node for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

`utility.aggregation.create` to create the node,
`print.utility.aggregation` or `summary.utility.aggregation` to print its definition, and
`plot.utility.aggregation` to plot the node

and

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.cond.create`,
`utility.endnode.firstavail.create`,
`utility.aggregation.create`,
`utility.conversion.intpol.create`,
`utility.conversion.parfun.create`

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

`evaluate.utility.endnode.parfun1d`
Evaluate Node

Description

Calculate values or utilities at the node for given levels of the attributes.

Usage

```
## S3 method for class 'utility.endnode.parfun1d'
evaluate(x, attrib, par = NA, ...)
```

Arguments

x	node to be evaluated.
attrib	numeric vector with labelled components providing the levels of a single set of attributes or data frame for which each row provides such a set of attributes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before evaluation.
...	currently no other arguments are implemented or passed further.

Value

Numeric vector of results of values or utilities at the node for all provided sets of attribute levels.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

utility.aggregation.create to create the node,
print.utility.aggregation or summary.utility.aggregation to print its definition, and
plot.utility.aggregation to plot the node

and

utility.endnode.discrete.create,
utility.endnode.intpol1d.create,
utility.endnode.intpol2d.create,
utility.endnode.cond.create,
utility.endnode.firstavail.create,

utility.aggregation.create,
utility.conversion.intpol.create,
utility.conversion.parfun.create

to create other nodes.

Examples

```
# see
help(utility)
# for examples.
```

---

plot.utility.aggregation *Plot Node Definition or Underlying Objectives Hierarchy*

Description

Plot node definition or underlying objective hierarchy.

Usage

```
## S3 method for class 'utility.aggregation'
plot(x,
     u           = NA,
     uref        = NA,
     par         = NA,
     type        = c("hierarchy", "table", "node", "nodes"),
     nodes       = NA,
     col         = utility.calc.colors(),
     gridlines   = c(0.2, 0.4, 0.6, 0.8),
     main        = "",
     cex.main    = 1,
     cex.nodes   = 1,
     cex.attrib  = 1,
     f.reaches   = 0.2,
     f.nodes     = 0.2,
     with.attrib = TRUE,
     levels      = NA,
     plot.val    = TRUE,
     col.val     = "black",
     lwd.val     = 1,
     print.val   = TRUE,
     two.lines   = FALSE,
     ticks       = c(0,0.2,0.4,0.6,0.8,1),
     ...)
```

Arguments

x	node to be plotted.
u	(optional) vector or data frame with elements or columns labelled according to the nodes of the hierarchy containing values or utilities. Typically, this will be the complete output or an output row of the function evaluate.utility.aggregation. This input is only considered if the argument type is specified to be either "hierarchy" or "table". It is then used to color-code the boxes of the hierarchy representing value nodes or the table. If u is a data frame with more than one row and the argument type is equal to "hierarchy", then the median and quantile boxes are plotted for value nodes or the expected utility for utility nodes unless the argument main contains as many elements as the number of rows of u. In the latter case, separate hierarchies with color-coded boxes for value nodes are produced for all rows of u. For type equals "table", this argument can be a list of data frames to make it possible to plot uncertainty ranges from the samples provided in the list.
uref	(optional) vector or data frame with elements or columns labelled according to the nodes of the hierarchy containing values or utilities. Typically, this will be the complete output or an output row of the function evaluate.utility.aggregation. This input is only considered if the argument type is specified to be "hierarchy". It is then used to color-code the upper part of the boxes of the hierarchy to allow for a comparison with the results provided by the argument u which are shown in the lower part of the boxes.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node. Note that this affects only the node definitions plotted if the argument type is specified to be "node" or "nodes". To color-code hierarchies or tables for different parameter values, the parameters have to be passed to evaluate.utility.aggregation before passing the results of this function to this plotting routine.
type	(optional) specifies the type of plot to be produced. Options: "hierarchy", "table", "node" or "nodes". "hierarchy": produces a plot of the objectives hierarchy including color-coded results for values or utilities if these values are provided by the arguments u and/or uref. "table": produces a table with color-coded results for values or utilities if these values are provided by the argument u. "node": produces a plot of the definition of the current node. "nodes": produces plots of node definitions for all nodes defined by the attribute nodes.
nodes	(optional) character vector specifying the nodes for which the definitions will be plotted or which will be considered in a table. The default value of NA indicates that all nodes will be plotted. This argument only affects the output if the argument type was indicated to be either "table" or "nodes".
col	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to

have a non-equidistant color-coding). This attribute is only used for value nodes and if values are provided by the arguments `u` and/or `uref`.

`gridlines` (optional) numeric vector of levels at which gridlines are plotted in node definitions. This attribute is only used if the argument `type` is specified to be either `"node"` or `"nodes"`.

`main` (optional) title(s) of the plot. If the argument `type` is equal to `"hierarchy"` and the a vector of titles with the same length as the number of rows of the argument `u` is provided, a color-coded hierarchy is plotted for each row of `u`. Otherwise, the medians and colored boxes indicating 90% credibility or occurrence ranges are plotted at all nodes.

`cex.main` (optional) scaling factor for title of the plot.

`cex.nodes` (optional) scaling factor for node labels used in the plot.

`cex.attrib` (optional) scaling factor for attribute labels used in the plot.

`f.reaches` (optional) fraction of the width of the plot reserved for the row labels of the table if the argument `type` is equal to `"table"`.

`f.nodes` (optional) fraction of the height of the plot reserved for the column labels of the table if the argument `type` is equal to `"table"`.

`with.attrib` (optional) indicates if attributes should be listed if the argument `type` is equal to `"hierarchy"`.

`levels` (optional) how many levels of the hierarchy should be plotted (NA means to plot all levels).

`plot.val` (optional) plot value (for hierarchy without uncertainty) or median (for hierarchy with uncertainty) as a vertical line within the box.

`col.val` (optional) color of the vertical line indicating the value or median within the box (default black).

`lwd.val` (optional) line width of the vertical line indicating the value or median within the box (default 1).

`print.val` (optional) print value as a number when plotting a table of boxes.

`two.lines` (optional) choose whether two lines should be used for the labels in the hierarchy plot.

`ticks` (optional) positions of tick marks for hierarchy and table plots (NA or numeric(0) avoids tick marks).

`...` additional arguments passed to the R plotting routine.

Note

Note that the plotting routines
`plot.utility.conversion.intpol`
`plot.utility.conversion.parfun`
are exactly the same so that all hierarchies can be plotted with exactly the same commands irrespective of the type of the top-level node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.aggregation.create` for how to construct such a node and `evaluate.utility.aggregation` for how to evaluate the node.

See `utility.calc.colors` for an example of how to construct color schemes and `utility.get.colors` for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

---

plot.utility.conversion.intpol

Plot Node Definition or Underlying Objectives Hierarchy

Description

Plot node definition or underlying objective hierarchy.

Usage

```
## S3 method for class 'utility.conversion.intpol'
plot(x,
     u           = NA,
     uref        = NA,
     par         = NA,
     type        = c("hierarchy", "table", "node", "nodes"),
     nodes       = NA,
     col         = utility.calc.colors(),
     gridlines   = c(0.2, 0.4, 0.6, 0.8),
     main        = "",
```

```
cex.main    = 1,
cex.nodes   = 1,
cex.attrib  = 1,
f.reaches   = 0.2,
f.nodes     = 0.2,
with.attrib = TRUE,
levels      = NA,
plot.val    = TRUE,
print.val   = TRUE,
two.lines   = FALSE,
...)
```

Arguments

`x` node to be plotted.

`u` (optional) vector or data frame with elements or columns labelled according to the nodes of the hierarchy containing values or utilities. Typically, this will be the complete output or an output row of the function `evaluate.utility.aggregation`.
This input is only considered if the argument `type` is specified to be either `"hierarchy"` or `"table"`. It is then used to color-code the boxes of the hierarchy representing value nodes or the table. If `u` is a data frame with more than one row and the argument `type` is equal to `"hierarchy"`, then the median and quantile boxes are plotted for value nodes or the expected utility for utility nodes unless the argument `main` contains as many elements as the number of rows of `u`. In the latter case, separate hierarchies with color-coded boxes for value nodes are produced for all rows of `u`. For `type` equals `"table"`, this argument can be a list of data frames to make it possible to plot uncertainty ranges from the samples provided in the list.

`uref` (optional) vector or data frame with elements or columns labelled according to the nodes of the hierarchy containing values or utilities. Typically, this will be the complete output or an output row of the function `evaluate.utility.aggregation`.
This input is only considered if the argument `type` is specified to be `"hierarchy"`. It is then used to color-code the upper part of the boxes of the hierarchy to allow for a comparison with the results provided by the argument `u` which are shown in the lower part of the boxes.

`par` (optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node. Note that this affects only the node definitions plotted if the argument `type` is specified to be `"node"` or `"nodes"`. To color-code hierarchies or tables for different parameter values, the parameters have to be passed to `evaluate.utility.aggregation` before passing the results of this function to this plotting routine.

`type` (optional) specifies the type of plot to be produced.
Options: `"hierarchy"`, `"table"`, `"node"` or `"nodes"`.
`"hierarchy"`: produces a plot of the objectives hierarchy including color-coded results for values or utilities if these values are provided by the arguments `u`

and/or uref.
"table": produces a table with color-coded results for values or utilities if these values are provided by the argument u.
"node": produces a plot of the definition of the current node.
"nodes": produces plots of node definitions for all nodes defined by the attribute nodes.

nodes — (optional) character vector specifying the nodes for which the definitions will be plotted or which will be considered in a table. The default value of NA indicates that all nodes will be plotted. This argument only affects the output if the argument type was indicated to be either "table" or "nodes".

col — (optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes and if values are provided by the arguments u and/or uref.

gridlines — (optional) numeric vector of levels at which gridlines are plotted in node definitions. This attribute is only used if the argument type is specified to be either "node" or "nodes".

main — (optional) title(s) of the plot. If the argument type is equal to "hierarchy" and the a vector of titles with the same length as the number of rows of the argument u is provided, a color-coded hierarchy is plotted for each row of u. Otherwise, the medians and colored boxes indicating 90% credibility or occurrence ranges are plotted at all nodes.

cex.main — (optional) scaling factor for title of the plot.

cex.nodes — (optional) scaling factor for node labels used in the plot.

cex.attrib — (optional) scaling factor for attribute labels used in the plot.

f.reaches — (optional) fraction of the width of the plot reserved for the row labels of the table if the argument type is equal to "table".

f.nodes — (optional) fraction of the height of the plot reserved for the column labels of the table if the argument type is equal to "table".

with.attrib — (optional) indicates if attributes should be listed if the argument type is equal to "hierarchy".

levels — (optional) how many levels of the hierarchy should be plotted (NA means to plot all levels).

plot.val — (optional) plot value as a vertical line within the box.

print.val — (optional) print value as a number when plotting a table of boxes.

two.lines — (optional) choose whether two lines should be used for the labels in the hierarchy plot.

... — additional arguments passed to the R plotting routine.

Note

Note that the plotting routines
plot.utility.conversion.parfun
plot.utility.aggregation
are exactly the same so that all hierarchies can be plotted with exactly the same commands irrespective of the type of the top-level node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.conversion.intpol.create` for how to construct such a node and `evaluate.utility.conversion.intpol` for how to evaluate the node.

See `utility.calc.colors` for an example of how to construct color schemes and `utility.get.colors` for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

---

plot.utility.conversion.parfun

Plot Node Definition or Underlying Objectives Hierarchy

Description

Plot node definition or underlying objectives hierarchy.

Usage

```
## S3 method for class 'utility.conversion.parfun'
plot(x,
     u           = NA,
     uref        = NA,
     par         = NA,
     type        = c("hierarchy", "table", "node", "nodes"),
```

```
nodes       = NA,
col         = utility.calc.colors(),
gridlines   = c(0.2, 0.4, 0.6, 0.8),
main        = "",
cex.main    = 1,
cex.nodes   = 1,
cex.attrib  = 1,
f.reaches   = 0.2,
f.nodes     = 0.2,
with.attrib = TRUE,
levels      = NA,
plot.val    = TRUE,
print.val   = TRUE,
two.lines   = FALSE,
...)
```

Arguments

`x` node to be plotted.

`u` (optional) vector or data frame with elements or columns labelled according to the nodes of the hierarchy containing values or utilities. Typically, this will be the complete output or an output row of the function `evaluate.utility.aggregation`.
This input is only considered if the argument `type` is specified to be either `"hierarchy"` or `"table"`. It is then used to color-code the boxes of the hierarchy representing value nodes or the table. If `u` is a data frame with more than one row and the argument `type` is equal to `"hierarchy"`, then the median and quantile boxes are plotted for value nodes or the expected utility for utility nodes unless the argument `main` contains as many elements as the number of rows of `u`. In the latter case, separate hierarchies with color-coded boxes for value nodes are produced for all rows of `u`. For `type` equals `"table"`, this argument can be a list of data frames to make it possible to plot uncertainty ranges from the samples provided in the list.

`uref` (optional) vector or data frame with elements or columns labelled according to the nodes of the hierarchy containing values or utilities. Typically, this will be the complete output or an output row of the function `evaluate.utility.aggregation`.
This input is only considered if the argument `type` is specified to be `"hierarchy"`. It is then used to color-code the upper part of the boxes of the hierarchy to allow for a comparison with the results provided by the argument `u` which are shown in the lower part of the boxes.

`par` (optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node. Note that this affects only the node definitions plotted if the argument `type` is specified to be `"node"` or `"nodes"`. To color-code hierarchies or tables for different parameter values, the parameters have to be passed to `evaluate.utility.aggregation` before passing the results of this function to this plotting routine.

`type` (optional) specifies the type of plot to be produced.
Options: `"hierarchy"`, `"table"`, `"node"` or `"nodes"`.
`"hierarchy"`: produces a plot of the objectives hierarchy including color-coded results for values or utilities if these values are provided by the arguments `u` and/or `uref`.
`"table"`: produces a table with color-coded results for values or utilities if these values are provided by the argument `u`.
`"node"`: produces a plot of the definition of the current node.
`"nodes"`: produces plots of node definitions for all nodes defined by the attribute `nodes`.

`nodes` (optional) character vector specifying the nodes for which the definitions will be plotted or which will be considered in a table. The default value of NA indicates that all nodes will be plotted. This argument only affects the output if the argument `type` was indicated to be either `"table"` or `"nodes"`.

`col` (optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes and if values are provided by the arguments `u` and/or `uref`.

`gridlines` (optional) numeric vector of levels at which gridlines are plotted in node definitions. This attribute is only used if the argument `type` is specified to be either `"node"` or `"nodes"`.

`main` (optional) title(s) of the plot. If the argument `type` is equal to `"hierarchy"` and the a vector of titles with the same length as the number of rows of the argument `u` is provided, a color-coded hierarchy is plotted for each row of `u`. Otherwise, the medians and colored boxes indicating 90% credibility or occurrence ranges are plotted at all nodes.

`cex.main` (optional) scaling factor for title of the plot.

`cex.nodes` (optional) scaling factor for node labels used in the plot.

`cex.attrib` (optional) scaling factor for attribute labels used in the plot.

`f.reaches` (optional) fraction of the width of the plot reserved for the row labels of the table if the argument `type` is equal to `"table"`.

`f.nodes` (optional) fraction of the height of the plot reserved for the column labels of the table if the argument `type` is equal to `"table"`.

`with.attrib` (optional) indicates if attributes should be listed if the argument `type` is equal to `"hierarchy"`.

`levels` (optional) how many levels of the hierarchy should be plotted (NA means to plot all levels).

`plot.val` (optional) plot value as a vertical line within the box.

`print.val` (optional) print value as a number when plotting a table of boxes.

`two.lines` (optional) choose whether two lines should be used for the labels in the hierarchy plot.

`...` additional arguments passed to the R plotting routine.

Note

Note that the plotting routines
plot.utility.conversion.intpol
plot.utility.aggregation
are exactly the same so that all hierarchies can be plotted with exactly the same commands irrespective of the type of the top-level node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See utility.conversion.parfun.create for how to construct such a node and evaluate.utility.conversion.parfun for how to evaluate the node.

See utility.calc.colors for an example of how to construct color schemes and utility.get.colors for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

---

plot.utility.endnode.classcounts

Plot Node Definition

Description

Plot node definition.

Usage

```
## S3 method for class 'utility.endnode.classcounts'
plot(x,
     par       = NA,
     col       = utility.calc.colors(),
     gridlines = c(0.2, 0.4, 0.6, 0.8),
     main      = "",
     cex.main  = 1,
     ...)
```

Arguments

x	node to be plotted.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node.
col	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes.
gridlines	(optional) numeric vector of levels at which gridlines are plotted in the node definition.
main	(optional) title of the plot.
cex.main	(optional) scaling factor for title of the plot.
...	additional arguments passed to the R plotting routine.

Note

Note that the plotting routines for the other end nodes
plot.utility.endnode.discrete
plot.utility.endnode.intpol1d
plot.utility.endnode.parfun1d
plot.utility.endnode.intpol2d
plot.utility.endnode.cond
plot.utility.endnode.firstavail
are as far as possible the same so that all end nodes can be plotted with the same commands irrespective of the type of the end node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.classcounts.create` for how to construct such a node and `evaluate.utility.endnode.classcou` for how to evaluate the node.

See `utility.calc.colors` for an example of how to construct color schemes and `utility.get.colors` for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

`plot.utility.endnode.cond`

Plot Node Definition

Description

Plot node definition.

Usage

```
## S3 method for class 'utility.endnode.cond'
plot(x,
     par       = NA,
     col       = utility.calc.colors(),
     gridlines = c(0.2, 0.4, 0.6, 0.8),
     main      = "",
     cex.main  = 1,
     nodes     = x$name,
     ...)
```

Arguments

`x`	node to be plotted.
`par`	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node.

col	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes.
gridlines	(optional) numeric vector of levels at which gridlines are plotted in the node definition.
main	(optional) title of the plot.
cex.main	(optional) scaling factor for title of the plot.
nodes	(optional) character vector specifying the names of the nodes to be plotted.
...	additional arguments passed to the R plotting routine.

Note

Note that the plotting routines for the other end nodes
plot.utility.endnode.discrete
plot.utility.endnode.intpol1d
plot.utility.endnode.parfun1d
plot.utility.endnode.intpol2d
plot.utility.endnode.firstavail
are as far as possible the same so that all end nodes can be plotted with the same commands irrespective of the type of the end node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See utility.endnode.cond.create for how to construct such a node and evaluate.utility.endnode.cond for how to evaluate the node.

See utility.calc.colors for an example of how to construct color schemes and utility.get.colors for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

---

plot.utility.endnode.discrete

Plot Node Definition

Description

Plot node definition.

