Package 'decisionSupport'

November 3, 2014

Type Package

Title Quantitative Support of Decision Making under Uncertainty

Version 1.073

Date 2014-10-22

Author Lutz Goehring, Eike Luedeling (-#ToDo: Order?)

Maintainer ToDo: Who shall do it: Luedeling, Eike (ICRAF) <E.Luedeling@cgiar.org>

Description The package is subdivided in the Monte Carlo simulation and analysis, the Value of Information Analysis (VIA) and Partial Least Squares (PLS) analysis and Variable Importance in Projection (VIP).

License -#ToDo: What license is it under?

Depends rriskDistributions

Suggests testthat

R topics documented:

`barplot_vip` *Generate barplots of VIP results.*

Description

Generate barplots of VIP results.

Usage

```
barplot_vip(object, ...)
```

Arguments

`object`	ToDo
`...`	Todo

`decisionSupport` *Quantitative Support of Decision Making under Uncertainty*

Description

The decisionSupport package supports the Monte Carlo simulation of different decisions.

Details

The package is subdivided in the Monte Carlo simulation and analysis, the Value of Information Analysis (VIA) and Partial Least Squares (PLS) analysis and Variable Importance in Projection (VIP).

References

Hubbarrd, Douglas W., How to Measure Anything? - Finding the Value of "Intangibles" in Business, John Wiley & Sons, Hoboken, New Jersey, 2014, 3rd Ed, http://www.howtomeasureanything.com/.

`estimate_read_csv` *Read an Estimate from CSV - File.*

Description

This function reads an estimate from the specified csv files. In this context, an estimate of a variable is defined by its distribution type, its 90%-confidence interval `[lower,upper]` and its correlation to other variables. #ToDo: Implement characterization of distribution by mean and sd. Eventually, also by other quantiles.

Usage

```
estimate_read_csv(filename, strip.white = TRUE, ...)
```

Arguments

`filename`	Filename or estimate object.
`strip.white`	logical. Allows the stripping of leading and trailing white space from unquoted character fields (numeric fields are always stripped). See `scan` for further details (including the exact meaning of 'white space'), remembering that the columns may include the row names.
`...`	Further parameters to be passed to `read.csv`.

Details

```
list(basic=data.frame(description, lower, median, upper, distribution, variable, start, end, indic
```

Value

An object of type `estimate`.

CSV input file structures

The estimate is read from one or two csv files: the basic csv file which is mandatory and the correlation csv file which is optional. The basic csv file contains the definition of the distribution of all variables ignoring potential correlations. The correlation csv file only defines correlations.

The structure of the basic input file (mandatory): File name structure: `<basic-filename>.csv` Mandatory columns:

Column name	R-type	Explanation
`lower`	`numeric`	ToDo
`upper`	`numeric`	ToDo
`distribution`	`character`	ToDo
`variable`	`character`	ToDo

Optional columns:

Column name	R-type	Explanation
`description`	`character`	ToDo
`median`	`numeric`	ToDo

start	integer	ToDo
end	integer	ToDo
indicator	logical	ToDo

Columns without names are ignored. Rows where the variable field is empty are also dropped.

The structure of the correlation file (optional): File name structure: <basic-filename>_cor.csv Columns and rows are named by the corresponding variables. Only those variables need to be present which are correlated with others. The element ["rowname","columnname"] contains the correlation between the variables rowname and columnname. Uncorrelated elements can be left empty, i.e. as NA, or defined as 0. The element ["name","name"] has to be set to 1. The matrix must be given in symmetric form.

See Also

read.csv

---

mcSimulation *Perform a Monte Carlo Simulation.*

Description

This method solves the following problem. Given a multivariate random variable $x = (x_1, \ldots, x_k)$ with joint probability distribution P, i.e.

$$x \sim P.$$

Then the continuous function

$$f : R^k \to R^l, y = f(x)$$

defines another random variable with distribution

$$y \sim f(P).$$

Given a probability density ρ of x that defines P the problem is the determination of the probability density ϕ that defines $f(P)$. This method samples the probability density ϕ of y by Monte Carlo simulation.

Usage

```
mcSimulation(estimate, model_function, ..., numberOfSimulations, randomMethod,
  functionSyntax = "data.frameNames")
```

Arguments

estimate	Filename or estimate object representing the joint probability distribution of the input variables.
model_function	A numeric function; The function that describes the value of a certain project.
...	Optional arguments of model_function.
numberOfSimulations	The number of Monte Carlo simulations to be run.
randomMethod	character. The method to be used to sample the distribution representing the input estimate.
functionSyntax	character. The syntax which has to be used to implement the model function. Possible values are globalNames, data.frameNames or matrixNames. Details are given below.

