Package ‘mc2d’

December 1, 2008

Type Package

Title Tools for Two-Dimensional Monte-Carlo Simulations

Version 0.1-4

Date 2008-12-01

Author Regis Pouillot <rpouillot@yahoo.fr>, Marie Laure Delignette-Muller <ml.delignette@vet-lyon.fr> and Jean-Baptiste Denis <jbdenis@jouy.inra.fr>

Maintainer Regis Pouillot <rpouillot@yahoo.fr>

Depends mvtnorm

Description Various distributions and utilities to ease the use of R to build and study Two-Dimensional Monte-Carlo simulations

License GPL (>= 2)

R topics documented:

`NA.mcnode` *Finite, Infinite, NA and NaN Numbers in mcnode.*

Description

`is.na`, `is.nan`, `is.finite` and `is.infinite` return a logical `mcnode` of the same dimension as `x`.

Usage

```
## S3 method for class 'mcnode':
is.na(x)
## S3 method for class 'mcnode':
is.nan(x)
## S3 method for class 'mcnode':
is.finite(x)
## S3 method for class 'mcnode':
is.infinite(x)
```

Arguments

`x`	A `mcnode` object.

Value

A logical `mcnode` object.

Author(s)

Regis Pouillot

See Also

`is.finite`, `NA`

Examples

```
x <- log(mcstoc(rnorm, nsv=1001))
x
is.na(x)
```

`Ops.mcnode` *Operations on mcnode Objects*

Description

This function alters the way operations are performed on `mcnode` objects for a better consistancy of the theory.

Usage

```
## S3 method for class 'mcnode':
Ops(e1, e2)
```

Arguments

`e1`	An `mcnode` object, a vector or an array.
`e2`	An optionnal `mcnode` object, a vector or a matrix with at least one of both objects as an `mcnode`.

Details

This method will be used for any of the Group `Ops` functions, i.e.:

- `"+"`, `"-"`, `"*"`, `"/"`, `"^"`, `"%%"`, `"%/%"`
- `"&", "|", "!"`
- `"==", "!=", "<", "<=", ">=", ">"`

The rules are as following (illustrated with a `"+"` function and ignoring the `nvariates` dimension):

- `"0" + "0" = "0"`;
- `"0" + "V" = "V"`: classical recycling of the scalar;
- `"0" + "U" = "U"`: classical recycling of the scalar;
- `"0" + "VU" = "VU"`: classical recycling of the scalar;
- `"V" + "V" = "V"`: if both of the same `(nsv)` dimension;
- `"V" + "U" = "VU"`: the `"U"` object will be recycled "by row". The `"V"` object will be recycled classically "by column";
- `"V" + "VU" = "VU"`: if the dimension of the `"V"` is `(nsv)` and the dimension of the `"VU"` is `(nsv x nsu)`. The `"V"` object will be recycled classically "by column";

- `"U" + "U" = "U"`: if both of the same `(nsu)` dimension;
- `"U" + "VU" = "VU"`: if the dimension of the `"U"` is `(nsu)` and the dimension of the `"VU"` is `(nsv x nsu)`. The `"U"` object will be recycled "by row";
- `"VU" + "VU" = "VU"`: if the dimension of the `"VU"` nodes is `(nsu x nsv);`

A vector or an array may be combined with an `mcnode` of size `(nsv x nsu)` if an `mcnode` of this dimension may be built from this vector/array using the `mcdata` function. See `mcdata` for the rules.

The `outm` attribute is transferred as following: `"each" + "each" = "each"`; `"none" + other = other`; `other1 + other2 = other1`. The `outm` attribute of the resulting node may be changed using the `outm` function.

For multivariate nodes, a recycling on the `nvariates` dimension is done if a `(nsu x nsv x nvariates)` node is combined with a `(nsu x nsv x 1)` node.

Value

The results as a `mcnode` object.

Author(s)

Regis Pouillot

See Also

`mcdata`, `mcstoc`

Examples

```
oldvar <- ndvar()
oldunc <- ndunc()
ndvar(30)
ndunc(20)

## Given
x0 <- mcdata(3, type="0")
xV <- mcdata(1:ndvar(), type="V")
xU <- mcdata(1:ndunc(), type="U")
xVU <- mcdata(1:(ndunc()*ndvar()), type="VU")
x0M <- mcdata(c(5, 10), type="0", nvariates=2)
xVM <- mcdata(1:(2*ndvar()), type="V", nvariates=2)
xUM <- mcdata(1:(2*ndunc()), type="U", nvariates=2)
xVUM <- mcdata(1:(2*(ndunc()*ndvar())), type="VU", nvariates=2)

## All possible combinations
## "0"
-x0
x0 + 3

## "V"
-xV
3 + xV
xV * (1:ndvar())
xV * x0
xV - xV
```

```
## "U"
-xU
xU + 3
(1:ndunc()) * xU
xU * x0
xU - xU

## Watch out the resulting type
xV + xU
xU + xV

## "VU"
-xVU
3 + xVU
(1:(ndunc()*ndvar())) * xVU
xVU + xV
x0 + xVU
xU + xVU
xVU - xVU

## Some Multivariates
x0M+3
xVM * (1:ndvar())
xVM - xV
xUM - xU
xVUM - xU
```

`bernoulli` *The Bernoulli Distribution*

Description

Density, distribution function, quantile function and random generation for the Bernoulli distribution with probability equals to `prob`.

Usage

```
dbern(x, prob=.5, log=FALSE)
pbern(q, prob=.5, lower.tail=TRUE, log.p=FALSE)
qbern(p, prob=.5, lower.tail=TRUE, log.p=FALSE)
rbern(n, prob=.5)
```

Arguments

`x,q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`prob`	vector of probabilities of success of each trial.
`log, log.p`	logical; if `TRUE`, probabilities `p` are given as `log(p)`.
`lower.tail`	logical; if `TRUE` (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

Details

These fonctions use the corresponding functions from the `binomial` distribution with argument `size = 1`. Thus, 1 is for success, 0 is for failure.

Value

`dbern` gives the density, `pbern` gives the distribution function, `qbern` gives the quantile function, and `rbern` generates random deviates.

Author(s)

Regis Pouillot

See Also

`Binomial`

Examples

```
rbern(n=10, prob=.5)
rbern(n=3, prob=c(0, .5, 1))
```

`betagen` *The Generalised Beta Distribution*

Description

Density, distribution function, quantile function and random generation for the Beta distribution defined on the `[min, max]` domain with parameters `shape1` and `shape2` (and optional non-centrality parameter `ncp`).

Usage

```
dbetagen(x, shape1, shape2, min=0, max=1, ncp=0, log=FALSE)
pbetagen(q, shape1, shape2, min=0, max=1, ncp=0, lower.tail=TRUE,
          log.p=FALSE)
qbetagen(p, shape1, shape2, min=0, max=1, ncp=0, lower.tail=TRUE,
          log.p=FALSE)
rbetagen(n, shape1, shape2, min=0, max=1, ncp=0)
```

Arguments

`x,q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of observations. If `length(n) > 1`, the length is taken to be the number required.
`shape1, shape2`	Positive parameters of the Beta distribution.
`min`	Vector of minima.