Usage

```
## S3 method for class 'utility.endnode.discrete'
plot(x,
     par       = NA,
     col       = utility.calc.colors(),
     gridlines = c(0.2, 0.4, 0.6, 0.8),
     main      = "",
     cex.main  = 1,
     ...)
```

Arguments

x	node to be plotted.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node.
col	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes.
gridlines	(optional) numeric vector of levels at which gridlines are plotted in the node definition.
main	(optional) title of the plot.
cex.main	(optional) scaling factor for title of the plot.
...	additional arguments passed to the R plotting routine.

Note

Note that the plotting routines for the other end nodes
plot.utility.endnode.intpol1d
plot.utility.endnode.parfun1d
plot.utility.endnode.intpol2d

plot.utility.endnode.cond
plot.utility.endnode.firstavail
are as far as possible the same so that all end nodes can be plotted with the same commands irrespective of the type of the end node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See utility.endnode.discrete.create for how to construct such a node and evaluate.utility.endnode.discrete for how to evaluate the node.

See utility.calc.colors for an example of how to construct color schemes and utility.get.colors for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

---

plot.utility.endnode.firstavail

Plot Node Definition

Description

Plot node definition.

Usage

```
## S3 method for class 'utility.endnode.firstavail'
plot(x,
     par       = NA,
     col       = utility.calc.colors(),
     gridlines = c(0.2, 0.4, 0.6, 0.8),
     main      = "",
     cex.main  = 1,
     nodes     = x$name,
     ...)
```

Arguments

x	node to be plotted.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node.
col	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes.
gridlines	(optional) numeric vector of levels at which gridlines are plotted in the node definition.
main	(optional) title of the plot.
cex.main	(optional) scaling factor for title of the plot.
nodes	(optional) character vector specifying the names of the nodes to be plotted.
...	additional arguments passed to the R plotting routine.

Note

Note that the plotting routines for the other end nodes
plot.utility.endnode.discrete
plot.utility.endnode.intpol1d
plot.utility.endnode.parfun1d
plot.utility.endnode.intpol2d
plot.utility.endnode.cond
are as far as possible the same so that all end nodes can be plotted with the same commands irrespective of the type of the end node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and

utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.firstavail.create` for how to construct such a node and `evaluate.utility.endnode.firstavai` for how to evaluate the node.

See `utility.calc.colors` for an example of how to construct color schemes and `utility.get.colors` for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

`plot.utility.endnode.intpol1d`

Plot Node Definition

Description

Plot node definition.

Usage

```
## S3 method for class 'utility.endnode.intpol1d'
plot(x,
     par       = NA,
     col       = utility.calc.colors(),
     gridlines = c(0.2, 0.4, 0.6, 0.8),
     main      = "",
     cex.main  = 1,
     xlim      = numeric(0),
     ...)
```

Arguments

`x`	node to be plotted.
`par`	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node.

`col`	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes.
`gridlines`	(optional) numeric vector of levels at which gridlines are plotted in the node definition.
`main`	(optional) title of the plot.
`cex.main`	(optional) scaling factor for title of the plot.
`xlim`	(optional) limits for x-axis of the plot (default is range).
`...`	additional arguments passed to the R plotting routine.

Note

Note that the plotting routines for the other end nodes
`plot.utility.endnode.discrete`
`plot.utility.endnode.parfun1d`
`plot.utility.endnode.intpol2d`
`plot.utility.endnode.cond`
`plot.utility.endnode.firstavail`
are as far as possible the same so that all end nodes can be plotted with the same commands irrespective of the type of the end node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.intpol1d.create` for how to construct such a node and `evaluate.utility.endnode.intpol1d` for how to evaluate the node.

See `utility.calc.colors` for an example of how to construct color schemes and `utility.get.colors` for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

`plot.utility.endnode.intpol2d`
Plot Node Definition

Description

Plot node definition.

Usage

```
## S3 method for class 'utility.endnode.intpol2d'
plot(x,
     par       = NA,
     col       = utility.calc.colors(),
     gridlines = c(0.2, 0.4, 0.6, 0.8),
     main      = "",
     cex.main  = 1,
     xlim      = numeric(0),
     ylim      = numeric(0),
     ...)
```

Arguments

`x`	node to be plotted.
`par`	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node.
`col`	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes.
`gridlines`	(optional) numeric vector of levels at which gridlines are plotted in the node definition. Not used for this type of node.
`main`	(optional) title of the plot.
`cex.main`	(optional) scaling factor for title of the plot.
`xlim`	(optional) limits of the x-axis of the plot (defaults to range).
`ylim`	(optional) limits of the y-axis of the plot (defaults to range).
`...`	additional arguments passed to the R plotting routine.

Note

Note that the plotting routines for the other end nodes
`plot.utility.endnode.discrete`
`plot.utility.endnode.parfun1d`
`plot.utility.endnode.intpol2d`
`plot.utility.endnode.cond`
`plot.utility.endnode.firstavail`
are as far as possible the same so that all end nodes can be plotted with the same commands irrespective of the type of the end node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.intpol1d.create` for how to construct such a node and `evaluate.utility.endnode.intpol1d` for how to evaluate the node.

See `utility.calc.colors` for an example of how to construct color schemes and `utility.get.colors` for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

plot.utility.endnode.parfun1d

Plot Node Definition

Description

Plot node definition.

Usage

```
## S3 method for class 'utility.endnode.parfun1d'
plot(x,
     par       = NA,
     col       = utility.calc.colors(),
     gridlines = c(0.2, 0.4, 0.6, 0.8),
     main      = "",
     cex.main  = 1,
     xlim      = numeric(0),
     ...)
```

Arguments

x	node to be plotted.
par	(optional) labelled numeric parameter vector providing parameters to modify the value or utility function before plotting the node.
col	(optional) character vector of colors to be used to color the interval between zero and unity in equidistant sections (use repetitions of the same color if you want to have a non-equidistant color-coding). This attribute is only used for value nodes.
gridlines	(optional) numeric vector of levels at which gridlines are plotted in the node definition.
main	(optional) title of the plot.
cex.main	(optional) scaling factor for title of the plot.
xlim	(optional) limits for x-axis of the plot (default is range).
...	additional arguments passed to the R plotting routine.

Note

Note that the plotting routines for the other end nodes
plot.utility.endnode.discrete
plot.utility.endnode.intpol1d
plot.utility.endnode.intpol2d
plot.utility.endnode.cond
plot.utility.endnode.firstavail
are as far as possible the same so that all end nodes can be plotted with the same commands irrespective of the type of the end node.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node and `evaluate.utility.endnode.parfun1d` for how to evaluate the node.

See `utility.calc.colors` for an example of how to construct color schemes and `utility.get.colors` for how to get colors for specifed value levels.

Examples

```
# see
help(utility)
# for examples.
```

---

print.utility.aggregation

Print Definitions of Node and Associated Hierarchy

Description

Print definition of node and associated hierarchy.

Usage

```
## S3 method for class 'utility.aggregation'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.aggregation.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

print.utility.conversion.intpol

Print Definitions of Node and Associated Hierarchy

Description

Print definition of node and associated hierarchy.

Usage

```
## S3 method for class 'utility.conversion.intpol'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.conversion.intpol.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

print.utility.conversion.parfun

Print Definitions of Node and Associated Hierarchy

Description

Print definition of node and associated hierarchy.

Usage

```
## S3 method for class 'utility.conversion.parfun'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.conversion.parfun.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

`print.utility.endnode.classcounts` *Print Node Definition*

Description

Print node defintion.

Usage

```
## S3 method for class 'utility.endnode.classcounts'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.classcounts.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

`print.utility.endnode.cond` *Print Node Definition*

Description

Print node defintion.

Usage

```
## S3 method for class 'utility.endnode.cond'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.cond.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

print.utility.endnode.discrete

Print Node Definition

Description

Print node defintion.

Usage

```
## S3 method for class 'utility.endnode.discrete'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.discrete.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

`print.utility.endnode.firstavail` *Print Node Definition*

Description

Print node defintion.

Usage

```
## S3 method for class 'utility.endnode.firstavail'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.firstavail.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

`print.utility.endnode.intpol1d` *Print Node Definition*

Description

Print node defintion.

Usage

```
## S3 method for class 'utility.endnode.intpol1d'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.intpol1d.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

`print.utility.endnode.intpol2d` *Print Node Definition*

Description

Print node defintion.

Usage

```
## S3 method for class 'utility.endnode.intpol2d'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.intpol2d.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

print.utility.endnode.parfun1d

Print Node Definition

Description

Print node defintion.

Usage

```
## S3 method for class 'utility.endnode.parfun1d'
print(x, ...)
```

Arguments

`x`	node to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.aggregation

Print Summary of Definitions of Node and Associated Hierarchy

---

Description

Print summary of definition of node and associated hierarchy.

Usage

```
## S3 method for class 'utility.aggregation'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.aggregation.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.conversion.intpol

Print Summary of Definitions of Node and Associated Hierarchy

---

Description

Print summary of definition of node and associated hierarchy.

Usage

```
## S3 method for class 'utility.conversion.intpol'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.conversion.intpol.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.conversion.parfun

Print Summary of Definitions of Node and Associated Hierarchy

---

Description

Print summary of definition of node and associated hierarchy.

Usage

```
## S3 method for class 'utility.conversion.parfun'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.conversion.parfun.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.endnode.classcounts

Print Summary of Node Definition

Description

Print summary of node definition.

Usage

```
## S3 method for class 'utility.endnode.classcounts'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.classcounts.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.endnode.cond

Print Summary of Node Definition

Description

Print summary of node definition.

Usage

```
## S3 method for class 'utility.endnode.cond'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.cond.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.endnode.discrete

Print Summary of Node Definition

Description

Print summary of node definition.

Usage

```
## S3 method for class 'utility.endnode.discrete'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.discrete.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.endnode.firstavail

Print Summary of Node Definition

Description

Print summary of node definition.

Usage

```
## S3 method for class 'utility.endnode.firstavail'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.firstavail.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.endnode.intpol1d

Print Summary of Node Definition

Description

Print summary of node definition.

Usage

```
## S3 method for class 'utility.endnode.intpol1d'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.intpol1d.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.endnode.intpol2d

Print Summary of Node Definition

Description

Print summary of node definition.

Usage

```
## S3 method for class 'utility.endnode.intpol2d'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.intpol2d.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

summary.utility.endnode.parfun1d

Print Summary of Node Definition

Description

Print summary of node definition.

Usage

```
## S3 method for class 'utility.endnode.parfun1d'
summary(object, ...)
```

Arguments

`object`	node of which a summary is to be printed.
`...`	currently no other arguments are implemented or passed further.

Note

In the current version of the package, the methods `print` and `summary` provide the same output.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node.

Examples

```
# see
help(utility)
# for examples.
```

---

updatepar — *Update Parameters in Node Definitions*

Description

Generic function to update parameters in all node defintions of the hierarchy defined by the given node.

Usage

```
updatepar(x, ...)
```

Arguments

`x`	node to be updated.
`...`	parameter values can be provided by an additional argument `par`.

Value

The node or node hierarchy with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`,
`utility.aggregation.create`,
`utility.conversion.intpol.create`,
`utility.conversion.parfun.create`

for how to construct the nodes and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.cond`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
`updatepar.utility.conversion.parfun`
for the updates of the specific nodes.

updatepar.utility.aggregation
Update Parameters in Node Definitions

Description

Update parameters in all node defintions of the hierarchy defined by the node.

Usage

```
## S3 method for class 'utility.aggregation'
updatepar(x, par=NA, ...)
```

Arguments

x	node to be updated.
par	parameter vector with labelled parameters to be updated.
...	currently no other arguments are implemented or passed further.

Value

The node hierarchy with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See utility.aggregation.create for how to construct such a node and

updatepar.utility.endnode.discrete
updatepar.utility.endnode.intpol1d

updatepar.utility.endnode.parfun1d
updatepar.utility.endnode.intpol2d
updatepar.utility.endnode.cond
updatepar.utility.conversion.intpol
updatepar.utility.conversion.parfun
for analogous updates of other nodes

---

updatepar.utility.conversion.intpol
Update Parameters in Node Definitions

Description

Update parameters in all node defintions of the hierarchy defined by the node.

Usage

```
## S3 method for class 'utility.conversion.intpol'
updatepar(x, par=NA, ...)
```

Arguments

x	node to be updated.
par	parameter vector labelled with parameter values to be updated.
...	currently no other arguments are implemented or passed further.

Value

The node hierarchy with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.cond`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.parfun`
for analogous updates of other nodes

---

updatepar.utility.conversion.parfun

Update Parameters in Node Definitions

Description

Update parameters in all node defintions of the hierarchy defined by the node.

Usage

```
## S3 method for class 'utility.conversion.parfun'
updatepar(x, par=NA, ...)
```

Arguments

`x`	node to be updated.
`par`	parameter vector with labelled parameters to be updated.
`...`	currently no other arguments are implemented or passed further.

Value

The node hierarchy with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.cond`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
for analogous updates of other nodes

`updatepar.utility.endnode.classcounts`
Update Parameters in Node Definitions

Description

Update parameters in all node defintions used to define the node.

Usage

```
## S3 method for class 'utility.endnode.classcounts'
updatepar(x, par=NA, ...)
```

Arguments

`x`	node to be updated.
`par`	parameter vector with labelled parameters to be updated.
`...`	currently no other arguments are implemented or passed further.

Value

The node with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.classcounts.create` for how to construct such a node and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.cond`
`updatepar.utility.endnode.firstavail`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
`updatepar.utility.conversion.parfun`
for analogous updates of other nodes

---

updatepar.utility.endnode.cond

Update Parameters in Node Definitions

Description

Update parameters in all node defintions used to define the node.

Usage

```
## S3 method for class 'utility.endnode.cond'
updatepar(x, par=NA, ...)
```

Arguments

`x`	node to be updated.
`par`	parameter vector with labelled parameters to be updated.
`...`	currently no other arguments are implemented or passed further.

Value

The node with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.cond.create` for how to construct such a node and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.firstavail`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
`updatepar.utility.conversion.parfun`
for analogous updates of other nodes

`updatepar.utility.endnode.discrete` *Update Parameters in Node Definition*

Description

Update parameters in node defintion.

Usage

```
## S3 method for class 'utility.endnode.discrete'
updatepar(x, par=NA, ...)
```

Arguments

x	node to be updated.
par	parameter vector with labelled parameters to be updated.
...	currently no other arguments are implemented or passed further.

Value

The node with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node and

`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.cond`
`updatepar.utility.endnode.firstavail`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
`updatepar.utility.conversion.parfun`
for analogous updates of other nodes

updatepar.utility.endnode.firstavail

Update Parameters in Node Definitions

Description

Update parameters in all node defintions used to define the node.

Usage

```
## S3 method for class 'utility.endnode.firstavail'
updatepar(x, par=NA, ...)
```

Arguments

x	node to be updated.
par	parameter vector with labelled parameters to be updated.
...	currently no other arguments are implemented or passed further.

Value

The node with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See utility.endnode.firstavail.create for how to construct such a node and

updatepar.utility.endnode.discrete
updatepar.utility.endnode.intpol1d

updatepar.utility.endnode.parfun1d
updatepar.utility.endnode.intpol2d
updatepar.utility.endnode.cond
updatepar.utility.aggregation
updatepar.utility.conversion.intpol
updatepar.utility.conversion.parfun
for analogous updates of other nodes

---

updatepar.utility.endnode.intpol1d
Update Parameters in Node Definition

Description

Update parameters in node defintion.

Usage

```
## S3 method for class 'utility.endnode.intpol1d'
updatepar(x, par=NA, ...)
```

Arguments

x	node to be updated.
par	parameter vector with labelled parameters to be updated.
...	currently no other arguments are implemented or passed further.

Value

The node with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.cond`
`updatepar.utility.endnode.firstavail`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
`updatepar.utility.conversion.parfun`
for analogous updates of other nodes

---

updatepar.utility.endnode.intpol2d

Update Parameters in Node Definition

Description

Update parameters in node defintion.

Usage

```
## S3 method for class 'utility.endnode.intpol2d'
updatepar(x, par=NA, ...)
```

Arguments

`x`	node to be updated.
`par`	parameter vector with labelled parameters to be updated.
`...`	currently no other arguments are implemented or passed further.

Value

The node with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.parfun1d`
`updatepar.utility.endnode.cond`
`updatepar.utility.endnode.firstavail`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
`updatepar.utility.conversion.parfun`
for analogous updates of other nodes

`updatepar.utility.endnode.parfun1d`
Update Parameters in Node Definition

Description

Update parameters in node defintion.

Usage

```
## S3 method for class 'utility.endnode.parfun1d'
updatepar(x, par=NA, ...)
```

Arguments

`x`	node to be updated.
`par`	parameter vector with labelled parameters to be updated.
`...`	currently no other arguments are implemented or passed further.

Value

The node with updated parameters is returned.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See `utility.endnode.parfun1d.create` for how to construct such a node and

`updatepar.utility.endnode.discrete`
`updatepar.utility.endnode.intpol1d`
`updatepar.utility.endnode.intpol2d`
`updatepar.utility.endnode.cond`
`updatepar.utility.endnode.firstavail`
`updatepar.utility.aggregation`
`updatepar.utility.conversion.intpol`
`updatepar.utility.conversion.parfun`
for analogous updates of other nodes

`utility.aggregate.add` *Additive aggregation of values or utilities*

Description

Function to perform an additive aggregation (weighted mean) of values or utilities.