Details

If `functionSyntax="globalNames"`, the variable names used in the definition of `model_function` have to be defined globally. `model_function` has to be of the form `function(x,varnames)`. If `functionSyntax="data.frameNames"`, the model function is constructed, e.g. like this:
`profit<-function(x){ x["revenue"]-x["costs"] }` or like this (ToDo: (i) check if both are allowed and (ii) which one is better?): `profit<-function(x){ x$revenue-x$costs }` If `functionSyntax="matrixNames"`, the model function is constructed, e.g. like this:
`profit<-function(x){ x[,"revenue"]-x[,"costs"] }`

Value

An object of class `mcSimulation`.

`phi`	an l-variate probability distribution
`x`	a dataframe containing the sampled $x-$ values
`y`	a dataframe containing the simulated $y-$ values

Examples

```
profit<-function(x){
   x$revenue-x$costs
 }
 variable=c("revenue","costs")
 distribution=c("norm","norm")
 lower=c(10000,  5000)
 upper=c(100000, 50000)
 base=data.frame(distribution=distribution,lower=lower,upper=upper,row.names=variable)
 estimate=list(base=base, correlation_matrix="" )
 class(estimate)<-"estimate"
 predictionProfit<-mcSimulation( estimate=estimate,
                                 model_function=profit,
                                 numberOfSimulations=1000,
                                 functionSyntax="data.frameNames")
 print(summary(predictionProfit))
 plot(predictionProfit)
```

`NPV` *Net Present Value (NPV).*

Description

Net Present Value (NPV).

Usage

```
NPV(x, discount_rate)
```

Arguments

`x`	ToDo
`discount_rate`	ToDo

`plot.mcSimulation` *Plot results of a Monte Carlo Simulation.*

Description

This function plots results of `mcSimulation`.

Usage

```
## S3 method for class 'mcSimulation'
plot(x, ...)
```

Arguments

`x`	An object of class `mcSimulation`.
`...`	Further arguments #ToDo

`plot.plsr.mcSimulation`
Plot results of the PLSR for Monte Carlo.

Description

This function plots results of `plsr.mcSimulation`.

Usage

```
## S3 method for class 'plsr.mcSimulation'
plot(x, ...)
```

Arguments

`x`	An object of class `plsr.mcSimulation`.
`...`	Further arguments #ToDo

`plot.via` *Plot results of a Value of Information Analysis.*

Description

This function plots results of `via`.

Usage

```
## S3 method for class 'via'
plot(x, ...)
```

Arguments

`x`	An object of class `via`.
`...`	Further arguments #ToDo

plsr.mcSimulation *Partial Least Squares of a Monte Carlo Simulation.*

Description

Perform a Partial Least Squares regression of a Monte Carlo simulation.

Usage

```
plsr.mcSimulation(x, ...)
```

Arguments

x	Todo
...	Todo

print.mcSimulation *Print Basic Results from Monte Carlo Simulation.*

Description

This function prints basic results from Monte Carlo simulation and returns it invisible.

Usage

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

Arguments

x	An object of class mcSimulation.
...	Further arguments #ToDo

print.plsr.mcSimulation *Print Basic Results of the PLSR for Monte Carlo.*

Description

This function prints basic results of the PLSR for Monte Carlo and returns it invisible.

Usage

```
## S3 method for class 'plsr.mcSimulation'
print(x, ...)
```

Arguments

x	An object of class plsr.mcSimulation.
...	Further arguments #ToDo

`print.summary.mcSimulation` *Print the Summary of a Monte Carlo Simulation.*

Description

This function prints the summary of of `mcSimulation` obtained by `summary.mcSimulation`.

Usage

```
## S3 method for class 'summary.mcSimulation'
print(x, ...)
```

Arguments

`x`	An object of class `mcSimulation`.
`...`	Further arguments #ToDo

`print.summary.plsr.mcSimulation` *Print the Summary of of the PLSR for Monte Carlo.*

Description

This function prints the summary of of `plsr.mcSimulation` obtained by `summary.plsr.mcSimulation`.

Usage

```
## S3 method for class 'summary.plsr.mcSimulation'
print(x, ...)
```

Arguments

`x`	An object of class `plsr.mcSimulation`.
`...`	Further arguments #ToDo

print.summary.via *Print the Summary of a Value of Information Analysis.*

Description

This function prints the summary of of `via` obtained by `summary.via`.