`max`	Vector of maxima.
`ncp`	Non-centrality parameter of the Beta distribution.
`log, log.p`	Logical; if `TRUE`, probabilities `p` are given as `log(p)`.
`lower.tail`	Logical; if `TRUE` (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

Details

$$x \sim betagen(shape1, shape2, min, max, ncp)$$

if

$$\frac{x - min}{max - min} \sim beta(shape1, shape2, ncp)$$

These functions use the `Beta` distribution functions after correct parametrisation.

Value

`dbetagen` gives the density, `pbetagen` gives the distribution function, `qbetagen` gives the quantile function, and `rbetagen` generates random deviates.

Author(s)

Regis Pouillot

See Also

`Beta`

Examples

```
curve(dbetagen(x, shape1=3, shape2=5, min=1, max=6), from = 0, to = 7)
curve(dbetagen(x, shape1=1, shape2=1, min=2, max=5), from = 0, to = 7, lty=2, add=TRUE)
curve(dbetagen(x, shape1=.5, shape2=.5, min=0, max=7), from = 0, to = 7, lty=3, add=TRUE)
```

`converg` *Graph of Running Statistics in the Variability or in the Uncertainty Dimension.*

Description

This function provides basic graphs to evaluate the convergence of a node of a `mc` or a `mccut` object in the variability or in the uncertainty dimension.

Usage

```
converg(x, node=length(x), margin=c("var", "unc"), nvariates=1, iter=1,
        probs=c(0.025, 0.975), lim=c(0.025, 0.975), griddim=NULL,
        log=FALSE)
```

Arguments

`x`	A `mcnode` object, a `mc` object or a `mccut` object.
`node`	The node to be considered in a `mc` object or a `mccut` object, displayed either as the order number or the name of the node. By default: the last node of the object.The corresponding node should not be of type `"0"` in a `mc` object or of type `"0"` or `"V"` in a `mccut` object.
`margin`	The margin used to plot the graph. `margin` is used only if the node is a `"VU"` `mcnode`.
`nvariates`	The variates to be considered. `nvariates` is used only for multivariates nodes.
`iter`	If `margin == "var"` and the node is a `"VU" mcnode`, `iter` specify the iteration in the uncertainty dimension to be used for the graph.
`probs`	The quantiles to be provided in the variability dimension.
`lim`	The quantiles to be used in the uncertainty dimension.
`griddim`	A vector of two integers, indicating the size of the grid of the graph. If `NULL`, the grid is calculated to produce a "nice" graph.
`log`	If `TRUE`, the data will be log transformed.

Details

If the node is of type `"V"`, the running mean, median and `probs` quantiles according to the variability dimension will be provided. If the node is of type `"VU"` and `margin="var"`, this graph will be provided on one simulation in the uncertainty dimension (chosen by `iter`).

If the node is of type `"U"` the running mean, median and `lim` quantiles according to the uncertainty dimension will be provided.

If the node is of type `"VU"` (with `margin="unc"` or from a `mccut` object), one graph are provided for each of the mean, median and `probs` quantiles calculated in the variability dimension.

Note

This function may be used on a `mccut` object only if a `summary.mc` function was used in the third block of the `evalmccut` call. The values used as `probs` arguments in `converg` should have been used in the `summary.mc` function of this third block.

Author(s)

Regis Pouillot

Examples

```
data(total)
converg(xVU, margin="var")
converg(xVU, margin="unc")
```

`cornode` *Builds a Rank Correlation using the Iman and Connover Method.*

Description

This function builds a rank correlation structure between columns of a matrix or between `mcnode` objects using the Iman and Connover method (1982).

Usage

```
cornode(..., target, outrank=FALSE, result=FALSE, seed=NULL)
```

Arguments

`...`	A matrix (each of its `n` columns but the first one will be reordered) or `n mcnode` objects (each elements but the first one will be reordered).
`target`	A scalar (only if `n=2`) or a `(n x n)` matrix of correlation.
`outrank`	Should the order be returned?
`result`	Should the correlation eventually obtained be printed?
`seed`	The random seed used for building the correlation. If `NULL` the `seed` is unchanged.

Details

The arguments should be named.

The function accepts for `data` a matrix or:

- some `"V" mcnode` objects separated by a comma;
- some `"U" mcnode` objects separated by a comma;
- some `"VU" mcnode` objects separated by a comma. In that case, the structure is built columns by colums (the first column of each `"VU" mcnode` will have a correlation structure, the second ones will have a correlation structure, ...).
- one `"V" mcnode` as a first element and some `"VU" mcnode` objects, separated by a comma. In that case, the structure is built between the `"V" mcnode` and each column of the `"VU" mcnode` objects. The correlation result (`result = TRUE`) is not provided in that case.

The number of variates of the elements should be equal.

`target` should be a scalar (two columns only) or a real symmetric positive-definite square matrix. Only the upper triangular part of `target` is used (see `chol`).

The final correlation structure should be checked because it is not always possible to build the target correlation structure.

In a Monte-Carlo simulation, note that the order of the values within each `mcnode` will be changed by this function (excepted for the first one of the list). As a consequence, previous links between variables will be broken. The `outrank` option may help to rebuild these links (see the Examples).

Value

If `rank = FALSE`: the matrix or a list of rearranged `mcnodes`.

If `rank = TRUE`: the order to be used to rearranged the matrix or the `mcnodes` to build the desired correlation structure.

References

Connover W., Iman R. (1982). A distribution-free approach to inducing rank correlation among input variables. Technometric, 3, 311-334.

Examples

```
x1 <- rnorm(1000)
x2 <- rnorm(1000)
x3 <- rnorm(1000)
mat <- cbind(x1, x2, x3)
## Target
(corr <- matrix(c(1, 0.5, 0.2, 0.5, 1, 0.2, 0.2, 0.2, 1), ncol=3))
## Before
cor(mat, method="spearman")
matc <- cornode(mat, target=corr, result=TRUE)
## The first row is unchanged
all(matc[, 1] == mat[, 1])

##Using mcnode and outrank
cook <- mcstoc(rempiricalD, values=c(0, 1/5, 1/50), prob=c(0.027, 0.373, 0.600), nsv=1000
serving <- mcstoc(rgamma, shape=3.93, rate=0.0806, nsv=1000)
roundserv <- mcdata(round(serving), nsv=1000)
## Strong relation between roundserv and serving (of course)
cor(cbind(cook, roundserv, serving), method="spearman")

##The classical way to build the correlation structure
matcorr <- matrix(c(1, 0.5, 0.5, 1), ncol=2)
matc <- cornode(cook=cook, roundserv=roundserv, target=matcorr)
## The structure between cook and roundserv is OK but ...
## the structure between roundserv and serving is lost
cor(cbind(cook=matc$cook, serv=matc$roundserv, serving), method="spearman")

##An alternative way to build the correlation structure
matc <- cornode(cook=cook, roundserv=roundserv, target=matcorr, outrank=TRUE)
## Rebuilding the structure
roundserv[] <- roundserv[matc$roundserv, , ]
serving[] <- serving[matc$roundserv, , ]
## The structure between cook and roundserv is OK and ...
## the structure between roundserv and serving is preserved
cor(cbind(cook, roundserv, serving), method="spearman")
```

`dimmcnode` *Dimension of mcnode and mc Objects*

Description

Provides the dimension (i.e. the number of simulations in the variability dimension, the number of simulations in the uncertainty dimension and the maximum number of variates of a `mcnode` or a `mc` object.