Usage

```
utility.aggregate.add(u, par)
```

Arguments

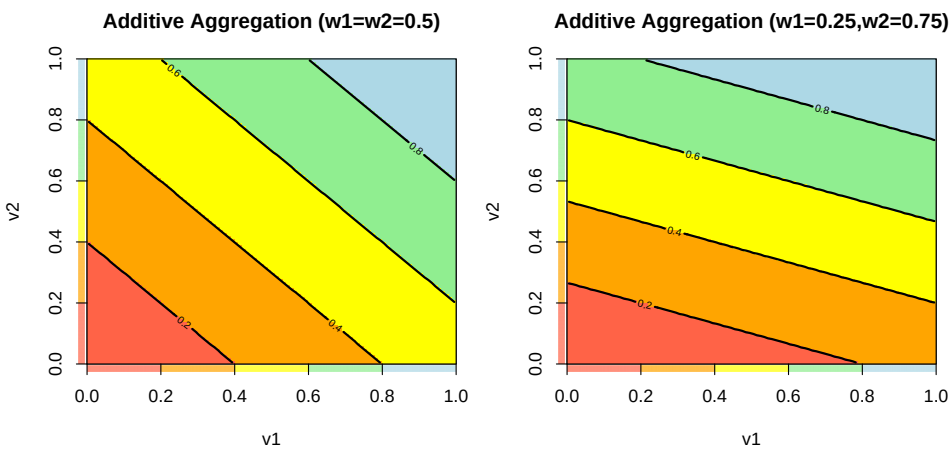
`u`	numeric vector of values or utilities to be aggregated.
`par`	numeric vector of weights for calculating the weighted mean of the values provided in the argument `u`. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector `u`, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = \sum_{i=1}^{n} w_i u_i$$

The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case:



Value

numeric value representing the weighted mean of the components of `u`.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.add(c(0.2,0.8), par=c(1,1))
```

utility.aggregate.addmin

Mixture of additive and minimum aggregation

Description

Function to perform a mixture of additive and minimum aggregation. The parameter vector must contain the weights for additive aggregation followed by the weight of additive aggregation. The weight for minimum aggregation is then unity minus the weight for additive aggregation. If this additional weight is zero, we return to minimum aggregation, if it is unity, we will have additive aggregation.

Usage

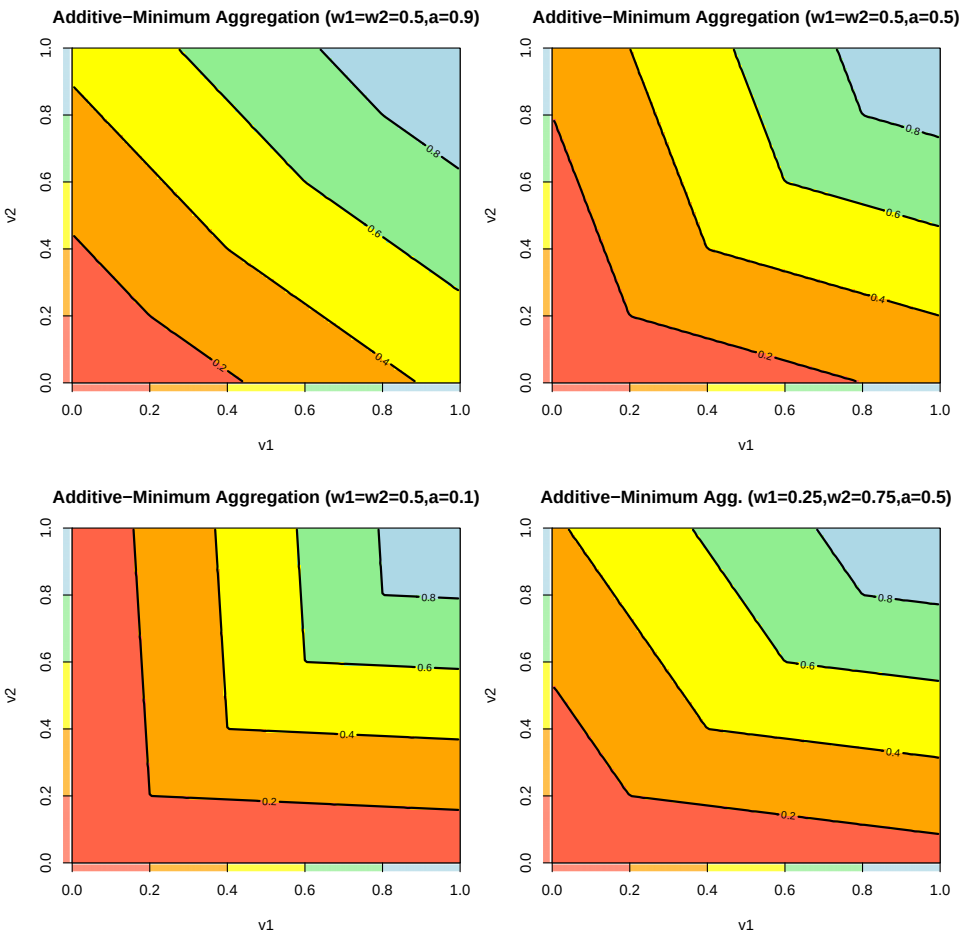
```
utility.aggregate.addmin(u, par)
```

Arguments

u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for additive aggregation appended by the weight for additive aggregation. The weight for minimum aggregation is then unity minus the weight for additive aggregation. If this additional weight is zero, we return to minimum aggregation, if it is unity, we will have additive aggregation. The weights for additive aggregation need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is a mixture of the functions utility.aggregate.add and utility.aggregate.min. The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.addmin(c(0.2,0.8), par=c(1,1,0.5))
```

`utility.aggregate.addpower`
Additive power aggregation of values or utilities

Description

Function to perform a weighted power aggregation of values or utilities.

Usage

```
utility.aggregate.addpower(u, par)
```

Arguments

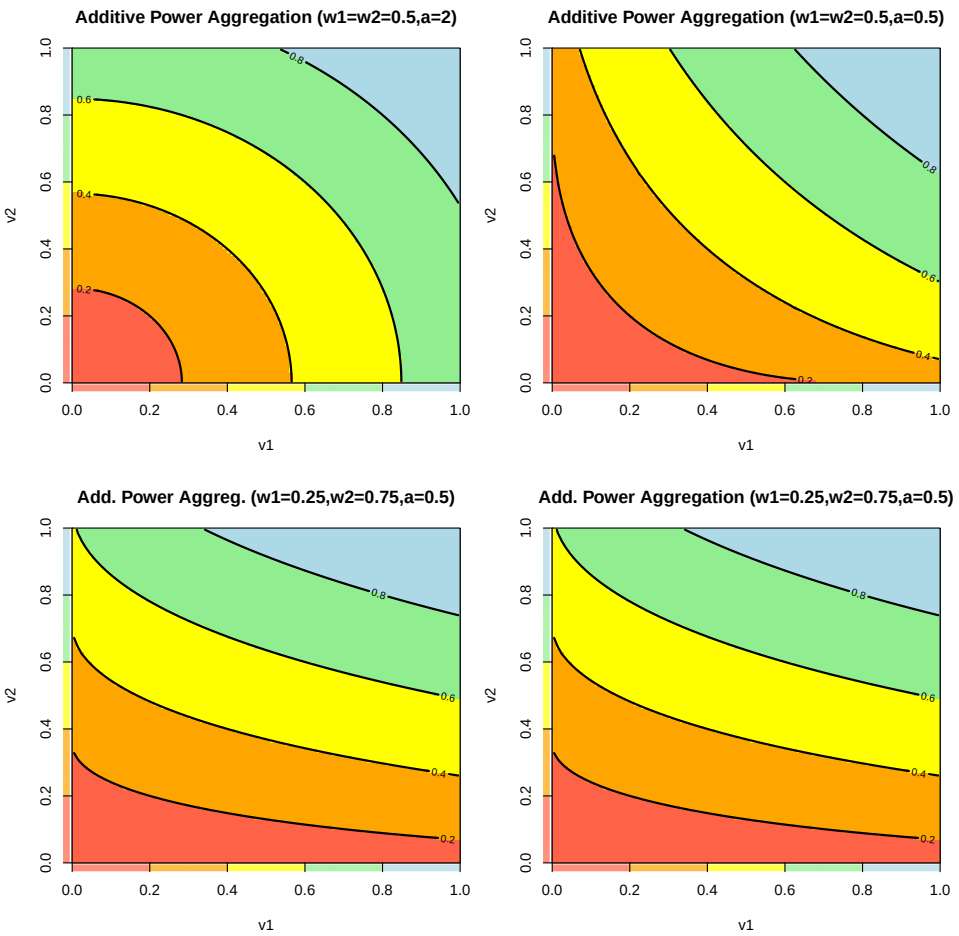
`u`	numeric vector of values or utilities to be aggregated.
`par`	numeric vector of weights appended by the power of the aggregation function (see details below). The weights need not be normalized, they will be normalized before use. In case of missing values in the vector `u`, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = \left(\sum_{i=1}^{n} w_i u_i^{\alpha} \right)^{1/\alpha}$$

where α is the last parameter appended to the weights.
The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.addpower(c(0.2,0.8), par=c(1,1,2))
```

`utility.aggregate.addsplitpower`
Splitted weighted power aggregation of values or utilities

Description

Function to perform a splitted weighted power aggregation of values or utilities.

Usage

```
utility.aggregate.addsplitpower(u, par)
```

Arguments

`u`	numeric vector of values or utilities to be aggregated.
`par`	numeric vector of weights appended by the power of the aggregation function and the position of the split between concave and convex transformation (see details below). The weights need not be normalized, they will be normalized before use. In case of missing values in the vector `u`, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

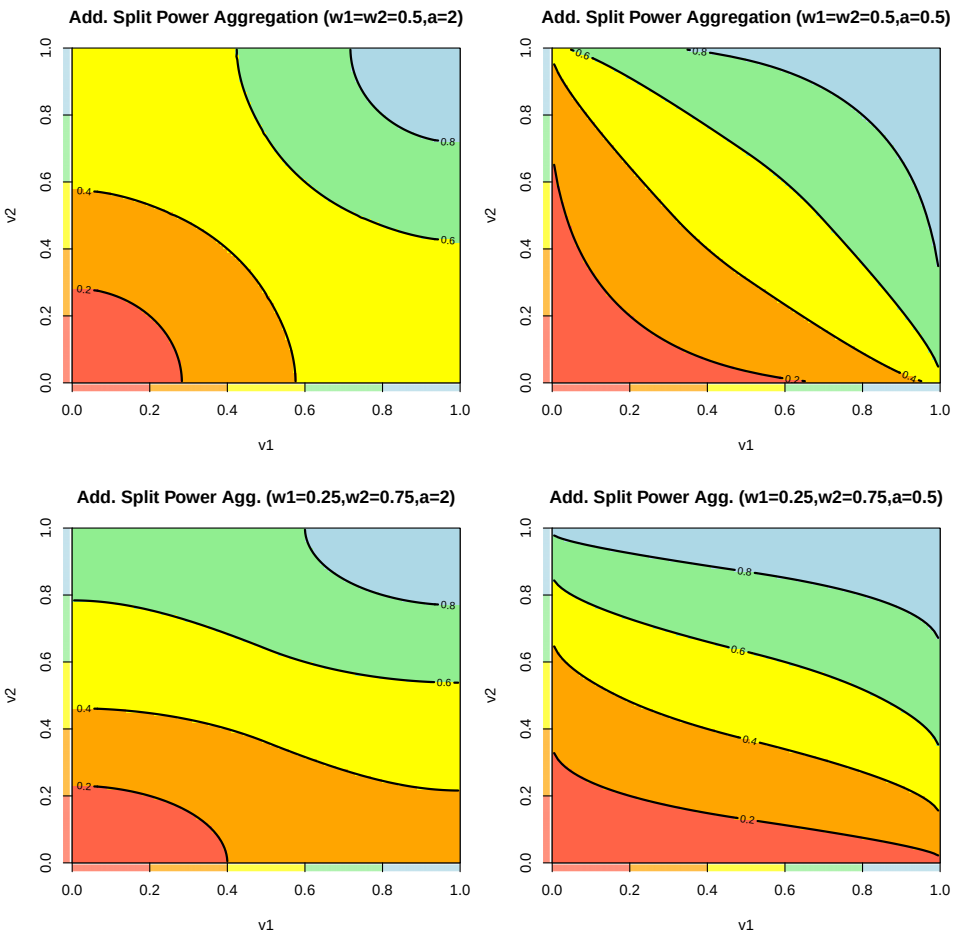
$$u = g^{-1}\left(\sum_{i=1}^{n} w_i g(u_i)\right)$$

with

$$g(v) = \begin{cases} s\left(\frac{v}{s}\right)^{\alpha} & \text{for } v \leq s \\ 1-(1-s)\left(\frac{1-v}{1-s}\right)^{\alpha} & \text{for } v \geq s \end{cases}$$

$$g^{-1}(v) = \begin{cases} s\left(\frac{v}{s}\right)^{1/\alpha} & \text{for } v \leq s \\ 1-(1-s)\left(\frac{1-v}{1-s}\right)^{1/\alpha} & \text{for } v \geq s \end{cases}$$

where α and s are the two last parameters appended to the weights.
The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case (the split parameter, s, is chosen to be 1/2 in all four plots):



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.addsplitpower(c(0.2,0.8), par=c(1,1,2,0.5))
```

utility.aggregate.bonusmalus

Bonus-malus aggregation of values or utilities

Description

Function to perform an aggregation of valus or utilities that considers some of the inputs only as bonus (only considered if value is larger then the aggregated value of the non bonus or malus input) or malus (only considered if value is smaller then the aggregated value of the non bonus or malus input).

Usage

```
utility.aggregate.bonusmalus(u,par,def.agg="utility.aggregate.add")
```

Arguments

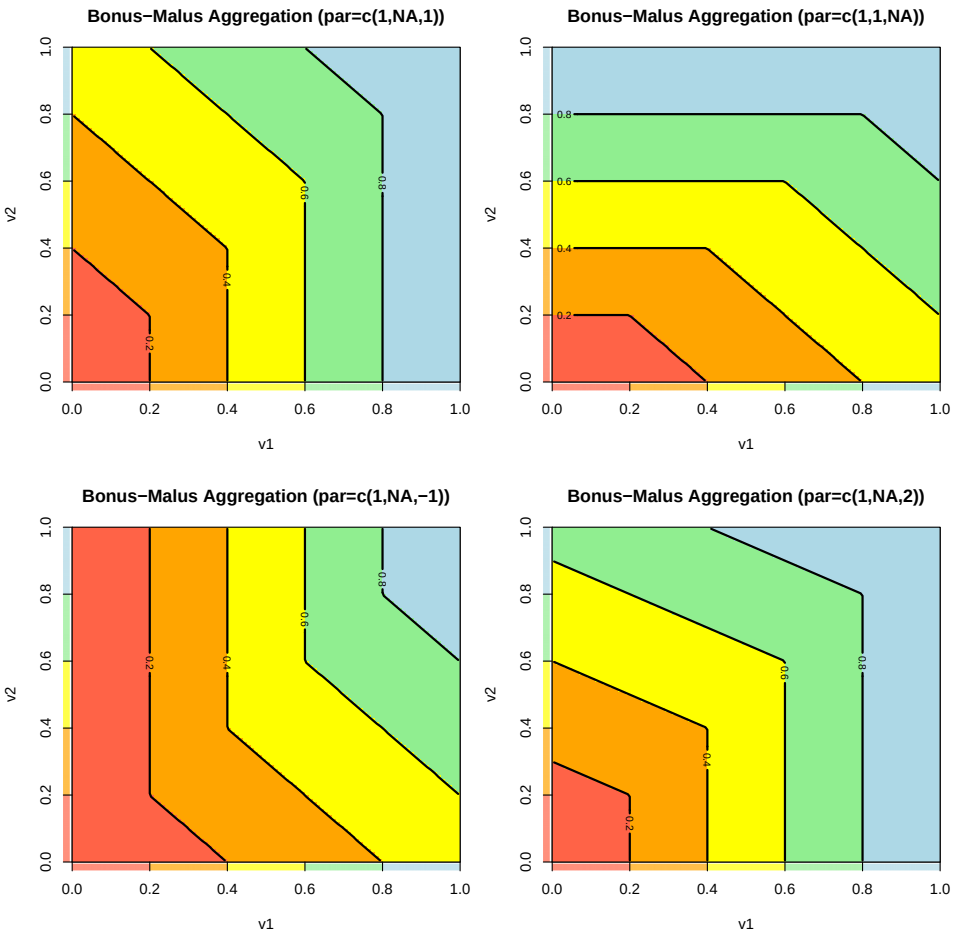
u	numeric vector of values or utilities to be aggregated.
par	numeric vector combining the parameters of the default aggregation technique (see argument def.agg) with those specifying the bonus-malus behaviour. The arguments of def.agg) must match the number of arguments of this function for the number of inputs reduced to those that are not treated as bonus or malus. This parameter vector is then appended by the parameters characterizing the bonus-malus behavior. This is a parameter vector of the same length as the number of sub-objectives. Its elements must be NA for the sub-objectives onsidered for the default aggregation technique, the weights relative to the aggregated value of the non-bonus and non-malus sub-objectives for the sub-objectives to be considered as bonus objectives, and the weights with a negative sign for those to be considered as malus objectives. Note that the weights of the bonus or malus attributes are relative to the aggregated result of the non-bonus and non-malus inputs and the negative signs will only be used for identifying malus sub-objectives and will be eliminated when calculating the weighted mean.
def.agg	(optional) character string specifying the name of the function used for aggregation of the non-bonus and non-malus sub-objectives. Note that for use of this aggregation technique in the function utility.aggregation.create, this argument has to be specified as the input argument def.agg (default aggregation) unless it should be additive (default).