Usage

```
## S3 method for class 'summary.via'
print(x, ...)
```

Arguments

`x`	An object of class `via`.
`...`	Further arguments #ToDo

print.via *Print Basic Results of the Value of Information Analysis.*

Description

This function prints basic results of the Value of Information Analysis and returns it invisible.

Usage

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

Arguments

`x`	An object of class `via`.
`...`	Further arguments #ToDo

random *Generate random numbers for a certain probability distribution.*

Description

This function generates multivariate random numbers for general multivariate distributions.

Usage

```
random(rho, n, method, ...)
```

Arguments

rho	Distribution to be randomly sampled.
n	Number of generated observations.
method	Particular method to be used for random number generation.
...	Optional arguments to be passed to the particular random number generating function.

`random.default` *Generate random numbers based on the first two moments of a certain probability distribution.*

Description

This function generates random numbers for general multivariate distributions that can be characterized by the joint first two moments, viz. the mean and covariance.

Usage

```
## Default S3 method:
random(rho = list(distribution_type, mean, sd), n, method,
  ...)
```

Arguments

rho	list; Distribution to be randomly sampled.
n	Number of generated observations
method	Particular method to be used for random number generation.
...	Optional arguments to be passed to the particular random number generating function.

`random.estimate` *Generate random numbers based on the first two moments of a certain probability distribution.*

Description

This function generates random numbers for general multivariate distributions that can be characterized by the joint first two moments, viz. the mean and covariance.

Usage

```
## S3 method for class 'estimate'
random(rho, n, method, ...)
```

Arguments

rho	estimate object; Multivariate distribution to be randomly sampled.
n	Number of generated observations
method	Particular method to be used for random number generation.
...	Optional arguments to be passed to the particular random number generating function.

---

random_estimate_1d *Generate univariate random numbers based on an estimate.*

Description

This function generates random numbers for general univariate distributions.

Usage

```
random_estimate_1d(rho, n, method, ...)
```

Arguments

rho	estimate object; Univariate distribution to be randomly sampled.
n	Number of generated observations
method	Particular method to be used for random number generation.
...	Optional arguments to be passed to the particular random number generating function.

---

rdist90ci *Generate univariate random numbers based on the 90%-confidence interval.*

Description

This function generates random numbers for general univariate distributions based on the 90% confidence interval.

Usage

```
rdist90ci(distribution, n, lower, upper)
```

Arguments

distribution	character; A character string that defines the univariate distribution to be randomly sampled.
n	Number of generated observations.
lower	numeric; lower bound of the 90% confidence intervall.
upper	numeric; upper bound of the 90% confidence intervall.

Details

The follwing table shows the available distributions and their identification as a character string:

Distribution encoding	Distribution
constant	ToDo
normal	ToDo
pos_normal	ToDo
normal_0_1	ToDo
poisson	ToDo
binomial	ToDo
uniform	ToDo
lognorm	ToDo
lognorm_lim2	ToDo

`rdistq_fit` *Generate univariate random numbers based on quantiles.*

Description

This function generates random numbers for a set of univariate distributions based on the distribution quantiles. Internally, this is achieved by fitting the distribution function to the given quantiles using `rriskFitdist.perc`.

Usage

```
rdistq_fit(distribution, n, percentiles = c(0.05, 0.5, 0.95), quantiles)
```

Arguments

`distribution`	A character string that defines the univariate distribution to be randomly sampled.
`n`	Number of generated observations.
`percentiles`	Numeric vector giving the percentiles.
`quantiles`	Numeric vector giving the quantiles.

Details

The follwing table shows the available distributions and their identification as a character string:

Identification	Distribution	Number of quantiles
norm	Normal distribution	>=2
beta	Beta distribution	ToDo
cauchy	ToDo	ToDo
logis	ToDo	ToDo
t	ToDo	ToDo
chisq	ToDo	ToDo
chisqnc	ToDo: implement?	ToDo
exp	ToDo	ToDo
f	ToDo	ToDo
gamma	ToDo	ToDo
lnorm	ToDo	ToDo
unif	ToDo	ToDo
weibull	ToDo	ToDo

triang	ToDo	ToDo
gompertz	ToDo	ToDo
pert	ToDo	ToDo
tnorm	Truncated normal distribution	ToDo

The default for percentiles is 0.05, 0.5 and 0.95, so for the default, the quantiles argument should be a vector with 3 elements. If this is to be longer, the percentiles argument has to be adjusted to match the length of quantiles.