Usage

```
dimmcnode(x)
dimmc(x)
```

Arguments

`x`	a `mcnode` or a `mc` object.

Value

A vector of three scalars: the dimension of variability (1 for `"0"` and `"U" mcnode`), the dimension of uncertainty (1 for `"0"` and `"V" mcnode`) and the number of variates (the maximal number of variates for an `mc` object.

Note

This function does not test if the object is correctly built. See `is.mcnode` and `is.mc` .

Author(s)

Regis Pouillot

Examples

```
data(total)
dimmcnode(xVUM2)
dimmc(total)
```

`dirichlet` *The Dirichlet Distribution*

Description

Density function and random generation from the Dirichlet distribution.

Usage

```
ddirichlet(x, alpha)
rdirichlet(n, alpha)
```

Arguments

`x`	A vector containing a single deviate or a matrix containing one random deviate per row.
`alpha`	A vector of shape parameters, or a matrix of shape parameters by rows. Recycling (by row) is permitted.
`n`	Number of random vectors to generate. If length(n) $>$ 1, the length is taken to be the number required.

Details

The Dirichlet distribution is the multidimensional generalization of the beta distribution. The original code was adapted to provide a kind of "vectorization" used in multivariates `mcnode`.

Value

`ddirichlet` gives the density. `rdirichlet` returns a matrix with `n` rows, each containing a single Dirichlet random deviate.

Author(s)

Code is adapted from `MCMCpack`. It originates from Greg's Miscellaneous Functions (gregmisc).

See Also

`Beta`

Examples

```
dat <- c(1, 10, 100, 1000, 1000, 100, 10, 1)
(alpha <- matrix(dat, nrow=4, byrow=TRUE))
round(x <- rdirichlet(4, alpha), 2)
ddirichlet(x, alpha)

## rdirichlet used with mcstoc
mcalpha <- mcdata(dat, type="V", nsv=4, nvariates=2)
(x <- mcstoc(rdirichlet, type="V", alpha=mcalpha, nsv=4, nvariates=2))
unclass(x)
x <- mcstoc(rdirichlet, type="VU", alpha=mcalpha, nsv=4, nsu=10, nvariates=2)
unclass(x)
```

`dmultinomial` *The Vectorized Multinomial Distribution*

Description

Generate multinomially distributed random number vectors and compute multinomial probabilities.

Usage

```
dmultinomial(x, size=NULL, prob, log=FALSE)
rmultinomial(n, size, prob)
```

Arguments

`x`	Vector of length K of integers in 0:size.
`n`	Number of random vectors to draw.
`size`	A vector of integers, say N, specifying the total number of objects that are put into K boxes in the typical multinomial experiment. For `dmultinom`, it defaults to `sum(x)`. The first element correspond to the vector `prob` or the first row of `prob`, ...
`prob`	Numeric non-negative vector of length K, or matrix of size `(x x K)` specifying the probability for the K classes; is internally normalized to sum 1.
`log`	Logical; if TRUE, log probabilities are computed.

Details

This function is the vectorized version of `rmultinom` and `dmultinom`. Recycling is permitted.

Examples

```
prob <- c(1, 2, 7)
rmultinomial(4, 1000, prob)
rmultinomial(4, c(10, 100, 1000, 10000), prob)

## rmultinomial used with mcstoc
## (uncertain size and prob)
s <- mcstoc(rpois, "U", lambda=50)
p <- mcstoc(rdirichlet, "U", nvariates=3, alpha=c(4, 10, 20))
mcstoc(rmultinomial, "VU", nvariates=3, size=s, prob=p)
```

`ec` *An exemple on Escherichia coli in ground beef*

Description

The fictive example is as following:

A batch of ground beef is contaminated with *E. coli*, with a mean concentration `conc`.

Consumers may eat the beef "rare", "medium rare" or "well cooked". If "rare", no bacteria is killed. If "medium rare", 1/5 of bacteria survive. If "well cooked", 1/50 of bacteria survive.

The serving size is variable.

The risk of infection follows an exponential model.

For the one-dimensional model, it is assumed that:

```
conc <- 10
cook <- sample(n, x=c(1,1/5,1/50),replace=TRUE,prob=c(0.027,0.373,0.600))
serving <- rgamma(n, shape=3.93,rate=0.0806)
expo <- conc * cook * serving
dose <- rpois(n, lambda=expo)
risk <- 1-(1-0.001)^dose
```

For the two-dimensional model, it is assumed moreover that the concentration and the `r` parameter of the dose response are uncertain.

```
conc <- rnorm(n,mean=10,sd=2)
r <- runif(n ,min=0.0005,max=0.0015)
```

Usage

```
ec
```

Format

A list of two expression to be passed in `mcmodel`.

modEC1 Expression for a one dimension model.

modEC2 Expression for a two dimensions model.

Source

Fictive example

References

None

`empirical` *The Discrete Empirical Distribution*

Description

Density, distribution function and random generation for a discrete empirical distribution. This function is vectorized to accept different sets of `values` or `prob`.

Usage

```
dempiricalD(x, values, prob=NULL, log=FALSE)
pempiricalD(q, values, prob=NULL, lower.tail=TRUE, log.p=FALSE)
qempiricalD(p, values, prob=NULL, lower.tail=TRUE, log.p=FALSE)
rempiricalD(n, values, prob=NULL)
```

Arguments

`x, q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of random values. If length(n) > 1, the length is taken to be the number required.
`values`	Vector or matrix of numerical values. See details.
`prob`	Optionnal vector or matrix of count or probabilities. See details.
`log, log.p`	logical; if `TRUE`, probabilities `p` are given as `log(p)`.
`lower.tail`	logical; if `TRUE` (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

Details

If `prob` is missing, the discrete distribution is obtained directly from the vector of `values`, otherwise `prob` is used to weight the values. `prob` is normalized before use. Thus, `prob` may be the count of each `values`. `prob` values should be non negative and their sum should not be 0.