Details

The aggregation function is defined by

$$u = \frac{u^{\text{agg}}_{i \notin b, i \notin m} + \sum\limits_{\substack{i \in b \wedge u_i > u^{\text{agg}}_{i \notin b, i \notin m} \\ i \in m \wedge u_i < u^{\text{agg}}_{i \notin b, i \notin m}}} \mid w_i \mid u_i}{1 + \sum\limits_{\substack{i \in b \wedge u_i > u^{\text{agg}}_{i \notin b, i \notin m} \\ i \in m \wedge u_i < u^{\text{agg}}_{i \notin b, i \notin m}}} \mid w_i \mid}$$

The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Note

This is the same function as utility.aggregate.cobbdouglas

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

utility.aggregation.create

Aggregation techniques provided by uncsim:

utility.aggregate.add for additive aggregation (weighted arithmetic mean),
utility.aggregate.min for minimum aggregation,
utility.aggregate.max for maximum aggregation,
utility.aggregate.geo or utility.aggregate.cobbdouglas for geometric or Cobb-Douglas aggregation (weighted geometric mean),
utility.aggregate.geooff for geometric aggregation with offset,
utility.aggregate.revgeo for reverse geometric aggregation,
utility.aggregate.revgeooff for reverse geometric aggregation with offset,
utility.aggregate.harmo for harmonic aggregation (weighted harmonic mean),
utility.aggregate.harmooff for harmonic aggregation with offset,
utility.aggregate.revharmo for reverse harmonic aggregation,
utility.aggregate.revharmooff for reverse harmonic aggregation with offset,
utility.aggregate.mult for multiplicative aggregation,
utility.aggregate.mix for a mixture of additive, minimum, and geometric aggregation,
utility.aggregate.addmin for a mixture of additive and minimum aggregation.

utility.aggregate.addpower for additive power aggregation (weighted power mean),
utility.aggregate.revaddpower for reverse additive power aggregation,
utility.aggregate.addsplitpower for splitted additive power aggregation,
utility.aggregate.revaddsplitpower for reverse splitted additive power aggregation,
utility.aggregate.bonusmalus for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.bonusmalus(c(0.2,0.8), par=c(1,NA,1))
utility.aggregate.bonusmalus(c(0.2,0.8), par=c(1,1,NA))
utility.aggregate.bonusmalus(c(0.2,0.8), par=c(1,NA,-1))
utility.aggregate.bonusmalus(c(0.2,0.8), par=c(1,-1,NA))
```

---

utility.aggregate.cobbdouglas

Cobb-Douglas aggregation of values or utilities

Description

Function to perform a Cobb-Douglas aggregation (weighted geometric mean) of values or utilities.

Usage

```
utility.aggregate.cobbdouglas(u, par)
```

Arguments

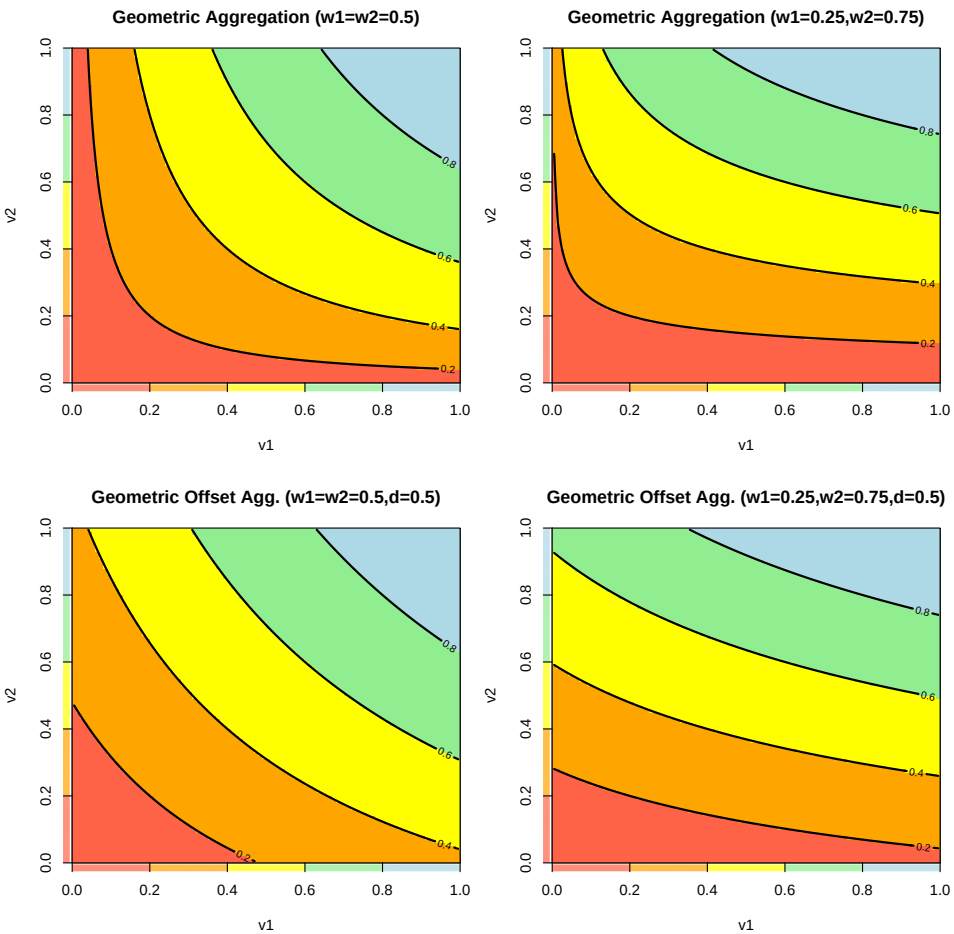
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for calculating the weighted geometric mean of the values provided in the argument u. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = \prod_{i=1}^{n} u_i^{w_i}$$

The following figure shows examples of the behaviour of this aggregation function and its generalization to utility.aggregate.geooff for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Note

This is the same function as `utility.aggregate.geo`

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.cobbdouglas(c(0.2,0.8), par=c(1,1))
```

`utility.aggregate.geo` *Geometric aggregation of values or utilities*

Description

Function to perform a geometric aggregation (weighted geometric mean) of values or utilities.

Usage

```
utility.aggregate.geo(u, par)
```

Arguments

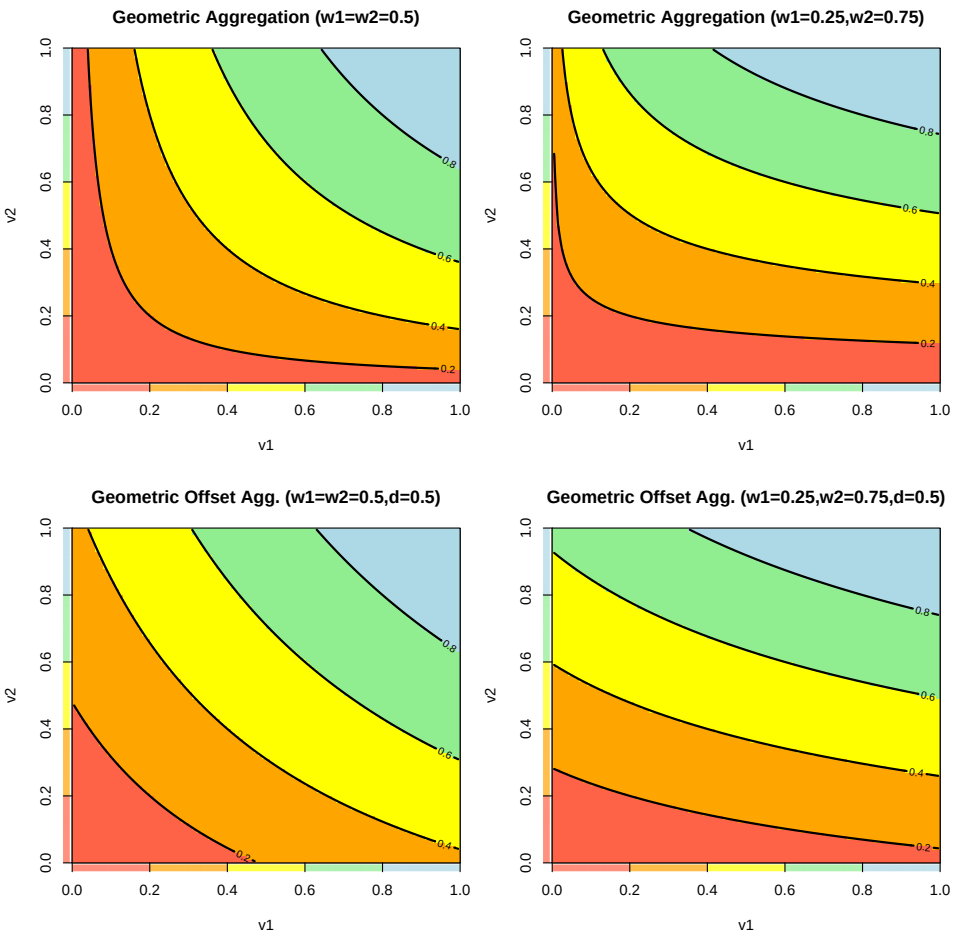
`u`	numeric vector of values or utilities to be aggregated.
`par`	numeric vector of weights for calculating the weighted geometric mean of the values provided in the argument `u`. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector `u`, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = \prod_{i=1}^{n} u_i^{w_i}$$

The following figure shows examples of the behaviour of this aggregation function and its generalization to `utility.aggregate.geooff` for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Note

This is the same function as `utility.aggregate.cobbdouglas`

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.geo(c(0.2,0.8), par=c(1,1))
```

utility.aggregate.geooff

Geometric aggregation of values or utilities with offset

Description

Function to perform a geometric aggregation (weighted geometric mean) of values or utilities with offset. The offset is added to the arguments and subtracted from the result.

Usage

```
utility.aggregate.geooff(u, par)
```

Arguments

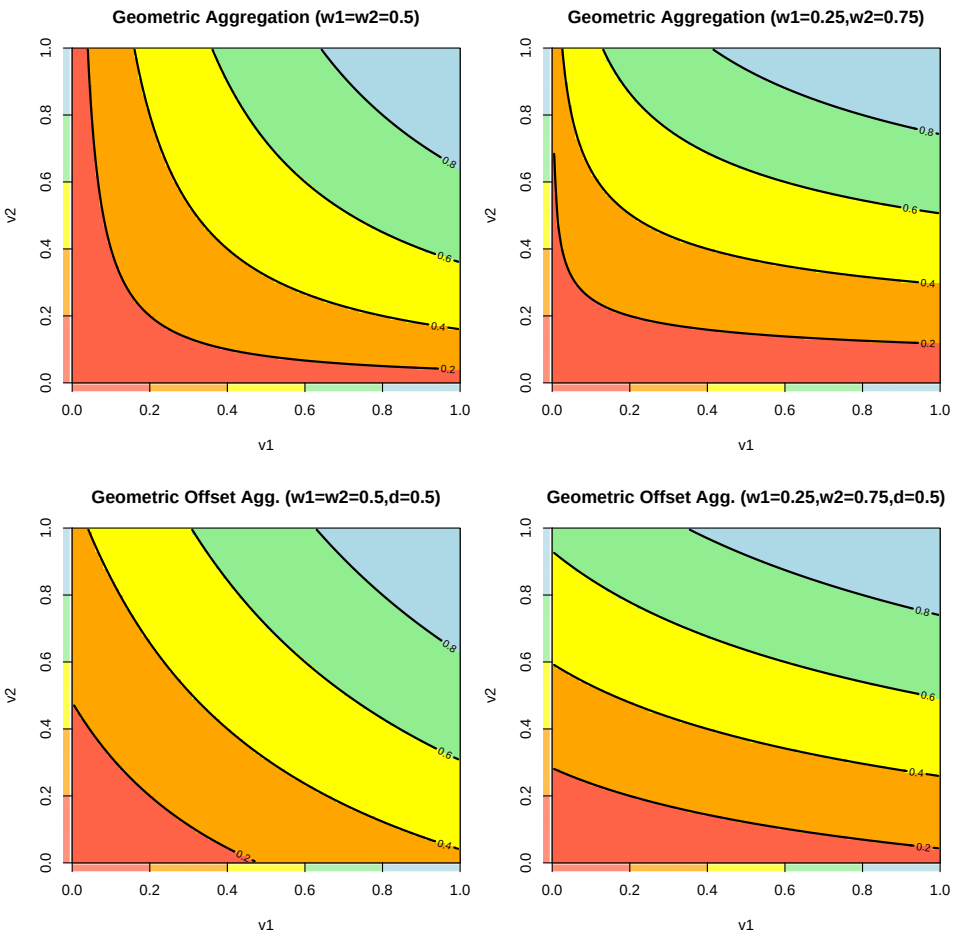
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights appended by an offset for calculating the weighted geometric mean minus an offset of the values provided in the argument u plus the offset. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = \prod_{i=1}^{n} (u_i + \delta)^{w_i} - \delta$$

where δ is the last parameter appended to the weights.
The following figure shows examples of the behaviour of this aggregation function and its special case utility.aggregate.geo for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.geooff(c(0.2,0.8), par=c(1,1,0.1))
```

utility.aggregate.harmo

Harmonic aggregation of values or utilities

Description

Function to perform a harmonic aggregation (weighted harmonic mean) of values or utilities.

Usage

```
utility.aggregate.harmo(u, par)
```

Arguments

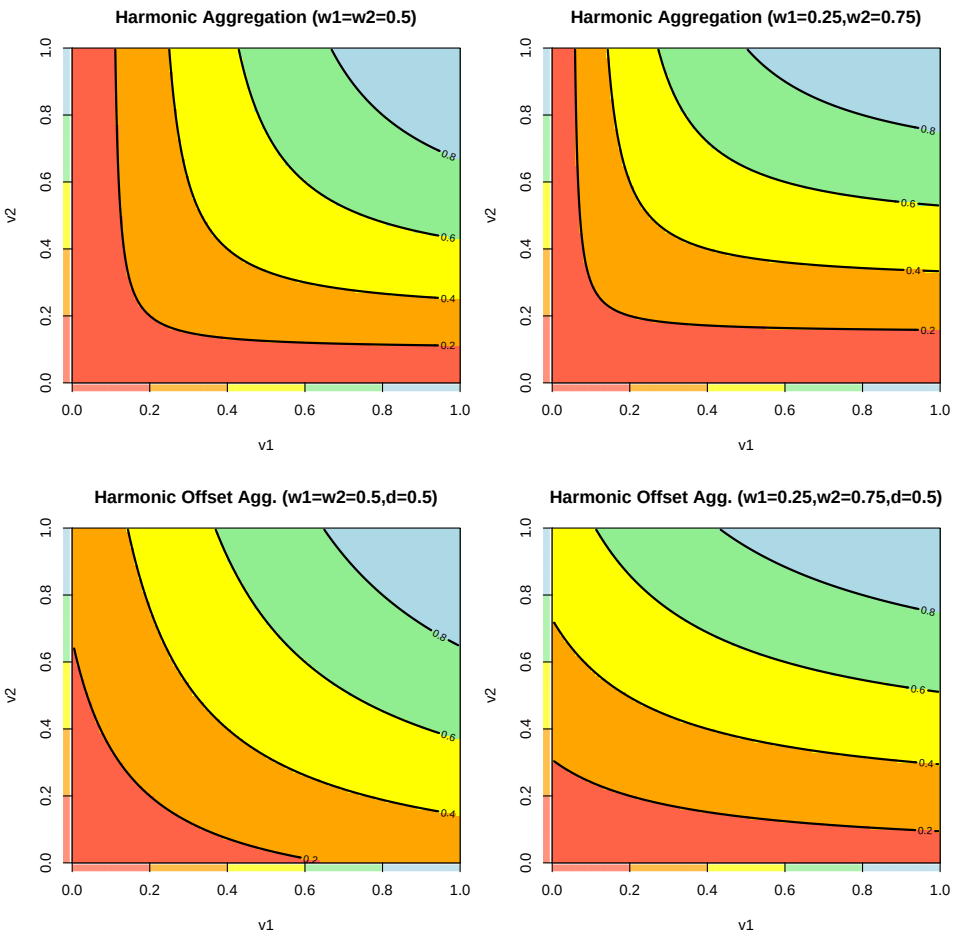
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for calculating the weighted harmonic mean of the values provided in the argument u. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = \frac{1}{\sum_{i=1}^{n} \frac{w_i}{u_i}}$$

The following figure shows examples of the behaviour of this aggregation function and its generalization to utility.aggregate.harmooff for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.harmo(c(0.2,0.8), par=c(1,1))
```

utility.aggregate.harmooff

Harmonic aggregation of values or utilities with offset

Description

Function to perform a harmonic aggregation (weighted harmonic mean) of values or utilities with offset. The offset is added to the arguments and subtracted from the result.

Usage

```
utility.aggregate.harmooff(u, par)
```

Arguments

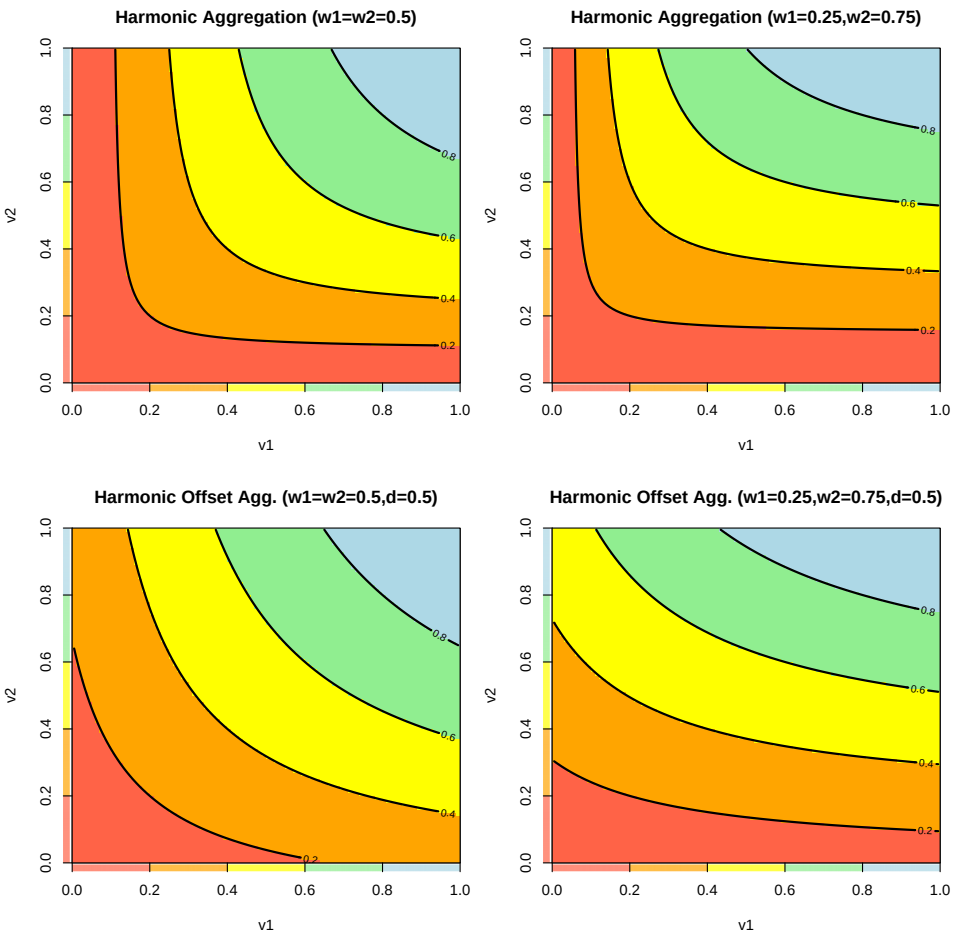
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights appended by an offset for calculating the weighted harmonic mean minus an offset of the values provided in the argument u plus the offset. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = \frac{1}{\sum_{i=1}^{n} \frac{w_i}{u_i + \delta}} - \delta$$

where δ is the last parameter appended to the weights.
The following figure shows examples of the behaviour of this aggregation function and its special case utility.aggregate.harmo for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.harmooff(c(0.2,0.8), par=c(1,1,0.1))
```

`utility.aggregate.max` *Maximum aggregation of values or utilities*

Description

Function to perform a maximum aggregation of values or utilities.

Usage

```
utility.aggregate.max(u, par = NA)
```

Arguments

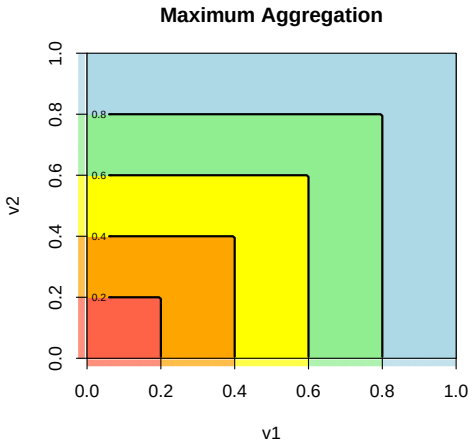
`u`	numeric vector of values or utilities to be aggregated.
`par`	unused argument used for compatibility with other aggregation techniques that require parameters.