Value

ToDo

---

summary.mcSimulation *Summarize Results from Monte Carlo Simulation.*

Description

summary.mcSimulation produces result summaries of the results of a Monte Carlo simulation obtained by the function mcSimulation.

Usage

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

Arguments

object	An object of class mcSimulation.
...	Further arguments #ToDo

Value

An object of class summary.mcSimulation.

---

summary.plsr.mcSimulation
Summarize Results of the PLSR for Monte Carlo.

Description

summary.plsr.mcSimulation produces result summaries of the PLSR for Monte Carlo obtained by the function plsr.mcSimulation.

Usage

```
## S3 method for class 'plsr.mcSimulation'
summary(object, ...)
```

Arguments

object	An object of class plsr.mcSimulation.
...	Further arguments #ToDo

Value

An object of class summary.plsr.mcSimulation.

---

summary.via *Summarize Results of the Value of Information Analysis.*

Description

summary.via produces result summaries of the results of of the Value of Information Analysis obtained by the function via.

Usage

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

Arguments

object	An object of class via.
...	Further arguments #ToDo

Value

An object of class summary.via.

---

uncertaintyAnalysis *Uncertainty Analysis wrapper function.*

Description

This function performs a Monte Carlo simulation from input files and analyses the results via Partial Least Squares Regression (PLSR) and calculates the Variable Importance on Projection (VIP). Results are safed as plots.

Usage

```
uncertaintyAnalysis(result_path, input_file, fun, iterations,
  write_table = TRUE, indicators = FALSE, log_scales = FALSE)
```

Arguments

result_path	Path were the result plots and tables are safed.
input_file	Path to input csv file, which gives the input estimate.
fun	The model function.
iterations	The number of Monte Carlo simulations to be performed.
write_table	logical; If the full Monte Carlo simulation results and PLSR results should be written to file.
indicators	logical; If indicator variables should be respected specially.
log_scales	logical; If the scales in the pls plots should be logarithmic.

---

via *Value of Information Analysis (VIA)*

Description

This function performs a Value of Information Analysis of a binary decision problem. For two alternative projects which generate different values the Expected Value of Perfect Information for individual variables or clusters of variables is calculated.

Usage

```
via(decision_model_function, estimate, certainVariables, decisionPrinciple, ...)
```

Arguments

decision_model_function	either a list of the two project functions: list(decision_0, decision_1) with either component being scalar or only one function if its value is the NPV of the project and the decision alternative is the status quo.
estimate	An object of class estimate; The status quo information on the variables, viz. the status quo estimate.
certainVariables	A list of named vectors. Each list element represents a cluster of variables for which the EVPI is calculated. Each component of the vector gives the value of the variable that is assumed to be known with certainty.
decisionPrinciple	A character, equal to either "riskMinimization" or "valueMaximization", which defines the decision principle to use, viz. either risk minimization or maximization of expected net present value.
...	ToDo

Value

An object of class via which contains the clustered EVPI for each cluster ordered by the value (greatest first).

References

Hubbard, Douglas W., ch. 7: Quantifying the Value of Information, in: How to Measure Anything? - Finding the Value of "Intangibles" in Business, John Wiley & Sons, Hoboken, New Jersey, 2014, 3rd Ed.
Jeffrey, Scott R. and Pannell, David J. (2013), Economics of Prioritising Environmental Research: An Expected Value of Partial Perfect Information (EVPPI) Framework, Working Paper, School of Agricultural and Resource Economics, University of Western Australia, http://purl.umn.edu/144944.

---

vv *Value Varier Function.*

Description

Value Varier Function.

Usage

```
vv(var_mean, var_CV, distribution = "normal", n, absolute_trend = NA,
  relative_trend = NA)
```

Arguments

var_mean	ToDo
var_CV	ToDo
distribution	ToDo
n	ToDo
absolute_trend	ToDo
relative_trend	ToDo

---

vvNPV *Value Varier Function with NPV Calculation.*

Description

Value Varier Function with NPV Calculation.

Usage

```
vvNPV(var_mean, var_CV, discount_rate, distribution = "normal", n,
  absolute_trend = NA, relative_trend = NA)
```

Arguments

var_mean	ToDo
var_CV	ToDo
discount_rate	ToDo
distribution	ToDo
n	ToDo
absolute_trend	ToDo
relative_trend	ToDo

Index