`values` and/or `prob` may vary: in that case, `values` and/or `prob` should be sent as matrixes, the first row being used for the first element of `x`, `q`, `p` or the first random value, the second row for the second element of `x`, `q`, `p` or random value, ... Recycling is permitted if the number of rows of `prob` and `values` are equal or if the number of rows of `prob` and/or `values` are one.

`rempiricalD(n, values, prob)` with `values` and `prob` as vectors is equivalent to `sample(x=values, size=n, replace=TRUE, prob=prob)`.

Value

`dempiricalD` gives the density, `pempiricalD` gives the distribution function, `qempiricalD` gives the quantile function and `rempiricalD` generates random deviates.

Note

In the future, the fonctions should be written for non numerical values.

Author(s)

Regis Pouillot

See Also

`sample`.

Examples

```
dempiricalD(1:6, 2:6, prob=c(10, 10, 70, 0, 10))
pempiricalD(1:6, 2:6, prob=c(10, 10, 70, 0, 10))
qempiricalD(seq(0, 1, 0.1), 2:6, prob=c(10, 10, 70, 0, 10))
table(rempiricalD(10000, 2:6, prob=c(10, 10, 70, 0, 10)))

## varying values
(values <- matrix(1:10, ncol=5))
## the first x apply to the first row : p = 0.2
## the second x to the second one: p = 0
dempiricalD(c(1, 1), values)
```

`evalmcmod` *Evaluates a Monte-Carlo model*

Description

Evaluates a `mcmodel` object (or a valid expression) using a specified number of simulations and with (or without) a specified seed.

Usage

```
evalmcmod(expr, nsv=ndvar(), nsu=ndunc(), seed=NULL)
```

Arguments

`expr`	A model of class `mcmodel` or a valid expression.
`nsv`	The number of simulations in the dimension of variability used in the evaluation.
`nsu`	The number of simulations in the dimension of uncertainty used in the evaluation.
`seed`	The random seed used for the evaluation. If `NULL` the `seed` is unchanged.

Details

The model is evaluated. The intermediate variables used to build the `mc` object are not stored.

Value

The results of the evaluation. It should be a `mc` object.

Note

The seed is set at the beginning of the evaluation. Thus, the complete similarity of two evaluations with similar seed is not certain, depending on the structure of your model.

Author(s)

Regis Pouillot

See Also

`mcmodel`

`evalmccut` to evaluate high dimension Monte Carlo Model in a loop.

Examples

```
data(ec)
ec$modEC1
evalmcmod(ec$modEC1, nsv=100, nsu=100, seed=666)
```

---

hist.mc   *Histogram of a Monte Carlo Simulation*

Description

Shows histogram of a `mcnode` or a `mc` object.

Usage

```
## S3 method for class 'mc':
hist(x, griddim=NULL, xlab=names(x), ylab="Frequency", main="", ...)
## S3 method for class 'mcnode':
hist(x, ...)
```

Arguments

`x`	An `mcnode` or an `mc` object.
`griddim`	A vector of two integers, indicating the size of the grid of plots. If `NULL`, the grid is calculated to produce a "nice" graph.
`xlab`	A vector of labels for the x-axis for drawn histograms (those whose `outm(x)!="none"`). May be recycled.
`ylab`	A vector of labels for the y-axis for drawn histograms. May be recycled.
`main`	A vector of main title of histograms for drawn histograms. May be recycled.
`...`	Other arguments to be passed to all calls of `hist`.

Note

For Two-dimensional `mc`, the histogram is based on all data (variability and uncertainty) pooled together.

Author(s)

Regis Pouillot

Examples

```
data(total)
hist(xVUM3)
hist(total)
```

`is.mc` *Tests mc and mcnode Objects*

Description

`is.mc` tests `mc` objects and `is.mcnode` tests `mcnode` objects.

Usage

```
is.mc(x)
is.mcnode(x)
```

Arguments

`x`	An `mc` or a `mcnode` object.

Details

`is.mc` tests if `x` is a list of `mcnode`, each elements being of compatible dimension. It tests if the class `"mc"` is affected to the object.

`is.mcnode` tests if `x` is an array of numeric or logical, if it has a "type" attribute and compatible dimensions, and if the class `"mcnode"` is affected to the object.

Value

`TRUE` or `FALSE`

Author(s)

Regis Pouillot

Examples

```
data(total)
is.mcnode(xVU)
is.mcnode(total)
is.mc(total)
```

`lhs` *Random Latin Hypercube Sampling*

Description

Creates a Latin Hypercube Sample (LHS) of the specified distribution.

Usage

```
lhs(distr=runif, nsv, nsu, nvariates=1, ...)
```

Arguments

`distr`	The function for generating random sample or its name. If `distr` is "rdist", the function "qdist" must be the quantile function of this distribution with argument `p` as a vector of probabilities, as all univariates distributions of the `stat` library.
`nsv`	The number of raws of the final matrix.
`nsu`	The number of columns of the final matrix
`nvariates`	The number of variates
`...`	All arguments to be passed to `distr` except the size of the sample.

Value

A `nsv x nsu` matrix of random variates.

Note

The resulting lhs is in fact a latin hypersquare sampling: the lhs is provided only in the first 2 dimensions.

It is not possible to send truncated distribution with `rtrunc`. Use `mcstoc` for this purpose, with `lhs=TRUE` and `rtrunc=TRUE`.

The ... arguments will be recycled.

Author(s)

adapted from a code of Rob Carnell (library `lhs`)

See Also

`mcstoc`

Examples

```
ceiling(lhs(runif, nsu=10, nsv=10)*10)
```

mc *Monte Carlo Object*

Description

Creates `mc` objects from `mcnode` or `mc` objects.

Usage

```
mc(..., name=NULL, devname=FALSE)
```

Arguments

`...`	`mcnode` and/or `mc` object(s) to be gathered in a `mc` object separated by a coma.
`name`	Vector of character of the same length of the final `mc` object. If NULL, the name will be given from the name of the elements.
`devname`	Develop the name from the name of the `mc` objects, if any.

Details

A `mc` object is a list of `mcnode` objects. `mcnode` objects must be of coherent dimensions.

If one of the arguments is a `mc` object, the name of the elements of this `mc` object are used. `devname = TRUE` will develop the name, using as a prefix the name of the `mc` object.

Finally, names are transformed to be unique.

Value

An object of class `mc`.

Author(s)

Regis Pouillot

See Also

`mcnode`, the basic element of a `mc` object.

To evaluate `mc` objects: `mcmodel`, `evalmcmod`, `evalmccut`

Informations about an `mc` object: `is.mc`, `dimmc`

To study `mc` objects: `print.mc`, `summary.mc`, `plot.mc`, `converg`, `hist.mc`, `tornado`, `tornadounc.mc`

Examples

```
x <- mcstoc(runif)
y <- mcdata(3, type="0")
z <- x * y
(m <- mc(x, y, z, name=c('n1', 'n2', 'n3')))
mc(m, x, devname=TRUE)
```

`mc.control` *Sets or Gets the Default Number of Simulations.*

Description

Sets or retrieves the default number of simulations.