Details

The aggregation function is defined by

$$u = \max_{i=1}^{n} u_i$$

The following figure shows the behaviour of this aggregation function for the two-dimensional case:



Value

maximum of the components of u.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.max(c(0.2,0.8))
```

`utility.aggregate.min` *Minimum aggregation of values or utilities*

Description

Function to perform a minimum aggregation of values or utilities.

Usage

```
utility.aggregate.min(u, par = NA)
```

Arguments

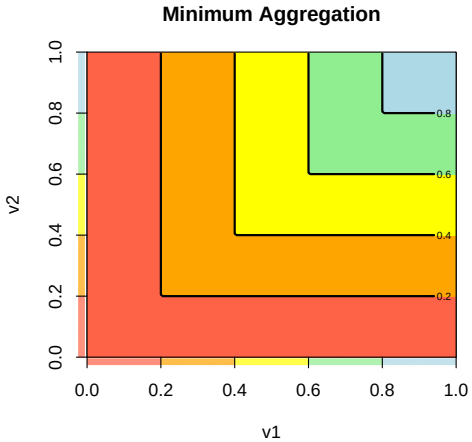
`u`	numeric vector of values or utilities to be aggregated.
`par`	unused argument used for compatibility with other aggregation techniques that require parameters.

Details

The aggregation function is defined by

$$u = \min_{i=1}^{n} u_i$$

The following figure shows the behaviour of this aggregation function for the two-dimensional case:



Value

minimum of the components of `u`.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,

utility.aggregate.revaddsplitpower for reverse splitted additive power aggregation, utility.aggregate.bonusmalus for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.min(c(0.2,0.8))
```

---

utility.aggregate.mix *Mixed aggregation of values and utilities*

Description

Function to perform a mixed aggregation of values and utilities. The mixture consists of a weighted mean of the additive, minimum and geometric aggregation techniques.

Usage

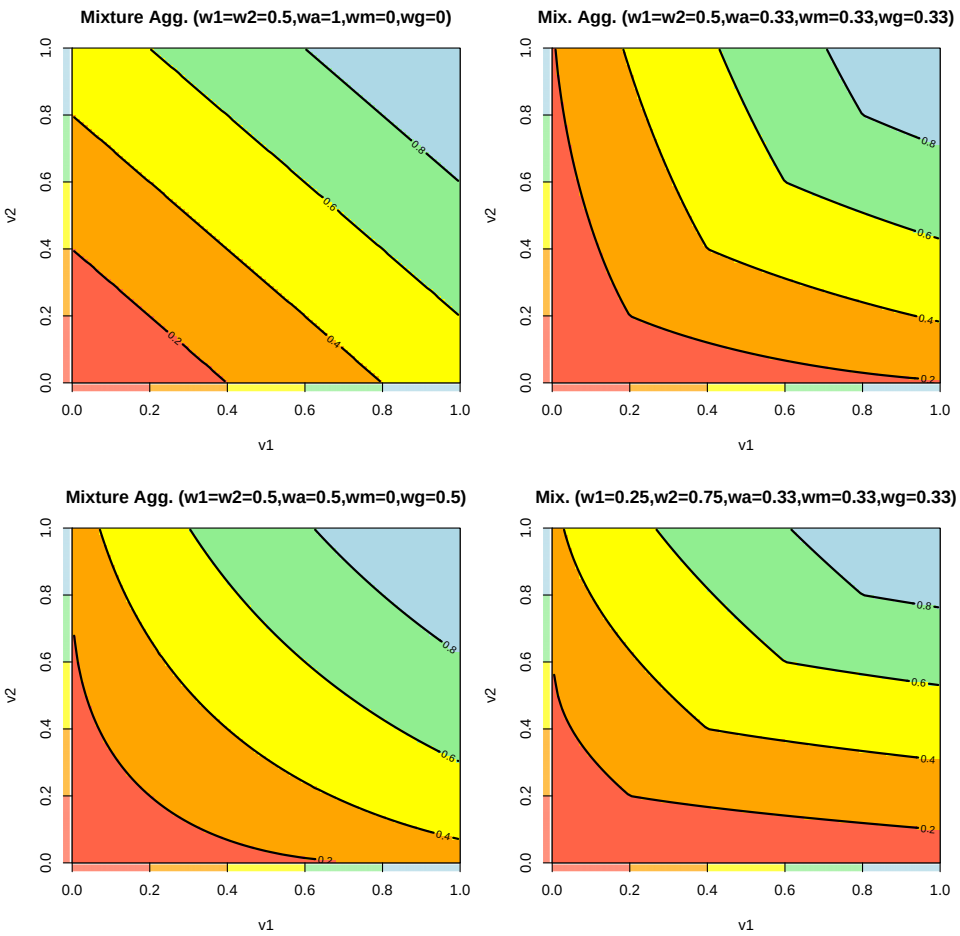
```
utility.aggregate.mix(u, par)
```

Arguments

u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for calculating the weighted mean of the values provided in the argument u followed by the three weights of the additive, minimum and geometric aggregation techniques. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is a mixture of the functions utility.aggregate.add, utility.aggregate.min, and utility.aggregate.geo. The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

utility.aggregation.create

Aggregation techniques provided by uncsim:

utility.aggregate.add for additive aggregation (weighted arithmetic mean),
utility.aggregate.min for minimum aggregation,
utility.aggregate.max for maximum aggregation,
utility.aggregate.geo or utility.aggregate.cobbdouglas for geometric or Cobb-Douglas aggregation (weighted geometric mean),
utility.aggregate.geooff for geometric aggregation with offset,
utility.aggregate.revgeo for reverse geometric aggregation,
utility.aggregate.revgeooff for reverse geometric aggregation with offset,
utility.aggregate.harmo for harmonic aggregation (weighted harmonic mean),
utility.aggregate.harmooff for harmonic aggregation with offset,
utility.aggregate.revharmo for reverse harmonic aggregation,
utility.aggregate.revharmooff for reverse harmonic aggregation with offset,
utility.aggregate.mult for multiplicative aggregation,
utility.aggregate.mix for a mixture of additive, minimum, and geometric aggregation,
utility.aggregate.addmin for a mixture of additive and minimum aggregation.
utility.aggregate.addpower for additive power aggregation (weighted power mean),
utility.aggregate.revaddpower for reverse additive power aggregation,
utility.aggregate.addsplitpower for splitted additive power aggregation,
utility.aggregate.revaddsplitpower for reverse splitted additive power aggregation,
utility.aggregate.bonusmalus for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.mix(c(0.2,0.8),par=c(1,1 , 1,0,0))
utility.aggregate.mix(c(0.2,0.8),par=c(1,1 , 0,1,0))
utility.aggregate.mix(c(0.2,0.8),par=c(1,1 , 0,0,1))
utility.aggregate.mix(c(0.2,0.8),par=c(1,1 , 1,1,1))
```

---

utility.aggregate.mult

Multiplicative aggregation of values or utilities

Description

Function to perform a multiplicative aggregation of values or utilities.

Usage

```
utility.aggregate.mult(u, par)
```

Arguments

`u`	numeric vector of values or utilities to be aggregated.
`par`	numeric vector of weights for calculating the multiplicative combination of the values provided in the argument `u`. Note that for this aggregation technique, the result depends on the sum of the weights that need not be unity.

Details

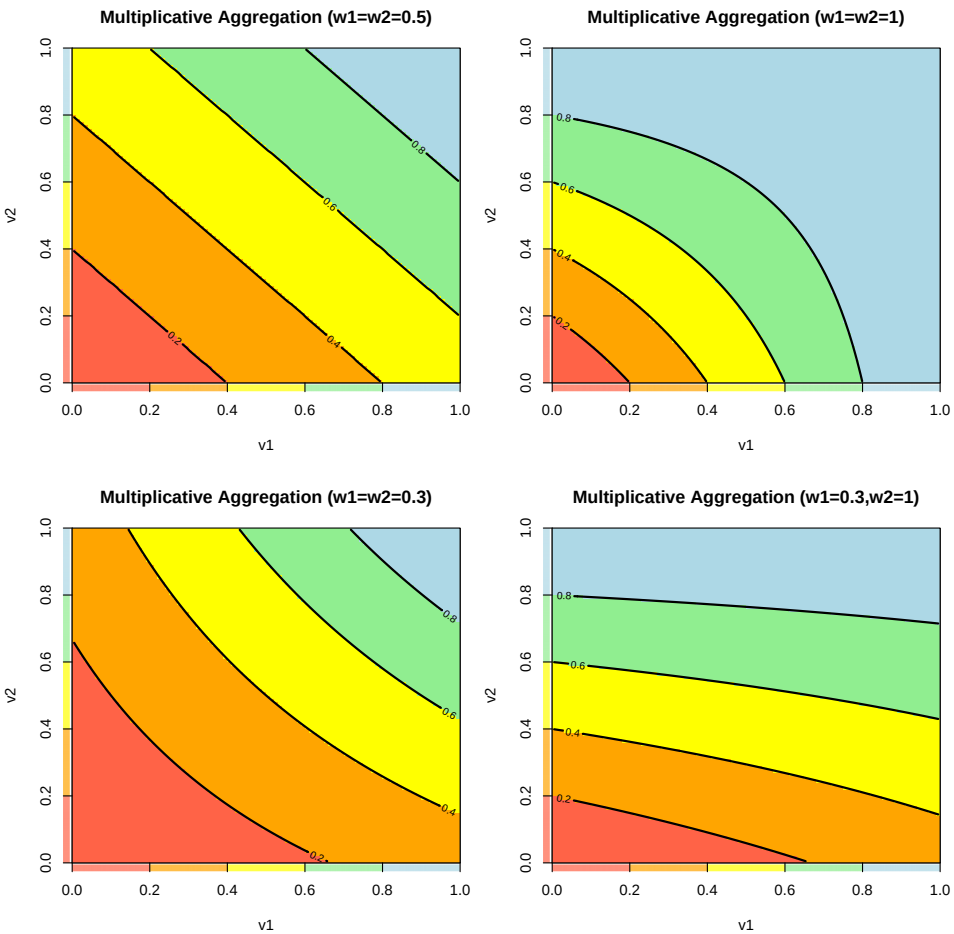
To derive the aggregated value, we first solve the implicit equation

$$k + 1 = \prod_{i=1}^{n} (1 + k w_i)$$

for k and then calculate the aggregated value as

$$u = \frac{\prod_{i=1}^{n} (1 + k w_i u_i)}{k}$$

See Keeney and Raiffa, Decisions with multiple objectives, 1976, pp. 307, 347-348 for details.
The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case:



Value

numeric value corresponding to the multiplicative aggregation of the values provided in the vector u.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.mult(c(0.2,0.8),par=c(0.3,0.3))
```

`utility.aggregate.revaddpower`
Reverse additive power aggregation of values or utilities

Description

Function to perform a reverse weighted power aggregation of values or utilities.

Usage

```
utility.aggregate.revaddpower(u, par)
```

Arguments

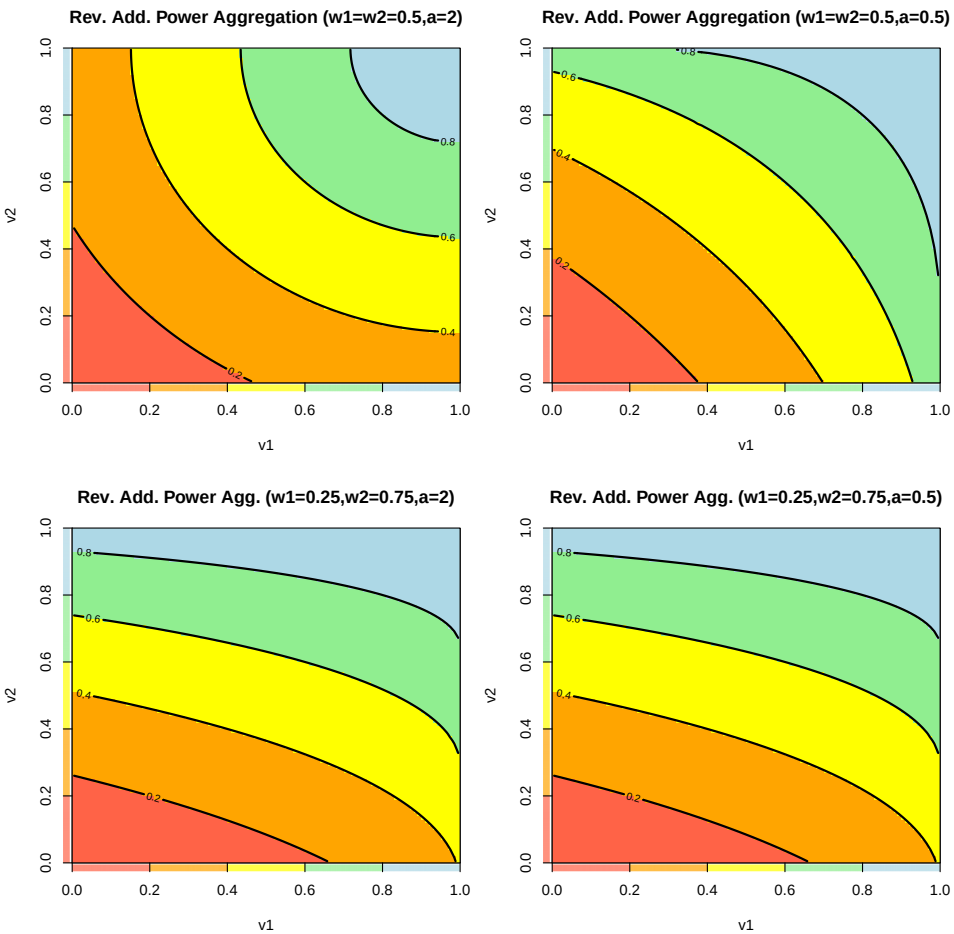
`u`	numeric vector of values or utilities to be aggregated.
`par`	numeric vector of weights appended by the power of the aggregation function (see details below). The weights need not be normalized, they will be normalized before use. In case of missing values in the vector `u`, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = 1 - \left(\sum_{i=1}^{n} w_i (1 - u_i)^{\alpha} \right)^{1/\alpha}$$

where α is the last parameter appended to the weights.
The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.revaddpower(c(0.2,0.8), par=c(1,1,2))
```

`utility.aggregate.revaddsplitpower`
Reverse splitted additive power aggregation of values or utilities

Description

Function to perform a reverse splitted weighted power aggregation of values or utilities.

Usage

```
utility.aggregate.revaddsplitpower(u, par)
```

Arguments

`u`	numeric vector of values or utilities to be aggregated.
`par`	numeric vector of weights appended by the power of the aggregation function and the position of the split between concave and convex transformation (see details below). The weights need not be normalized, they will be normalized before use. In case of missing values in the vector `u`, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

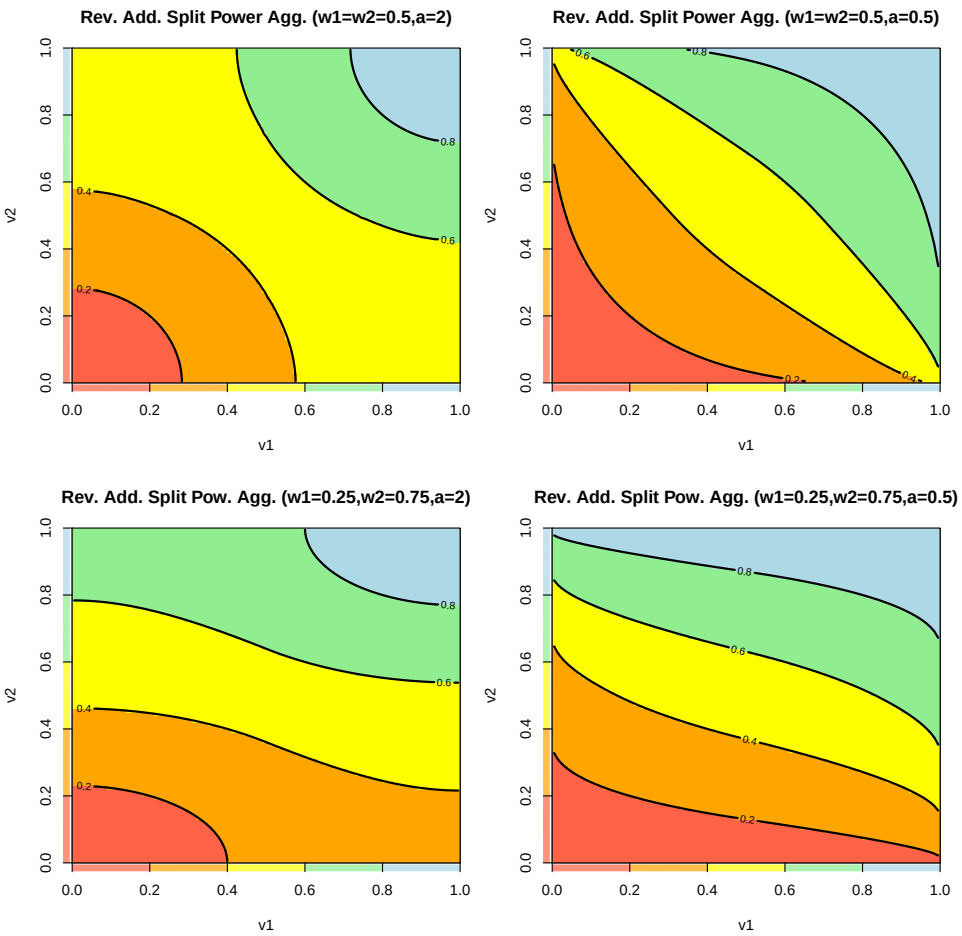
$$u = 1 - g^{-1}\left(\sum_{i=1}^{n} w_i g(1-u_i)\right)$$

with

$$g(v) = \begin{cases} s\left(\frac{v}{s}\right)^{\alpha} & \text{for } v \leq s \\ 1-(1-s)\left(\frac{1-v}{1-s}\right)^{\alpha} & \text{for } v \geq s \end{cases}$$

$$g^{-1}(v) = \begin{cases} s\left(\frac{v}{s}\right)^{1/\alpha} & \text{for } v \leq s \\ 1-(1-s)\left(\frac{1-v}{1-s}\right)^{1/\alpha} & \text{for } v \geq s \end{cases}$$

where α and s are the two last parameters appended to the weights.
The following figure shows examples of the behaviour of this aggregation function for the two-dimensional case (the split parameter, s, is chosen to be 1/2 in all four plots):



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.revaddsplitpower(c(0.2,0.8,0.5), par=c(1,1,2,0.5))
```

utility.aggregate.revgeo

Reverse geometric aggregation of values or utilities

Description

Function to perform a reverse geometric aggregation (unity minus the weighted geometric mean of unity minus the arguments) of values or utilities.

Usage

```
utility.aggregate.revgeo(u, par)
```

Arguments

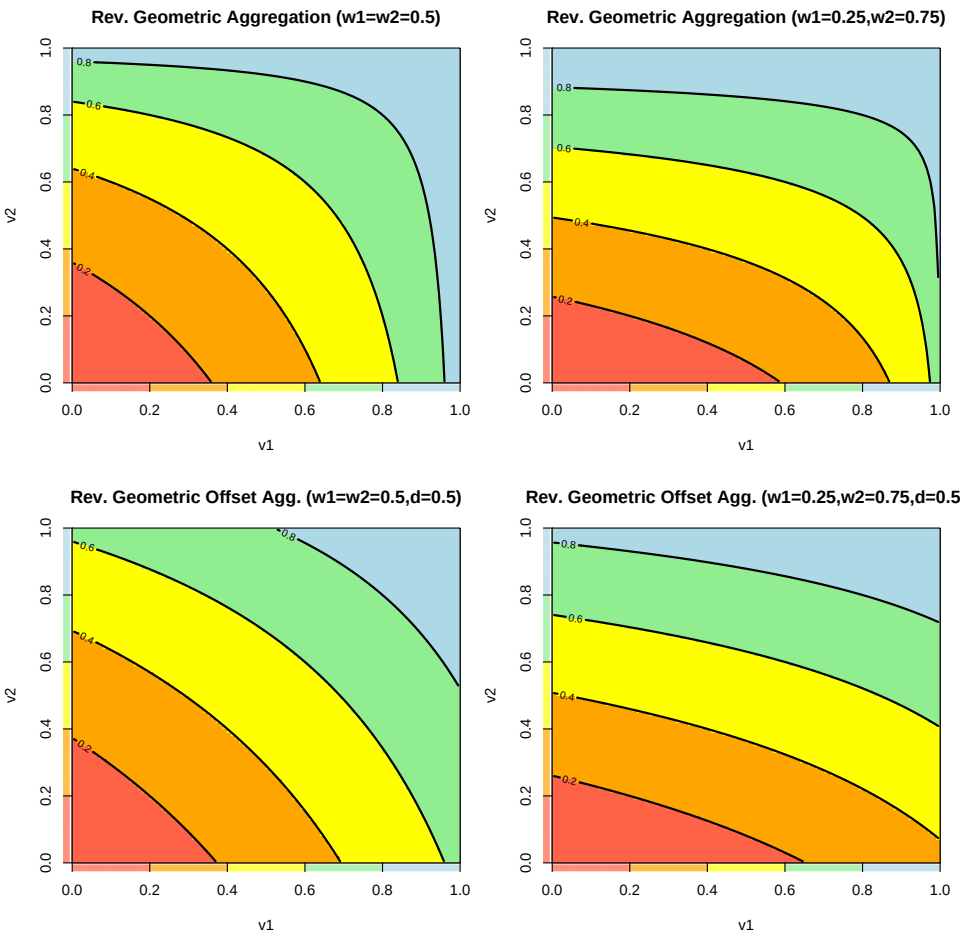
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for calculating the reverse weighted geometric mean of the values provided in the argument u. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = 1 - \prod_{i=1}^{n} (1 - u_i)^{w_i}$$

The following figure shows examples of the behaviour of this aggregation function and its generalization to utility.aggregate.revgeooff for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.revgeo(c(0.2,0.8), par=c(1,1))
```

utility.aggregate.revgeooff

Reverse geometric aggregation of values or utilities with offset

Description

Function to perform a reverse geometric aggregation (unity minus the weighted geometric mean of unity minus the arguments) of values or utilities with offset.

Usage

```
utility.aggregate.revgeooff(u, par)
```

Arguments

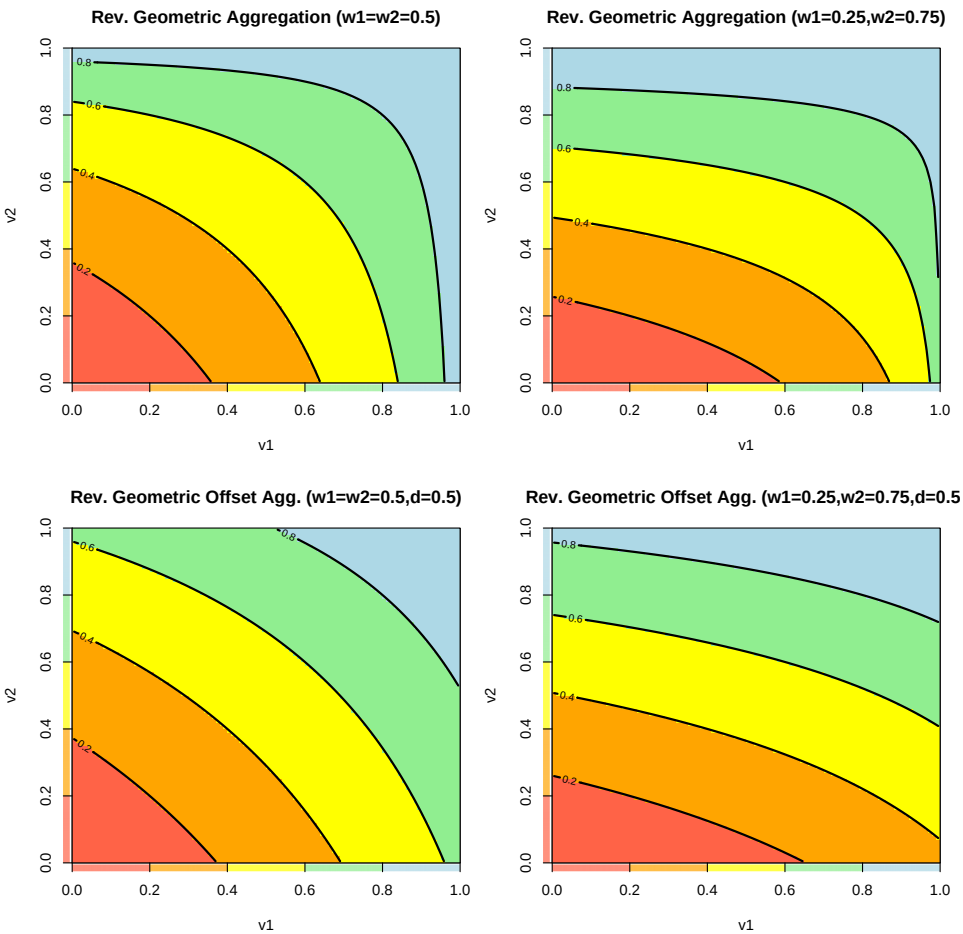
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for calculating the reverse weighted geometric mean of the values provided in the argument u. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = 1 - \prod_{i=1}^{n} (1 - u_i + \delta)^{w_i} + \delta$$

where δ is the last parameter appended to the weights.
The following figure shows examples of the behaviour of this aggregation function and its special case utility.aggregate.revgeo for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.revgeooff(c(0.2,0.8), par=c(1,1,0.1))
```

utility.aggregate.revharmo

Reverse harmonic aggregation of values or utilities

Description

Function to perform a reverse harmonic aggregation (unity minus the weighted harmonic mean of unity minus the arguments) of values or utilities.

Usage

```
utility.aggregate.revharmo(u, par)
```

Arguments

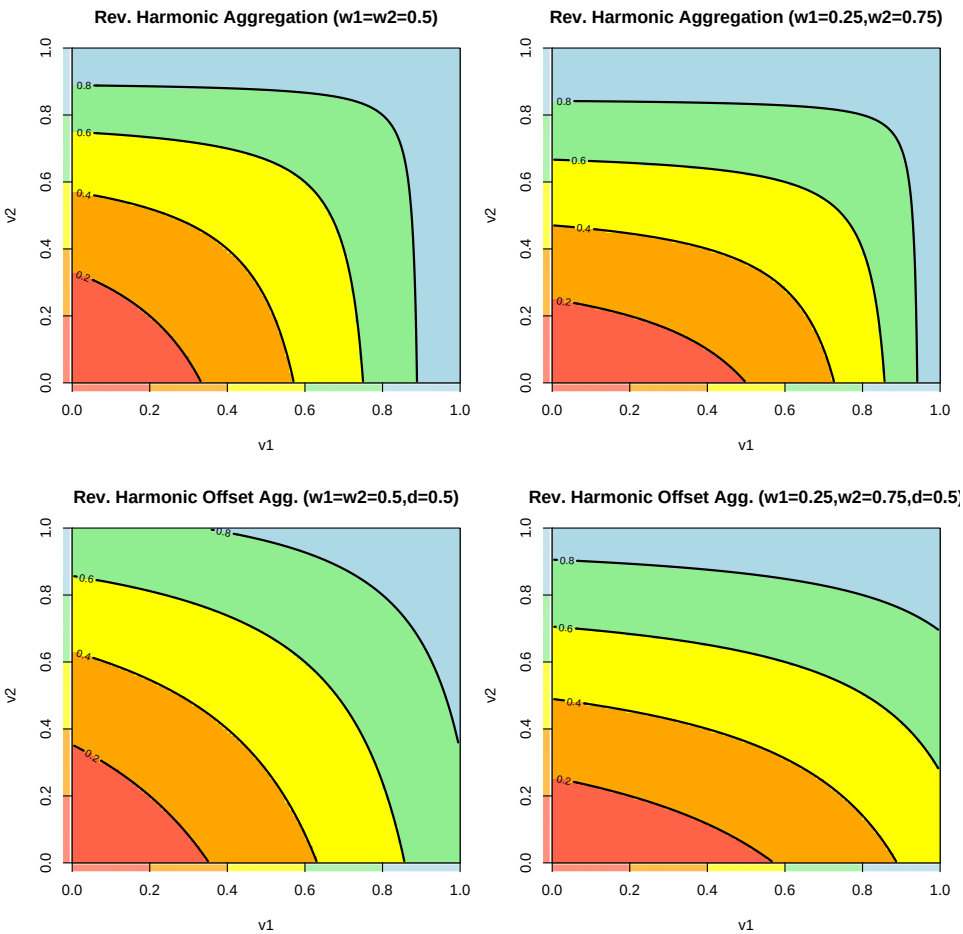
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for calculating the reverse weighted harmonic mean of the values provided in the argument u. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = 1 - \frac{1}{\sum_{i=1}^{n} \frac{w_i}{1 - u_i}}$$

The following figure shows examples of the behaviour of this aggregation function and its generalization to utility.aggregate.revharmooff for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.revharmo(c(0.2,0.8), par=c(1,1))
```

utility.aggregate.revharmooff

Reverse harmonic aggregation of values or utilities with offset

Description

Function to perform a reverse harmonic aggregation (unity minus the weighted harmonic mean of unity minus the arguments) of values or utilities with offset.

Usage

```
utility.aggregate.revharmooff(u, par)
```

Arguments

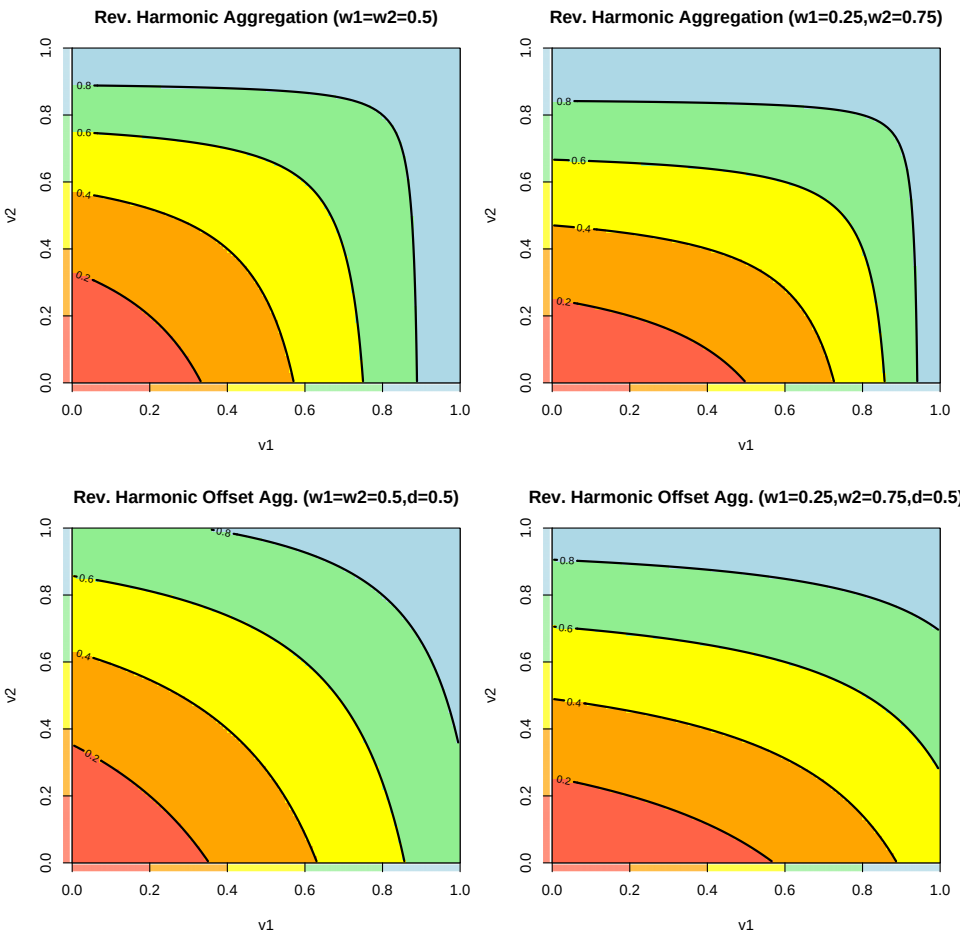
u	numeric vector of values or utilities to be aggregated.
par	numeric vector of weights for calculating the reverse weighted harmonic mean of the values provided in the argument u. The weights need not be normalized, they will be normalized before use. In case of missing values in the vector u, the weights of the non-missing components will be rescaled to sum to unity.

Details

The aggregation function is defined by

$$u = 1 - \frac{1}{\sum_{i=1}^{n} \frac{w_i}{1 - u_i + \delta}} + \delta$$

where δ is the last parameter appended to the weights.
The following figure shows examples of the behaviour of this aggregation function and its special case utility.aggregate.revharmo for the two-dimensional case:



Value

The function returns the aggregated value or utility.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Constructor of aggregation node:

`utility.aggregation.create`

Aggregation techniques provided by uncsim:

`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric aggregation,
`utility.aggregate.addmin` for a mixture of additive and minimum aggregation.
`utility.aggregate.addpower` for additive power aggregation (weighted power mean),
`utility.aggregate.revaddpower` for reverse additive power aggregation,
`utility.aggregate.addsplitpower` for splitted additive power aggregation,
`utility.aggregate.revaddsplitpower` for reverse splitted additive power aggregation,
`utility.aggregate.bonusmalus` for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.

Examples

```
utility.aggregate.revharmooff(c(0.2,0.8), par=c(1,1,0.1))
```

utility.aggregation.create

Construct an aggregation node

Description

Function to construct an aggregation node for value or utlity functions.

Usage

```
utility.aggregation.create(name.node,
                           nodes,
                           name.fun,
                           par,
                           names.par    = rep(NA, length(par)),
                           required     = FALSE,
                           num.required = 1,
                           col          = "black",
                           shift.levels = 0,
                           add.arg.fun  = NULL)
```

Arguments

`name.node` name of the node to be constructed as a character string.

`nodes` list of nodes to be aggregated.

`name.fun` name of the function to be used for aggregation. This function must accept the arguments `u` and `par` which pass a vector of values or utilities to be aggregated and the parameters of the function, respectively. The function can have an additional argument specified below as `add.arg.fun`. The function must then return the corresponding aggregated value or utility.
Examples of functions provided by the package are
`utility.aggregate.add` for additive aggregation (weighted arithmetic mean),
`utility.aggregate.min` for minimum aggregation,
`utility.aggregate.max` for maximum aggregation,
`utility.aggregate.geo` or `utility.aggregate.cobbdouglas` for geometric or Cobb-Douglas aggregation (weighted geometric mean),
`utility.aggregate.geooff` for geometric aggregation with offset,
`utility.aggregate.revgeo` for reverse geometric aggregation,
`utility.aggregate.revgeooff` for reverse geometric aggregation with offset,
`utility.aggregate.harmo` for harmonic aggregation (weighted harmonic mean),
`utility.aggregate.harmooff` for harmonic aggregation with offset,
`utility.aggregate.revharmo` for reverse harmonic aggregation,
`utility.aggregate.revharmooff` for reverse harmonic aggregation with offset,
`utility.aggregate.mult` for multiplicative aggregation,
`utility.aggregate.mix` for a mixture of additive, minimum, and geometric

aggregation,
utility.aggregate.addmin for a mixture of additive and minimum aggregation.
utility.aggregate.addpower for additive power aggregation (weighted power mean),
utility.aggregate.revaddpower for reverse additive power aggregation,
utility.aggregate.addsplitpower for splitted additive power aggregation,
utility.aggregate.revaddsplitpower for reverse splitted additive power aggregation,
utility.aggregate.bonusmalus for an aggregation technique that considers some of the values or utilities of sub-objectives only as bonus or malus.
Follow the links for the aggregation functions for their use, for the underlying mathematical expressions and for graphical illustrations.

`par` numeric vector of parameter values to be passed to the function specified unter `name.fun`.

`names.par` (optional) vector of parameter names corresponging to the vector of values specified under `par`. Only required to provide access to the values through a named parameter vector.

`required` (optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.

`num.required` number of lower-level values or utilities that must at least be available to make the evaluation possible.

`col` (optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.

`shift.levels` (optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

`add.arg.fun` (optional) an additional argument to the aggregation function `name.fun`. The value(s) given here will alway be passed to the aggregation function.

Value

The function returns the created object of type `utility.aggregation` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Description of aggregation techniques:

Langhans, S.D., Reichert, P. and Schuwirth, N., The method matters: A guide for indicator aggregation in ecological assessments. Ecological Indicators 45, 494-507, 2014.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

print.utility.aggregation,
summary.utility.aggregation,
evaluate.utility.aggregation and
plot.utility.aggregation.

Create end nodes with

utility.endnode.discrete.create,
utility.endnode.intpol1d.create,
utility.endnode.intpol2d.create,
utility.endnode.parfun1d.create,
utility.endnode.cond.create, or
utility.endnode.firstavail.create.

Create conversion nodes with

utility.conversion.intpol.create, or
utility.conversion.parfun.create.

Examples

```
# define discrete end node for width variability
# (attribute "widthvariability_class" with levels "high",
# "moderate" and "none")

widthvar <-
  utility.endnode.discrete.create(
    name.node     = "width variability",
    attrib.levels = data.frame(widthvariability_class=
                               c("high","moderate","none")),
    u             = c(1,0.4125,0),
    names.u       = c("u.high","u_moderate","u.none"),
    required      = FALSE,
    utility       = FALSE)
```

```
# define 1d interpolation end node for bed modification with
# riprap
# (attribute "bedmodfract_percent" with levels from 0 to 100)

bedmod_riprap <-
  utility.endnode.intpol1d.create(
    name.node   = "bed modification riprap",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,100),
    u           = c(1,0.775,0.5625,0.24),
    required    = FALSE,
    utility     = FALSE)

# define 1d interpolation end node for bed modification with
# other material
# (attribute "bedmodfract_percent" with levels from 0 to 100)

bedmod_other <-
  utility.endnode.intpol1d.create(
    name.node   = "bed modification other",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,100),
    u           = c(1,0.775,0.5625,0),
    required    = FALSE,
    utility     = FALSE)