Usage

```
ndvar(n)
ndunc(n)
```

Arguments

`n`	Number of simulations.

Details

`ndvar()` gets and `ndvar(n)` sets the default number of simulation in the 1D simulations or the number of simulation in the variability dimension in the 2D simulations.

`ndunc()` gets and `ndunc(n)` sets the number of simulations in the uncertainty dimension in the 2D simulations.

`n` is rounded to its ceiling value.

The default values when loaded are 1001 for `ndvar` and 101 for `ndunc`.

Value

The current value, AFTER modification if `n` is present (!= `options`).

Author(s)

Regis Pouillot

Examples

```
(oldvar <- ndvar())
(oldunc <- ndunc())
mcstoc(runif, type="VU")
ndvar(12)
ndunc(21)
mcstoc(runif, type="VU")
ndvar(oldvar)
ndunc(oldunc)
```

`mcapply` *Apply Functions Over mc or mcnode Objects*

Description

Apply a function on all values or over a given dimension of an `mcnode` object. May be used for all `mcnode` of an `mc` object.

Usage

```
mcapply(x, margin=c("all", "var", "unc", "variates"), fun, ...)
```

Arguments

`x`	A `mc` or a `mcnode` object.
`margin`	The dimension on which applying the function. Maybe `"all"` (default) to apply the function on all values, `"var"` to apply the function on the variability dimension, `"unc"` to apply the function on the uncertainty dimension, or `"variates"` to apply the function on the variates.
`fun`	The function to be applied. When applied to a vector of length `n`, `fun` should return a vector of length `n` or `1`.
`...`	Optionnal arguments to `fun`.

Value

If `fun` returns a function of length `n` or if `margin="all"`, the returned `mcnodes` are of type and dimension of `x`. In other cases, the type of `mcnode` is changed.

Author(s)

Regis Pouillot

See Also

`apply`, `mc`, `mcnode`.

Examples

```
data(total)
xVUM
mcapply(xVUM, "unc", sum)
mcapply(xVUM, "var", sum)
mcapply(xVUM, "all", sum)
mcapply(xVUM, "variates", sum)
mcapply(total, "all", exp)
```

mccut — *Evaluates a Two-Dimensional Monte Carlo Model in a Loop.*

Description

`evalmccut` evaluates a Two-Dimensional Monte Carlo model using a loop on the uncertainty dimension. Within each loop, it calculates statistics in the variability dimension and stores them for further analysis. It allows to evaluate very high dimension models using (unlimited?) time instead of (limited) memory.

`mcmodelcut` builds a `mcmodelcut` object that can be sent to `evalmccut`.

Usage

```
evalmccut(model, nsv=ndvar(), nsu=ndunc(), seed=NULL, ind="index")
## S3 method for class 'mccut':
print(x, lim=c(0.025, 0.975), digits=3, ...)
mcmodelcut(x, is.expr=FALSE)
```

Arguments

`model`	a `mcmodelcut` object obtained using `mcmodelcut` function or (directly) a valid call including three blocks. See Details and Examples for the structure of the call.
`x`	a call or an expression (if `is.expr=TRUE`) including three blocks. See Details and Examples for the structure of the call.
`nsv`	The number of simulations for variability used in the evaluation.
`nsu`	The number of simulations for uncertainty used in the evaluation.
`seed`	The random seed used for the evaluation. If `NULL` the `seed` is unchanged.
`ind`	The variable name used in `model` to refers to the uncertainty. see Details and Example.
`is.expr`	`FALSE` to send a call, `TRUE` to send an expression (see `mcmodel` examples)
`lim`	A vector of values used for the quantile function (uncertainty dimension).
`digits`	Number of digits in the print.
`...`	Additional arguments to be passed in the final print function.

Details

This function should be used for high dimension Two-Dimensional Monte-Carlo simulations, when the memory limits of R are attained. The use of a loop will take (lots of) time, but less memory.

`x` (or `model` if a call is used directly in `evalmccut`) should be built as three blocks, separated by `{`.

1. The first block is evaluated once (and only once) before the first loop (step 1).
2. The second block, which should lead to an `mc` object, is evaluated using `nsu = 1` (step 2).
3. The third block is evaluated on the `mc` object. All resulting statistics are stored (step 3).
4. The steps 2 and 3 are repeated `nsu` times. At each iteration, the values of the loop index (from 1 to `nsu`) is given to the variable specified in `ind`.

5. Finally, the `nsu` statistics are returned in an invisible object of class `mccut`.

Understanding this, the call should be built like this: `{{block 1}{block 2}{block 3}}`

1. The first block (maybe empty) is an expression that will be evaluated only once. This block should evaluate all `"V" mcnode` and `"0" mcnodes`. It may evaluate and `"U" mcnode` that will be sent in the second and third block by column, and, optionnaly, some other codes (even `"VU" mcnode`, sent by columns) that can not be evaluated if `ndunc=1` (e.g. sampling without replacement in the uncertainty dimension).
2. The second block is an expression that leads to the `mc` object. It must end with an expression as `mymc <- mc(...)`. The variable specified as `ind` may be helpful to refer to the uncertainty dimension in this step
3. The last block should build a list of statistics refering to the `mc` object. The function `summary` should be used if a summary, a tornado on uncertainty (`tornadounc.mccut`) or a convergence diagnostic `converg` is needed, the function `plot.mc` should be used if a plot is needed, the function `tornado` should be used if a tornado is needed. Moreover, any other function that leads to a vector, a matrix, or a list of vector/matrix of statistics evaluated from the `mc` object may be used. list are time consuming.

IMPORTANT WARNING: do not forget to affect the results, since the print method provide only a summary of the results while all data may be stored in an `mccut` object.

Value

An object of class `mccut`. This is a list including statistics evaluated within the third block. Each list consists of all the `nsu` values obtained. The `print.mccut` method print the median, the mean, the `lim` quantiles estimated on each statistics on the uncertainty dimension.

Note

The methods and functions available on the `mccut` object is function of the statistics evaluated within the third block:

- a `print.mccut` is available as soon as one statistic is evaluated within the third block;
- a `summary.mccut` and a `tornadounc.mccut` are available if a `summary.mc` is evaluated within the third block;
- `converg` may be used if a `summary.mc` is evaluated within the third block;
- a `plot.mccut` is available if a `plot.mc` is evaluated within the third block. (Do not forget to use the argument `draw = FALSE` in the third block);
- a `tornado` is available if a `tornado` is evaluated within the third block.

The seed is set at the beginning of the evaluation. Thus, the complete similarity of two evaluations is not certain, depending of the structure of your model. Moreover, with a similar seed, the simulation will not be equal to the one obtained with `evalmcmod` since the random samples will not be obtained in the same order.