# define combination end node for bed modification
# (attributes "bedmodtype_class" and "bedmodfract_percent")

bedmod <-
  utility.endnode.cond.create(
    name.node     = "bed modification",
    attrib.levels = data.frame(bedmodtype_class=
                               c("riprap","other")),
    nodes         = list(bedmod_riprap,bedmod_other),
    required      = FALSE,
    utility       = FALSE)

# define 1d interpolation end node for bank modification with
# permeable material
# (attribute "bankmodfract_percent" with levels from 0 to 100)

bankmod_perm <-
  utility.endnode.intpol1d.create(
    name.node   = "bank modification perm",
    name.attrib = "bankmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,60,100),
    u           = c(1,0.8667,0.675,0.4125,0.24),
    required    = FALSE,
```

```
    utility     = FALSE)

# define 1d interpolation end node for bank modification with
# impermeable material
# (attribute "bankmodfract_percent" with levels from 0 to 100)

bankmod_imperm <-
  utility.endnode.intpol1d.create(
    name.node   = "bank modification imperm",
    name.attrib = "bankmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,60,100),
    u           = c(1,0.775,0.5625,0.24,0),
    required    = FALSE,
    utility     = FALSE)

# define combination end node for bank modification
# (attributes "bankmodtype_class" and "bankmodfract_percent")

bankmod <-
  utility.endnode.cond.create(
    name.node     = "bank modification",
    attrib.levels = data.frame(bankmodtype_class=
                               c("perm","imperm")),
    nodes         = list(bankmod_perm,bankmod_imperm),
    required      = FALSE,
    utility       = FALSE)

# define 2d interpolation end node for riparian zone width
# (attributes "riparianzonewidth_m" and "riparianzonewidth_m")

riparzone_width <-
  utility.endnode.intpol2d.create(
    name.node   = "riparian zone width",
    name.attrib = c("riverbedwidth_m","riparianzonewidth_m"),
    ranges      = list(c(0,16),c(0,30)),
    isolines    = list(list(x=c(0,16),y=c(0,0)),
                       list(x=c(0,2,10,16),y=c(5,5,15,15)),
                       list(x=c(0,16),y=c(15,15)),
                       list(x=c(0,16),y=c(30,30))),
    u           = c(0.0,0.6,1.0,1.0),
    lead         = 1,
    utility      = FALSE)

# define discrete end node for riparian zone vegetation
# (attriute "riparianzoneveg_class" with levels "natural",
# "seminatural" and "artificial")

riparzone_veg <-
  utility.endnode.discrete.create(
    name.node     = "riparian zone veg.",
    attrib.levels = data.frame(riparianzoneveg_class=
                               c("natural","seminatural","artificial")),
```

```
    u             = c(1,0.5625,0),
    required      = FALSE,
    utility       = FALSE)

# define aggregation node for riparian zone

riparzone <-
  utility.aggregation.create(
    name.node = "riparian zone",
    nodes     = list(riparzone_width,riparzone_veg),
    name.fun  = "utility.aggregate.cobbdouglas",
    par       = c(1,1),
    required  = FALSE)

# define aggregation node for ecomorphological state

morphol <-
  utility.aggregation.create(
    name.node = "ecomorphology",
    nodes     = list(widthvar,bedmod,bankmod,riparzone),
    name.fun  = "utility.aggregate.mix",
    par       = c(0.25,0.25,0.25,0.25,0,0,1),
    names.par = c("w_widthvar","w_bedmod","w_bankmod","w_riparzone",
                  "w_add","w_min","w_cobbdouglas"),
    required  = TRUE)

# print individual definitions

print(widthvar)
print(bedmod)

# print all definitions

print(morphol)

# plot objectives hierarchy with attributes

plot(morphol)

# plot individual nodes:

plot(widthvar)
plot(widthvar,par=c(u_moderate=0.2))
plot(bedmod_other)
plot(bankmod)
#plot(riparzone_width)

# plot selected node definitions of a hierarchy

plot(morphol,type="nodes",nodes=c("width variability",
                                  "bed modification other",
                                  "bank modification"))
```

```
# evaluate value function for data sets and plot colored hierarchies
# and table

attrib_channelized <- data.frame(widthvariability_class = "none",
                                 bedmodtype_class       = "riprap",
                                 bedmodfract_percent    = 50,
                                 bankmodtype_class      = "imperm",
                                 bankmodfract_percent   = 70,
                                 riverbedwidth_m        = 10,
                                 riparianzonewidth_m    = 5,
                                 riparianzoneveg_class  = "seminatural")
attrib_rehab       <- data.frame(widthvariability_class = "high",
                                 bedmodtype_class       = "riprap",
                                 bedmodfract_percent    = 50,
                                 bankmodtype_class      = "imperm",
                                 bankmodfract_percent   = 20,
                                 riverbedwidth_m        = 15,
                                 riparianzonewidth_m    = 15,
                                 riparianzoneveg_class  = "natural")

res_channelized     <- evaluate(morphol,attrib=attrib_channelized)
res_channelized_add <- evaluate(morphol,attrib=attrib_channelized,
                                par=c(w_add=1,w_min=0,w_cobbdouglas=0))
res_rehab           <- evaluate(morphol,attrib=attrib_rehab)
res_both            <- rbind(res_channelized,res_rehab)
rownames(res_both)  <- c("channelized","rehabilitated")

plot(morphol,u=res_channelized)
plot(morphol,u=res_channelized_add)
plot(morphol,u=res_rehab)
plot(morphol,u=res_rehab,uref=res_channelized)
plot(morphol,u=res_both,type="table")

# consideration of uncertain attribute levels (higher uncertainty for
# predicted state after rehabilitation than for observed channelized state):

sampsize <- 2000

attrib_channelized_unc <- data.frame(
  widthvariability_class = rep("high",sampsize),
  bedmodtype_class       = rep("riprap",sampsize),
  bedmodfract_percent    = rnorm(sampsize,mean=50,sd=5),
  bankmodtype_class      = rep("imperm",sampsize),
  bankmodfract_percent   = rnorm(sampsize,mean=70,sd=5),
  riverbedwidth_m        = rep(10,sampsize),
  riparianzonewidth_m    = rep(5,sampsize),
  riparianzoneveg_class  = c("seminatural","artificial")[rbinom(sampsize,1,0.5)+1])

attrib_rehab_unc <- data.frame(
  widthvariability_class = c("moderate","high")[rbinom(sampsize,1,0.5)+1],
  bedmodtype_class       = rep("riprap",sampsize),
  bedmodfract_percent    = rnorm(sampsize,mean=50,sd=15),
  bankmodtype_class      = rep("imperm",sampsize),
```

```
  bankmodfract_percent   = rnorm(sampsize,mean=20,sd=5),
  riverbedwidth_m        = rnorm(sampsize,mean=10,sd=2),
  riparianzonewidth_m    = rnorm(sampsize,mean=10,sd=2),
  riparianzoneveg_class  = c("natural","seminatural")[rbinom(sampsize,1,0.5)+1])

res_channelized_unc <- evaluate(morphol,attrib=attrib_channelized_unc)
res_rehab_unc       <- evaluate(morphol,attrib=attrib_rehab_unc)

plot(morphol,u=res_channelized_unc)
plot(morphol,u=res_rehab_unc)
plot(morphol,u=res_rehab_unc,uref=res_channelized_unc)
```

`utility.calc.colors` *Color Scheme for Value Functions*

Description

Function to calculate a color scheme for value functions.

Usage

```
utility.calc.colors(n = 5)
```

Arguments

n	number of colors.

Details

For n = 5 this function produces the standard colors red, orange, yellow, green and blue as used in river assessment programs. These colors are provided in a lighter version to improve readability of black text in front of the colored background. For large values of n quasi-continuous transitions are defined between these colors. Any other vector of colors can be used by the plotting routines.

Value

Character vector of colors.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See
plot.utility.endnode.discrete
plot.utility.endnode.intpol1d
plot.utility.endnode.parfun1d
plot.utility.endnode.intpol2d
plot.utility.endnode.cond
plot.utility.aggregation
plot.utility.conversion.intpol
plot.utility.conversion.parfun
for the use of such color vectors in plotting functions and
utility.get.colors
for getting colors corresponding to specified values.

Examples

```
utility.calc.colors(5)
utility.calc.colors(100)
```

---

utility.conversion.intpol.create

Construct an interpolation conversion node

Description

Function to construct a node converting values into utilities by interpolation.

Usage

```
utility.conversion.intpol.create(name.node,
                                 node,
                                 x,
                                 u,
                                 names.x      = rep(NA, length(x)),
                                 names.u      = rep(NA, length(u)),
                                 required     = FALSE,
                                 col          = "black",
                                 shift.levels = 0)
```

Arguments

`name.node`	name of the node to be constructed as a character string.
`node`	value node that is to be converted into a utility node.
`x`	numeric vector of values for which the utility is known.
`u`	numeric vector of utilities corresponding to the values given in the previous argument `x`.
`names.x`	(optional) vector of character strings with names of the components of the numeric vector `x` specified above. Only required to provide access to the values through a named parameter vector.
`names.u`	(optional) vector of character strings with names of the components of the numeric vector `u` specified above. Only required to provide acces through a named parameter vector.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.conversion.intpol1` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.conversion.intpol`,
`summary.utility.conversion.intpol`,
`evaluate.utility.conversion.intpol` and
`plot.utility.conversion.intpol`.

Create other conversion nodes with `utility.conversion.parfun.create`. Create end nodes with

`utility.endnode.discrete.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.cond.create`, or
`utility.endnode.firstavail.create`.

Create aggregation nodes with

`utility.aggregation.create`.

---

utility.conversion.parfun.create

Construct a parametric function conversion node

Description

Function to construct a node converting values into utilities by a parametric function.

Usage

```
utility.conversion.parfun.create(name.node,
                                 node,
                                 name.fun,
                                 par,
                                 names.par    = rep(NA, length(par)),
                                 required     = FALSE,
                                 col          = "black",
                                 shift.levels = 0)
```

Arguments

`name.node`	name of the node to be constructed as a character string.
`node`	value node that is to be converted into a utility node.

`name.fun`	name of the parametric function to be evaluated as a character string. The parametric function must have the arguments `u` and `par` which pass a vector of values and a vector of parameters to the function, respectively. The function has to return a vector of corresponding utilities.
`par`	numeric vector of parameter values to be passed to the function specified unter `name.fun`.
`names.par`	(optional) vector of parameter names corresponging to the vector of values specified under `par`. Only required to provide access to the values through a named parameter vector.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.conversion.parfun` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.conversion.parfun`,
`summary.utility.conversion.parfun`,
`evaluate.utility.conversion.parfun` and

plot.utility.conversion.parfun.

Create other conversion nodes with utility.conversion.intpol.create. Create end nodes with

utility.endnode.discrete.create,
utility.endnode.parfun1d.create,
utility.endnode.intpol2d.create,
utility.endnode.parfun1d.create,
utility.endnode.cond.create, or
utility.endnode.firstavail.create.

Create aggregation nodes with

utility.aggregation.create.

---

utility.endnode.classcounts.create

Construct an end node that evaluates counts in different quality classes.

Description

Function to construct a node that evaluates counts in different quality classes by assigning a value to the highest class with counts > 0 and optionally increments this value with the counts in this and lower classses and given increments per count unit. The user can choose whether these increments can lead to a value higher than the basic value for counts in the next better class or if this value limits the potential increase.

Usage

```
utility.endnode.classcounts.create(name.node,        # character(1)
                                   name.attrib,      # character(n)
                                   u.max.inc,        # list (n) of vect (>=1)
                                   names.u.max.inc = list(),
                                   exceed.next     = TRUE,
                                   utility         = TRUE,
                                   required        = FALSE,
                                   col             = "black",
                                   shift.levels    = 0)
```

Arguments

name.node	name of the node to be constructed as a character string.
name.attrib	vector of names of attributes (counts in classes in decreasing order of value of the classes).
u.max.inc	list of vectors specifying the basic value and the increments for each class and classes of less value.

`names.u.max.inc`
: (optional) list of vectors of names of parameters u.max.inc.

`exceed.next`
: (optional) logical variable to indicate whether the level corresponding to the species classified at the next higher level can be exceeded with increments; default value is TRUE.

`utility`
: (optional) logical variable indicating if a value function (`FALSE`) or a utility function (`TRUE`) is created. Default value is `TRUE`.

`required`
: (optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.

`col`
: (optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.

`shift.levels`
: (optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.endnode.classcounts` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.endnode.classcounts`,
`summary.utility.endnode.classcounts`,
`evaluate.utility.endnode.classcounts` and
`plot.utility.endnode.classcounts`.

Create other end nodes with

utility.endnode.discrete.create,
utility.endnode.intpol1d.create,
utility.endnode.parfun1d.create,
utility.endnode.intpol2d.create, or
utility.endnode.cond.create,
utility.endnode.firstavail.create,

Create other types of nodes with

utility.aggregation.create,
utility.conversion.intpol.create, or
utility.conversion.parfun.create.

---

utility.endnode.cond.create

Construct a conditional end node

Description

Function to construct a node that makes a choice between given end nodes based on the levels of discrete attributes.

Usage

```
utility.endnode.cond.create(name.node,
                            attrib.levels,
                            nodes,
                            utility      = TRUE,
                            required     = FALSE,
                            col          = "black",
                            shift.levels = 0)
```

Arguments

name.node	name of the node to be constructed as a character string.
attrib.levels	data frame with attribute names as column names and all discrete attribute level combinations in the rows. This may be a dependence on any number of attributes. As combinatorics can lead to a very large number of possible combinations, the node should not depend on a too large number of attributes, in particular if each attribute has many different levels expressed by numbers or character strings.
nodes	list of the length of the number of columns of the data frame specifed as argument attrib.levels above containing the nodes to be associated with the attribute level combinations specified in the rows of attrib.levels.

`utility`	(optional) logical variable indicating if a value function (`FALSE`) or a utility function (`TRUE`) is created. Default value is `TRUE`.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.endnode.cond` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.endnode.cond`,
`summary.utility.endnode.cond`,
`evaluate.utility.endnode.cond` and
`plot.utility.endnode.cond`.

Create other end nodes with

`utility.endnode.discrete.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,

utility.endnode.parfun1d.create, or
utility.endnode.firstavail.create.

Create other types of nodes with

utility.aggregation.create,
utility.conversion.intpol.create, or
utility.conversion.parfun.create.

Examples

```
bedmod_riprap <-
  utility.endnode.intpol1d.create(
    name.node   = "bed modification riprap",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,100),
    u           = c(1,0.775,0.5625,0.24),
    required    = FALSE,
    utility     = FALSE)

bedmod_other <-
  utility.endnode.intpol1d.create(
    name.node   = "bed modification other",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,100),
    u           = c(1,0.775,0.5625,0),
    required    = FALSE,
    utility     = FALSE)

bedmod <-
  utility.endnode.cond.create(
    name.node     = "bed modification",
    attrib.levels = data.frame(bedmodtype_class=
                               c("riprap","other")),
    nodes         = list(bedmod_riprap,bedmod_other),
    required      = FALSE,
    utility       = FALSE)

print(bedmod)
plot(bedmod)
```

---

utility.endnode.discrete.create

Construct a discrete value or utility end node

Description

Function to construct a discrete value or utility end node.

Usage

```
utility.endnode.discrete.create(name.node,
                                attrib.levels,
                                u,
                                names.u      = rep(NA, length(u)),
                                utility      = TRUE,
                                required     = FALSE,
                                col          = "black",
                                shift.levels = 0)
```

Arguments

`name.node`	name of the node to be constructed as a character string.
`attrib.levels`	data frame with attribute names as column names and all discrete attribute level combinations in the rows. This may be a dependence on any number of attributes. As combinatorics can lead to a very large number of possible combinations, the node should not depend on a too large number of attributes, in particular if each attribute has many different levels expressed by numbers or character strings.
`u`	numeric vector of the length of the number of columns of the data frame specifed as argument `attrib.levels` above specifying the values or utilities corresponding to the rows of `attrib.levels`.
`names.u`	(optional) vector of character strings with names of the components of the numeric vector `u` specified above. Only required to provide access to the values through a named parameter vector.
`utility`	(optional) logical variable indicating if a value function (`FALSE`) or a utility function (`TRUE`) is created. Default value is `TRUE`.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.endnode.discrete` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.endnode.discrete`,
`summary.utility.endnode.discrete`,
`evaluate.utility.endnode.discrete` and
`plot.utility.endnode.discrete`.

Create other end nodes with

`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`, or
`utility.endnode.firstavail.create`.

Create other types of nodes with

`utility.aggregation.create`,
`utility.conversion.intpol.create`, or
`utility.conversion.parfun.create`.

Examples

```
widthvar <-
  utility.endnode.discrete.create(
    name.node     = "width variability",
    attrib.levels = data.frame(widthvariability_class=
                               c("high","moderate","none")),
    u             = c(1,0.4125,0),
    names.u       = c("u.high","u_moderate","u.none"),
    required      = FALSE,
    utility       = FALSE)

print(widthvar)
```

```
plot(widthvar)
```

`utility.endnode.firstavail.create`
Construct an end node to get the results of the first available sub-node

---

Description

Function to construct a node that returns the results of the first sub-node for which results are available.

Usage

```
utility.endnode.firstavail.create(name.node,
                                  nodes,
                                  utility      = TRUE,
                                  required     = FALSE,
                                  col          = "black",
                                  shift.levels = 0)
```

Arguments

`name.node`	name of the node to be constructed as a character string.
`nodes`	list of nodes to be tried.
`utility`	(optional) logical variable indicating if a value function (`FALSE`) or a utility function (`TRUE`) is created. Default value is `TRUE`.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.endnode.firstavail` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.endnode.firstavail`,
`summary.utility.endnode.firstavail`,
`evaluate.utility.endnode.firstavail` and
`plot.utility.endnode.firstavail`.

Create other end nodes with

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`, or
`utility.endnode.cond.create`,

Create other types of nodes with

`utility.aggregation.create`,
`utility.conversion.intpol.create`, or
`utility.conversion.parfun.create`.

`utility.endnode.intpol1d.create`
Construct a single-attribute interpolation end node

Description

Function to construct a single-attribute interpolation end node.

Usage

```
utility.endnode.intpol1d.create(name.node,
                                name.attrib,
                                range,
                                x,
                                u,
                                names.x      = rep(NA, length(x)),
                                names.u      = rep(NA, length(u)),
                                utility      = TRUE,
                                required     = FALSE,
                                col          = "black",
                                shift.levels = 0)
```

Arguments

`name.node`	name of the node to be constructed as a character string.
`name.attrib`	name of the attribute on which the value or utility function depends as a character string.
`range`	numeric vector with two components specifying the minimum and the maximum of the attribute range.
`x`	numeric vector of attribute values for which the value or utility is known.
`u`	numeric vector of values or utilities corresponding to the attribute values given in the previous argument `x`.
`names.x`	(optional) vector of character strings with names of the components of the numeric vector `x` specified above. Only required to provide access to the values through a named parameter vector.
`names.u`	(optional) vector of character strings with names of the components of the numeric vector `u` specified above. Only required to provide acces through a named parameter vector.
`utility`	(optional) logical variable indicating if a value function (`FALSE`) or a utility function (`TRUE`) is created. Default value is `TRUE`.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.endnode.intpol1d` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.endnode.intpol1d`,
`summary.utility.endnode.intpol1d`,
`evaluate.utility.endnode.intpol1d` and
`plot.utility.endnode.intpol1d`.

Create other end nodes with

`utility.endnode.discrete.create`,
`utility.endnode.parfun1d.create`,
`utility.endnode.intpol2d.create`,
`utility.endnode.cond.create`, or
`utility.endnode.firstavail.create`.

Create other types of nodes with

`utility.aggregation.create`,
`utility.conversion.intpol.create`, or
`utility.conversion.parfun.create`.

Examples

```
bedmod_other <-
  utility.endnode.intpol1d.create(
    name.node   = "bed modification other",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    x           = c(0,10,30,100),
    u           = c(1,0.775,0.5625,0),
```

```
        required    = FALSE,
        utility     = FALSE)

    print(bedmod_other)
    plot(bedmod_other)
```

---

utility.endnode.intpol2d.create

Construct a two-attribute interpolation end node

Description

Function to construct a two-attribute interpolation end node.

Usage

```
utility.endnode.intpol2d.create(name.node,
                                name.attrib,
                                ranges,
                                isolines,
                                u,
                                names.u      = rep(NA, length(u)),
                                lead         = 0,
                                utility      = TRUE,
                                required     = FALSE,
                                col          = "black",
                                shift.levels = 0)
```

Arguments

name.node	name of the node to be constructed as a character string.
name.attrib	names of the attributes on which the value or utility function depends as a vector of two character strings.
ranges	list of two numeric vectors with two components each specifying the minimum and the maximum of the range of the corresponding attribute.
isolines	list of isoline definitions. Each definition consists of a list with elements x and y that each represents a numeric vector of x- (=first attribute) and y- (second attribute) values to characterize the shape of the isoline.
u	numeric vector of the same length as the outer list of the argument isolines specifying the corresponding values or utilities.
names.u	(optional) vector of character strings with names of the components of the numeric vector u specified above. Only required to provide acces through a named parameter vector.
lead	numeric value specifying which variable is the lead variable for interpolation. 1 indicates linear interpolation between isolines along lines with constant value of the first attribute, 2 along lines with constant values of the second attribute, and zero indicates to take the average of these two interpolation schemes.

`utility`	(optional) logical variable indicating if a value function (`FALSE`) or a utility function (`TRUE`) is created. Default value is `TRUE`.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.endnode.intpol2d` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

`print.utility.endnode.intpol2d`,
`summary.utility.endnode.intpol2d`,
`evaluate.utility.endnode.intpol2d` and
`plot.utility.endnode.intpol2d`.

Create other end nodes with

`utility.endnode.discrete.create`,
`utility.endnode.intpol1d.create`,
`utility.endnode.parfun1d.create`,

utility.endnode.cond.create, or
utility.endnode.firstavail.create.

Create other types of nodes with

utility.aggregation.create,
utility.conversion.intpol.create, or
utility.conversion.parfun.create.

Examples

```
riparzone_width <-
  utility.endnode.intpol2d.create(
    name.node   = "riparian zone width",
    name.attrib = c("riverbedwidth_m","riparianzonewidth_m"),
    ranges      = list(c(0,16),c(0,30)),
    isolines    = list(list(x=c(0,16),y=c(0,0)),
                       list(x=c(0,2,10,16),y=c(5,5,15,15)),
                       list(x=c(0,16),y=c(15,15)),
                       list(x=c(0,16),y=c(30,30))),
    u           = c(0.0,0.6,1.0,1.0),
    lead         = 1,
    utility      = FALSE)

print(riparzone_width)
plot(riparzone_width)
```

---

utility.endnode.parfun1d.create
Construct a single-attribute parametric function end node

Description

Function to construct a single-attribute parametric function end node.

Usage

```
utility.endnode.parfun1d.create(name.node,
                                name.attrib,
                                range,
                                name.fun,
                                par,
                                names.par    = rep(NA, length(par)),
                                utility      = TRUE,
                                required     = FALSE,
                                col          = "black",
                                shift.levels = 0)
```

Arguments

`name.node`	name of the node to be constructed as a character string.
`name.attrib`	name of the attribute on which the value or utility function depends as a character string.
`range`	numeric vector with two components specifying the minimum and the maximum of the attribute range.
`name.fun`	name of the parametric function to be evaluated as a character string. The parametric function must have the arguments `attrib` and `par` which pass a vector of attribute levels and a vector of parameters to the function, respectively. The function has to return a vector of corresponding values or utilities.
`par`	numeric vector of parameter values to be passed to the function specified unter `name.fun`.
`names.par`	(optional) vector of parameter names corresponging to the vector of values specified under `par`. Only required to provide access to the values through a named parameter vector.
`utility`	(optional) logical variable indicating if a value function (`FALSE`) or a utility function (`TRUE`) is created. Default value is `TRUE`.
`required`	(optional) logical variable indicating if the value of this node is required for aggregation at the next higher level. If this variable is `TRUE`, aggregation at the next higher level is not possible if this node returns NA. Default value is `FALSE`.
`col`	(optional) color used for plotting the bounding box of the node in the objective hierarchy. Default value is `"black"`.
`shift.levels`	(optional) number of hierarchical levels by which the node in the objective hierarchy is shifted to make a branch fit better to other branches. Default value is `0`.

Value

The function returns the created object of type `utility.endnode.parfun1d` with the properties specified in the arguments of the function.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Tradeoffs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

Print, evaluate and plot the node with

print.utility.endnode.parfun1d,
summary.utility.endnode.parfun1d,
evaluate.utility.endnode.parfun1d and
plot.utility.endnode.parfun1d.

Create other end nodes with

utility.endnode.discrete.create,
utility.endnode.intpol1d.create,
utility.endnode.intpol2d.create,
utility.endnode.cond.create, or
utility.endnode.firstavail.create.

Create other types of nodes with

utility.aggregation.create,
utility.conversion.intpol.create, or
utility.conversion.parfun.create.

Examples

```
bedmod_other <-
  utility.endnode.parfun1d.create(
    name.node   = "bed modification other",
    name.attrib = "bedmodfract_percent",
    range       = c(0,100),
    name.fun    = "utility.fun.exp",
    par         = c(-1,100,0),
    required    = FALSE,
    utility     = FALSE)

print(bedmod_other)
plot(bedmod_other)
```

---

utility.fun.exp *Exponential function for value or utility functions*

Description

Exponential function for value or utility functions.

Usage

```
utility.fun.exp(attrib, par)
```

Arguments

`attrib`	vector of attribute levels to calculate corresponding value or utility.
`par`	Vector of parameters: `par[1]`: absolute risk aversion `par[2]`: minimum of attribute range (default = 0) `par[3]`: maximum of attribute range (default = 1)

Details

The function evaluates the expression
`(1-exp(-par[1]*(a-par[2])/(par[3]-par[2])))/(1-exp(-par[1]))`.

Value

Vector of values or utilities corresponding to the attributes passed by argument `a`

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See the node constructors
`utility.endnode.intpol1d.create` and `utility.conversion.intpol.create`
in which this function can be used.

Examples

```
utility.fun.exp(0:10/10,par=c(2,0,1))
```

`utility.get.attrib.names` *Get Names of Attributes Used by a Value Functions*

Description

Function to get the names of the attributes used by a given value function.

Usage

```
utility.get.attrib.names(node)
```

Arguments

`node`	node of an objectives hierarchy with its associated value function (object of class `utility`).

Value

Character vector of names of attributes.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

`utility.get.colors` *Get Color Corresponding to Specified Value Levels*

Description

Function to get the colors from a given color scheme at specific value levels.

Usage

```
utility.get.colors(u,col=utility.calc.colors())
```

Arguments

`u`	value level representing the evaluation ot a value function (this value level has to be between zero and unity).
`col`	color scheme (vector of colors to be used for a division of the interval between zero and unity into equal intervals)).

Value

Character vector of colors.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

See Also

See
`utility.calc.colors`

Examples

```
utility.get.colors(c(0,0.5,1))
```

`utility.structure` *Extract Structure of Objectives Hierarchy*

Description

Function to extract the structure of an objectives hierarchy.

Usage

```
utility.structure(node)
```

Arguments

`node`	object containing the utility or value function.

Value

Data frame containing structural information of the objectives hierarchy.

Author(s)

Peter Reichert <peter.reichert@emeriti.eawag.ch>

References

Short description of the package:

Reichert, P., Schuwirth, N. and Langhans, S., Constructing, evaluating and visualizing value and utility functions for decision support, Environmental Modelling & Software 46, 283-291, 2013.

Textbooks on the use of utility and value functions in decision analysis:

Keeney, R. L. and Raiffa, H. Decisions with Multiple Objectives - Preferences and Value Trade-offs. John Wiley & Sons, 1976.

Eisenfuehr, F., Weber, M. and Langer, T., Rational Decision Making, Springer, Berlin, 2010.

Index

∗ **decision analysis**
utility-package, 3
∗ **objectives hierarchy**
utility-package, 3
∗ **utility function**
utility-package, 3
∗ **value function**
utility-package, 3

evaluate, 11
evaluate.utility.aggregation, *6*, 13, *29*, *31*, *32*, *35*, *142*
evaluate.utility.conversion.intpol, *6*, 14, *34*, *150*
evaluate.utility.conversion.parfun, *6*, 16, *37*, *151*
evaluate.utility.endnode.classcounts, *6*, 17, *39*, *153*
evaluate.utility.endnode.cond, *6*, 19, *40*, *155*
evaluate.utility.endnode.discrete, *6*, 20, *42*, *158*
evaluate.utility.endnode.firstavail, *6*, 22, *44*, *160*
evaluate.utility.endnode.intpol1d, *6*, 23, *45*, *47*, *162*
evaluate.utility.endnode.intpol2d, *6*, 25, *164*
evaluate.utility.endnode.parfun1d, *6*, 26, *49*, *167*

plot.utility.aggregation, *5*, *6*, *14*, *17*, *21*, *24*, *26*, *27*, 28, *33*, *37*, *142*, *148*
plot.utility.conversion.intpol, *5*, *6*, *15*, *30*, 31, *37*, *148*, *150*
plot.utility.conversion.parfun, *5*, *6*, *30*, *33*, 34, *148*, *152*
plot.utility.endnode.classcounts, *5*, *18*, 37, *153*
plot.utility.endnode.cond, *5*, *20*, *38*, 39, *42*, *43*, *45*, *47*, *48*, *148*, *155*
plot.utility.endnode.discrete, *5*, *38*, *40*, 41, *43*, *45*, *47*, *48*, *148*, *158*
plot.utility.endnode.firstavail, *5*, *23*, *38*, *40*, 42, *42*, *45*, *47*, *48*, *160*
plot.utility.endnode.intpol1d, *5*, *38*, *40*, *41*, *43*, 44, *48*, *148*, *162*
plot.utility.endnode.intpol2d, *5*, *38*, *40*, *41*, *43*, *45*, 46, *47*, *48*, *148*, *164*
plot.utility.endnode.parfun1d, *5*, *38*, *40*, *41*, *43*, *45*, *47*, 48, *148*, *167*
print.utility.aggregation, *5*, *14*, *17*, *21*, *24*, *26*, *27*, 49, *142*
print.utility.conversion.intpol, *5*, *15*, 50, *150*
print.utility.conversion.parfun, *5*, 51, *151*
print.utility.endnode.classcounts, *5*, *18*, 52, *153*
print.utility.endnode.cond, *5*, *20*, 53, *155*
print.utility.endnode.discrete, *5*, 54, *158*
print.utility.endnode.firstavail, *5*, *23*, 55, *160*
print.utility.endnode.intpol1d, *5*, 56, *162*
print.utility.endnode.intpol2d, *5*, 57, *164*
print.utility.endnode.parfun1d, *5*, 58, *167*

summary.utility.aggregation, *5*, *14*, *17*, *21*, *24*, *26*, *27*, 59, *142*
summary.utility.conversion.intpol, *5*, *15*, 60, *150*
summary.utility.conversion.parfun, *5*, 61, *151*

summary.utility.endnode.classcounts, *5*, *18*, 62, *153*
summary.utility.endnode.cond, *5*, *20*, 63, *155*
summary.utility.endnode.discrete, *5*, 64, *158*
summary.utility.endnode.firstavail, *5*, *23*, 65, *160*
summary.utility.endnode.intpol1d, *5*, 66, *162*
summary.utility.endnode.intpol2d, *5*, 67, *164*
summary.utility.endnode.parfun1d, *5*, 68, *167*

updatepar, 69
updatepar.utility.aggregation, *70*, 71, *73–77*, *79–82*
updatepar.utility.conversion.intpol, *70*, 72, *72*, *74–77*, *79–82*
updatepar.utility.conversion.parfun, *70*, *72*, 73, *73*, *75–77*, *79–82*
updatepar.utility.endnode.classcounts, 74
updatepar.utility.endnode.cond, *70*, *72–74*, 75, *75*, *77*, *79–82*
updatepar.utility.endnode.discrete, *70*, *71*, *73–75*, 76, *76*, *78*, *80–82*
updatepar.utility.endnode.firstavail, *75–77*, 78, *80–82*
updatepar.utility.endnode.intpol1d, *70*, *71*, *73–78*, 79, *81*, *82*
updatepar.utility.endnode.intpol2d, *70*, *72–77*, *79*, 80, *80*, *82*
updatepar.utility.endnode.parfun1d, *70*, *72–77*, *79*, *80*, 81, *81*
utility (utility-package), 3
utility-package, 3
utility.aggregate.add, 82, *84*, *85*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *116*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.addmin, *84*, 85, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *141*
utility.aggregate.addpower, *84*, *87*, 88, *90*, *93*, *97*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *141*
utility.aggregate.addsplitpower, *84*, *87*, *90*, 91, *93*, *97*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *141*
utility.aggregate.bonusmalus, *84*, *87*, *90*, *93*, 94, *97*, *99*, *102*, *105*, *108*, *111*, *113*, *116*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *141*
utility.aggregate.cobbdouglas, *84*, *87*, *90*, *93*, *96*, 97, *99*, *101*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.geo, *84*, *87*, *90*, *93*, *96*, *98*, *99*, 100, *102*, *103*, *105*, *108*, *111*, *113*, *115*, *116*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.geooff, *84*, *87*, *90*, *93*, *96*, *97*, *99*, *100*, *102*, 103, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.harmo, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, 106, *108*, *109*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.harmooff, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *106*, *108*, 109, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.max, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, 112, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.min, *84*, *85*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, 114, *115*, *116*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.mix, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, 116, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.mult, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, 118, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, *140*
utility.aggregate.revaddpower, *84*, *87*, *90*, *93*, *97*, *99*, *102*, *105*, *108*, *111*,

113, *115*, *118*, *121*, 122, *124*, *127*, *130*, *133*, *136*, *139*, *141*
utility.aggregate.revaddsplitpower, *84*, *87*, *90*, *93*, *97*, *99*, *102*, *105*, *108*, *111*, *113*, *116*, *118*, *121*, *124*, 125, *127*, *130*, *133*, *136*, *139*, *141*
utility.aggregate.revgeo, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, 128, *130, 131*, *133*, *136*, *139, 140*
utility.aggregate.revgeooff, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127, 128*, *130*, 131, *133*, *136*, *139, 140*
utility.aggregate.revharmo, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, 134, *136, 137*, *139, 140*
utility.aggregate.revharmooff, *84*, *87*, *90*, *93*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133, 134*, *136*, 137, *139, 140*
utility.aggregation.create, *4*, *12*, *14*, *16, 17*, *19–28*, *31*, *50*, *60*, *70, 71*, *84*, *87*, *90*, *93, 94*, *96*, *99*, *102*, *105*, *108*, *111*, *113*, *115*, *118*, *121*, *124*, *127*, *130*, *133*, *136*, *139*, 140, *150*, *152*, *154*, *156*, *158*, *160*, *162*, *165*, *167*
utility.calc.colors, *31*, *34*, *37*, *39, 40*, *42*, *44, 45*, *47*, *49*, 147, *170*
utility.conversion.intpol.create, *4*, *12*, *14, 15*, *17*, *19, 20*, *22, 23*, *25, 26*, *28*, *34*, *51*, *61*, *70*, *142*, 148, *152*, *154*, *156*, *158*, *160*, *162*, *165*, *167, 168*
utility.conversion.parfun.create, *5*, *12*, *14*, *16*, *19, 20*, *22, 23*, *25, 26*, *28*, *37*, *52*, *62*, *70*, *142*, 150, *150*, *154*, *156*, *158*, *160*, *162*, *165*, *167*
utility.endnode.classcounts.create, *4*, *18*, *39*, *53*, *63*, *75*, 152
utility.endnode.cond.create, *4*, *12*, *14*, *16, 17*, *19–21*, *23, 24*, *26, 27*, *40*, *54*, *64*, *70*, *76*, *142*, *150*, *152*, 154, *154*, *158*, *160*, *162*, *165*, *167*
utility.endnode.discrete.create, *4*, *12*, *14, 15*, *17, 18*, *20*, *23, 24*, *26, 27*, *42*, *55*, *65*, *70*, *142*, *150*, *152*, *154, 155*, 156, *160*, *162*, *164*, *167*
utility.endnode.firstavail.create, *4*, *14*, *16, 17*, *19–21*, *23*, *25–27*, *44*, *56*, *66*, *78*, *142*, *150*, *152*, *154*, *156*, *158*, 159, *162*, *165*, *167*
utility.endnode.intpol1d.create, *4*, *12*, *14, 15*, *17, 18*, *20, 21*, *23*, *26, 27*, *45*, *47*, *57*, *67*, *70*, *142*, *154*, *158*, 160, *160*, *164*, *167, 168*
utility.endnode.intpol2d.create, *4*, *12*, *14*, *16–18*, *20, 21*, *23, 24*, *27*, *58*, *68*, *70*, *142*, *150*, *152*, *154, 155*, *158*, *160*, *162*, 163, *167*
utility.endnode.parfun1d.create, *4*, *12*, *14, 15*, *17, 18*, *20, 21*, *23, 24*, *26*, *49*, *59*, *69, 70*, *73, 74*, *77*, *80–82*, *142*, *150*, *152*, *154–156*, *158*, *160*, *162*, *164*, 165
utility.fun.exp, 167
utility.get.attrib.names, 169
utility.get.colors, *31*, *34*, *37*, *39, 40*, *42*, *44, 45*, *47*, *49*, *148*, 170
utility.structure, 171