In order to avoid conflicts between the `model` evaluation and the function, the function uses upper case variables. Do not use upper case variables in your model.

The function should be re-adapted if a new function to be applied on `mc` objects is written.

Author(s)

Regis Pouillot

See Also

evalmcmod

Examples

```
modEC3 <- mcmodelcut({

## First block:
## Evaluates all the 0, V and U nodes.
 { cook <- mcstoc(rempiricalD, type = "V", values = c(0, 1/5,
 1/50), prob = c(0.027, 0.373, 0.6))
 serving <- mcstoc(rgamma, type = "V", shape = 3.93, rate = 0.0806)
 conc <- mcstoc(rnorm, type = "U", mean = 10, sd = 2)
 r <- mcstoc(runif, type = "U", min = 5e-04, max = 0.0015)
 }
## Second block:
## Evaluates all the VU nodes
## Leads to the mc object.
 {
 expo <- conc * cook * serving
 dose <- mcstoc(rpois, type = "VU", lambda = expo)
 risk <- 1 - (1 - r)^dose
 res <- mc(conc, cook, serving, expo, dose, r, risk)
 }
## Third block:
## Leads to a list of statistics: summary, plot, tornado
## or any function leading to a vector (et), a list (minmax),
## a matrix or a data.frame (summary)
 {
 list(
 sum = summary(res),
 plot = plot(res, draw=FALSE),
 minmax = lapply(res, range),
 et = sapply(res, sd)
 )
 }
})

evalmccut(modEC3, nsv = 101, nsu = 101, seed = 666)
```

---

mcmodel *Monte Carlo model*

Description

Specify a `mcmodel`, without evaluating it, for a further evaluation using evalmcmod.

Usage

```
mcmodel(x, is.expr=FALSE)
```

Arguments

`x`	An R call or an expression.
`is.expr`	`FALSE` to send a call, `TRUE` to send an expression (see Examples)

Details

The model should be put between `{` and the last line should be of the form `mc(...)`. Any reference to the number of simulation in the dimension of variability should be done via `ndvar()` or (preferred) `nsv`. Any reference to the number of simulations in the dimension of uncertainty should be done via `ndunc()` or (preferred) `nsu`.

Value

an R expression, with class `mcmodel`

Author(s)

Regis Pouillot

See Also

`expression`.

`evalmcmod` to evaluate the model.

`mcmodelcut` to evaluate high Dimension Monte Carlo Model in a loop.

Examples

```
modEC1 <- mcmodel({
 conc <- mcdata(10, "0")
 cook <- mcstoc(rempiricalD, values=c(0, 1/5, 1/50), prob=c(0.027, 0.373, 0.600))
 serving <- mcstoc(rgamma, shape=3.93, rate=0.0806)
 expo <- conc * cook * serving
 dose <- mcstoc(rpois, lambda=expo)
 risk <- 1-(1-0.001)^dose
 mc(conc, cook, serving, expo, dose, risk)
 })
evalmcmod(modEC1, nsv=100, nsu=100)
```

`mcnode` *Build mcnode Objects from Data or other mcnode Objects*

Description

Creates a `mcnode` object from a vector, an array or a `mcnode`.

Usage

```
mcdata(data, type=c("V", "U", "VU", "0"), nsv=ndvar(), nsu=ndunc(),
          nvariates=1, outm="each")
mcdatanocontrol(data, type=c("V", "U", "VU", "0"), nsv=ndvar(), nsu=ndunc(),
          nvariates=1, outm="each")
```

Arguments

`data`	The numeric/logical vector/matrix/array of data or the `mcnode` object.
`type`	The type of node to be built. By default, a `"V"` node.
`nsv`	The variability dimension (`type="V"` or `type="VU"`) of the node. By default: the current value in `mc.control`
`nsu`	The uncertainty dimension (`type="U"` or `type="VU"`) of the node. By default: the current value in `mc.control`
`nvariates`	The number of variates. By default: 1
`outm`	The output of the `mcnode` for multivariates nodes. May be "each" (default) if output should be provided for each variates considered independently, "none" for no output or a vector of name of function(s) (as a character string) that will be applied on the variates dimension before any output (ex: `"mean"`, `"median"`, `c("min", "max")`). The function should have no other arguments and send one value per vector of values (ex. do not use `"range"`). Note that the `outm` attribute may be changed at any time using the `outm` function.

Details

A `mcnode` object is the basic element of a `mc` object. It is an array of dimension `(nsv x nsu x nvariates)`. Four types of `mcnode` exists:

- `"V" mcnode`, for "Variability", are arrays of dimension `(nsv x 1 x nvariates)`. The alea in the data should reflect variability of the parameter.
- `"U" mcnode`, for "Uncertainty", are arrays of dimension `c(1 x nsu x nvariates)`. The alea in the data should reflect uncertainty of the parameter.
- `"VU" mcnode`, for "Variability and Uncertainty", are arrays of dimension `(nsv x nsu x nvariates)`. The alea in the data reflects separated variability (in rows) and uncertainty (in columns) of the parameter.
- `"0" mcnode`, for "Neither Variability or Uncertainty", are arrays of dimension `(1 x 1 x nvariates)`. No alea is considered for these nodes. `"0" mcnode` are not necessary in the univariate context (use scalar instead) but may be useful for operations on multivariate nodes.

Multivariate nodes (i.e. `nvariates != 1`) should be used for multivariate distributions implemented in `mc2d` (`rmultinomial`, `rmultinormal`, `rempiricalD` and `rdirichlet`).

For security, recycling rules are limited to fill the array using `data`. The general rules is that recycling is only permitted to fill a dimension from 1 to the final size of the dimension.

If the final dimension of the node is `(nsv x nsu x nvariates)` (with `nsv = 1` and `nsu = 1` for `"0"` nodes, `nsu = 1` for `"V"` nodes and `nsv = 1` for `"U"` nodes), `mcdata` accepts :

- Vectors of length `1` (recycled on all dimensions), vectors of length `(nsv * nsu)` (filling first the dimension of variability, then the dimension of uncertainty then recycling on nvariates), or vectors of length `(nsv * nsu * nvariates)` (filling first the dimension of variability, then the uncertainty, then the variates).
- Matrixes of dimensions `(nsv x nsu)`, recycling on variates.
- Arrays of dimensions `(nsv x nsu x nvariates)` or `(nsv x nsu x 1)`, recycling on variates.
- For `data` as `mcnode`, recycling is dealt to proper fill the array:

1. a "V" node accepts a "0" node of dimension (1 x 1 x nvariates) (recycling on variability) or of dimension (1 x 1 x 1) (recycling on variability and variates), or a "V" node of dimension (nsv x 1 x nvariates) or (nsv x 1 x 1) (recycling on variates),
2. a "U" node accepts a "0" node of dimension (1 x 1 x nvariates) (recycling on uncertainty) or of dimension (1 x 1 x 1) (recycling on uncertainty and variates), or a "U" node of dimension (1 x nsu x nvariates), or (1 x nsu x 1) (recycling on variates),
3. a "VU" node accepts a "0" node of dimension (1 x 1 x nvariates) (recycling on varaiability and uncertainty) or of dimension (1 x 1 x 1) (recycling on variability, uncertainty and variates), a "U" node of dimension (1 x nsu x nvariates) (recycling "by row" on the variability dimension), or of dimension (1 x nsu x 1) (recycled "by row" on the variability dimension then on variates), a "V" node of dimension (nsv x 1 x nvariates) (recycling on the uncertainty dimension) or of dimension (nsv x 1 x 1) (recycled on the uncertainty dimension then on variates), and a "VU" node of dimension (nsv x nsu x nvariates) or of dimension (nsv x nsu x 1) (recycling on variates).
4. a "0" node accepts a "0" node of dimension (1 x 1 x nvariates) or (1 x 1 x 1) (recycling on variates).

mcdatanocontrol is a dangerous version of mcnode which forces the dimension of data to be (nsv x nsu x nvariates) and gives the atributes and the class without any control. This function is useful when your model is tested since it is much more quicker.

Value

An mcnode object.

Author(s)

Regis Pouillot

See Also

mcstoc to build a stochastic mcnode object, mcprobtree to build a stochastic node fro a probability tree.

Ops.mcnode for operations on mcnode objects.

mc to build a Monte-Carlo object.

Informations about an mcnode: is.mcnode, dimmcnode, typemcnode.

To build a correlation structure between mcnode: cornode.

To study mcnode objects: print.mcnode, summary.mcnode, plot.mcnode, converg, hist.mcnode

To modify mcnode objects: NA.mcnode

Examples

```
oldvar <- ndvar()
oldunc <- ndunc()
ndvar(3)
ndunc(5)

(x0 <- mcdata(100, type="0"))
```

```
mcdata(matrix(100), type="0")

(xV <- mcdata(1:ndvar(), type="V"))
mcdata(matrix(1:ndvar(), ncol=1), type="V")

(xU <- mcdata(10*1:ndunc(), type="U"))
mcdata(matrix(10*1:ndunc(), nrow=1), type="U")

(xVU <- mcdata(1:(ndvar()*ndunc()), type="VU"))
mcdata(matrix(1:(ndvar()*ndunc()), ncol=5, nrow=3), type="VU")

##Do not use
## Not run:
mcdata(matrix(1:5, nrow=1), type="VU")
## End(Not run)
##use instead
mcdata(mcdata(matrix(1:ndunc(), nrow=1), type="U"), "VU")
##or
mcdata(matrix(1:ndunc(), nrow=1), type="U") + mcdata(0, "VU")

mcdata(x0, type="0")

mcdata(x0, type="V")
mcdata(xV, type="V")

mcdata(x0, type="U")
mcdata(xU, type="U")

mcdata(x0, type="VU")
mcdata(xU, type="VU")
mcdata(xV, type="VU")

##Multivariates
(x0M <- mcdata(1:2, type="0", nvariates=2))
mcdata(1, type="0", nvariates=2)

(xVM <- mcdata(1:(2*ndvar()), type="V", nvariates=2))
mcdata(1:ndvar(), type="V", nvariates=2)
mcdata(array(1:(2*ndvar()), dim=c(3, 1, 2)), type="V", nvariates=2)

mcdata(1, type="V", nvariates=2)
mcdata(x0, type="V", nvariates=2)
mcdata(x0M, type="V", nvariates=2)
mcdata(xV, type="V", nvariates=2)
mcdata(xVM, type="V", nvariates=2)

(xUM <- mcdata(10*(1:(2*ndunc())), type="U", nvariates=2))
mcdata(array(10*(1:(2*ndunc())), dim=c(1, 5, 2)), type="U", nvariates=2)

mcdata(1, type="U", nvariates=2)
mcdata(x0, type="U", nvariates=2)
mcdata(x0M, type="U", nvariates=2)
mcdata(xU, type="U", nvariates=2)
mcdata(xUM, type="U", nvariates=2)

(xVUM <- mcdata(1:(ndvar()*ndunc()), type="VU", nvariates=2))
mcdata(array(1:(ndvar()*ndunc()), dim=c(3, 5, 2)), type="VU", nvariates=2)
mcdata(1, type="VU", nvariates=2)
mcdata(x0, type="VU", nvariates=2)
mcdata(x0M, type="VU", nvariates=2)
mcdata(xV, type="VU", nvariates=2)
mcdata(xVM, type="VU", nvariates=2)
mcdata(xU, type="VU", nvariates=2)
mcdata(xUM, type="VU", nvariates=2)
mcdata(xVU, type="VU", nvariates=2)
mcdata(xVUM, type="VU", nvariates=2)

ndvar(oldvar)
ndunc(oldunc)
```

`mcprobtree` *Creates a Stochastic mcnode Object using a Probability Tree*

Description

This function builds a `mcnode` as a mixture of `mcstoc` functions or `mcnode` objects.

Usage

```
mcprobtree(mcswitch, mcvalues, type=c("V", "U", "VU", "0"), nsv=ndvar(),
           nsu=ndunc(), nvariates=1, outm="each", seed=NULL)
```

Arguments

`mcswitch`	A vector of probabilities/weights or a `mcnode` including the `mcstoc` functions/mcnodes to pick.
`mcvalues`	A named list of `mcnode`, `mcdata` functions or `mcstoc` functions, or a combination of those objects. Each element should lead to an `mcnode` of type `type` and of dimension `c(nsv x nsu x 1)` or `c(nsv x nsu x nvariates)`
`type`	The type of `mcnode` to be built. By default, a `"V"` node. see `mcnode` for details.
`nsv`	The number of simulations in the variability dimension of the final node.
`nsu`	The number of simulations in the uncertainty dimension of the final node.
`nvariates`	The number of variates of the final `mcnode`.
`outm`	The default output of the `mcnode` for multivariates nodes. see `outm`.
`seed`	The random seed used for the evaluation. If `NULL` the `seed` is unchanged.

Details

`mcswitch` may be:

- a vector of length the length of `mcvalues`. They need not sum to one, but they should be nonnegative and not all zero. In that case, each elements of `mcvalues` will appear in the final sample a random number of times as specified by this vector. (Note that there is a random process).

- a "0" mcnode to build any type of node.
- a "V" mcnode to build a "V" or a "VU" mcnode.
- a "U" mcnode to build a "U" or a "VU" mcnode.
- a "VU" mcnode to build a "VU".

The elements in mcvalues should be of same type and dimension as specified in type, nsv, nsu and nvariates. The name should correspond to the values in mcswitch, specified as character (See Examples). These elements will be evaluated only if needed : if the corresponding value is not present in mcswitch, the element will not be evaluated.

Value

An mcnode object.

Author(s)

Regis Pouillot

See Also

mcdata, mcstoc, switch.

Examples

```
## A mixture of normal (prob=0.75), uniform (prob=0.20) and constant (prob=0.05)
conc1 <- mcstoc(rnorm, type="VU", mean=10, sd=2)
conc2 <- mcstoc(runif, type="VU", min=-6, max=-5)
conc3 <- mcdata(0, type="VU")
## Randomly in the cells
whichdist <- mcstoc(rempiricalD, type="VU", values=1:3, prob= c(.75, .20, .05))
mcprobtree(whichdist, list("1"=conc1, "2"=conc2, "3"=conc3), type="VU")
## Which is equivalent to
mcprobtree(c(.75, .20, .05), list("1"=conc1, "2"=conc2, "3"=conc3), type="VU")
## Not that there is no control on the exact number of occurences.

## Randomly by colums (Uncertainty)
whichdist <- mcstoc(rempiricalD, type="U", values=1:3, prob= c(.75, .20, .05))
mcprobtree(whichdist, list("1"=conc1, "2"=conc2, "3"=conc3), type="VU")

## Randomly by line (Variability)
whichdist <- mcstoc(rempiricalD, type="V", values=1:3, prob= c(.75, .20, .05))
mcprobtree(whichdist, list("1"=conc1, "2"=conc2, "3"=conc3), type="VU")
```

---

mcstoc *Creates Stochastic mcnode Objects*

Description

Creates a mcnode object using a random generating function.

Usage

```
mcstoc(func=runif, type=c("V", "U", "VU", "0"), ..., nsv=ndvar(),
          nsu=ndunc(), nvariates=1, outm="each", nsample="n",
          seed=NULL, rtrunc=FALSE, linf=-Inf, lsup=Inf, lhs=FALSE)
```

Arguments

`func`	A function providing random data or its name as character.
`type`	The type of `mcnode` to be built. By default, a `"V"` node. see `mcnode` for details.
`...`	All other arguments but the size of the sample to be passed to `func`. These arguments should be vectors or `mcnode`s (arrays prohibited).
`nsv`	The number of simulations in the variability dimension.
`nsu`	The number of simulations in the uncertainty dimension.
`nvariates`	The number of variates of the output.
`outm`	The output of the `mcnode` for multivariates nodes. May be "each" (default) if an output should be provided for each variates considered independently, "none" for no output or a vector of functions (as a character string) that will be applied on the variates dimension before any output (ex: `"mean"`, `"median"`, `c("min","max")`). Each function should return 1 value when applied to 1 value (ex. do not use `"range"`). Note that the `outm` attribute may be changed further using the `outm` function.
`nsample`	The name of the parameter of the function giving the size of the vector. By default, `n`, as in most of the random sampling distributions of the `stats` library (with the exceptions of `rhyper` and `rwilcox` where `nsample="nn"` should be used).
`seed`	The random seed used for the evaluation. If `NULL` the `seed` is unchanged.
`rtrunc`	Should the distribution be truncated? See `rtrunc`.
`linf`	If truncated: lower limit. May be a scalar, an array or a mcnode.
`lsup`	If truncated: upper limit. May be a scalar, an array or a mcnode.
`lhs`	Should a Random Latin Hypercube Sampling be used? see `lhs`

Details

Note that arguments after ... must be matched exactly.

Any function who accepts vectors/matrix as arguments may be used (notably: all current random generator of the `stats` package). The arguments may be sent classically but it is strongly recommended to use consistant `mcnodes` if arguments should be recycled, since a complex recycling is handled for `mcnode` and not for vectors. The rules for compliance of `mcnode` arguments are as following (see below for special functions):

type="V" accepts `"0"` `mcnode` of dimension `(1 x 1 x nvariates)` or of dimension `(1 x 1 x 1)` (recycled) and `"V"` `mcnode` of dimension `(nsv x 1 x nvariates)` or `(nsv x 1 x 1)` (recycled).

type="U" accepts `"0"` `mcnode` of dimension `(1 x 1 x nvariates)` or of dimension `(1 x 1 x 1)` (recycled) and `"U"` `mcnode` of dimension `(1 x nsu x nvariates)` or of dimension `(1 x nsu x 1)` (recycled).

type="VU" accepts "0" mcnode of dimension (1 x 1 x nvariates) or of dimension (1 x 1 x 1) (recycled), "V" mcnode of dimension (nsv x 1 x nvariates) (recycled classicaly) or (nsv x 1 x 1) (recycled classically), "U" mcnode of dimension (1 x nsu x nvariates) (recycled by rows) or (1 x nsu x 1) (recycled by row on the uncertainty dimension and classicaly on variates), "VU" mcnode of dimension (nsv x nsu x nvariates) or of dimension (nsv x nsu x 1) (recycled).

type="0" accepts "0" mcnode of dimension (1 x 1 x nvariates) or (1 x 1 x 1) (recycled).

Multivariate nodes and multivariate distributions:

The number of variates should be provided (not guesses by the function). A multivariates node may be built using a univariate distribution and nvariates!=1. See examples.

rdirichlet needs for alpha a vector or a multivariates nodes and returns a multivariate node. rmultinomial needs for size and prob vectors and/or multivariate nodes and return a univariate or a multivariate node. rmultinormal needs for mean and sigma vectors and/or multivariate nodes and return a multivariate node. rempiricalD needs for values and prob vectors and/or multivariate nodes and return a a univariate or a multivariate node. See examples.

trunc=TRUE is valid for univariates distributions only. The distribution will be truncated on [linf, lsup]. The function 'func' should have a 'q' form (with first argument 'p') and a 'p' form, as all current random generator of the stats library. Example : 'rnorm' (has a 'qnorm' and a 'pnorm' form), 'rbeta', 'rbinom', 'rgamma', ...

If lhs=TRUE, a Random Hypercube Sampling will be used on nsv and nsu The function 'func' should have a 'q' form (with argument 'p'). lhs=TRUE is thus not allowed on multivariates distributions.

Value

An mcnode object.

Author(s)

Regis Pouillot

See Also

mcnode for a description of mcnode object, methods and functions on mcnode objects.

Ops.mcnode for operations on mcnode objects.

Examples

```
Oldnvar <- ndvar()
Oldnunc <- ndunc()
ndvar(5)
ndunc(4)

## compatibility with mcdata as arguments
x0 <- mcstoc(runif, type="0")
xV <- mcstoc(runif, type="V")
xU <- mcstoc(runif, type="U")
xVU <- mcstoc(runif, type="VU")

## "0" accepts mcdata "0"
mcstoc(runif, type="0", min=-10, max=x0)
```

```
## "V" accepts "0" mcdata and "V" mcdata
mcstoc(rnorm, type="V", mean=x0, sd=xV)

## "U" accepts "0" mcdata and "U" mcdata
mcstoc(rnorm, type="U", mean=x0, sd=xU)

## "VU" accepts "0" mcdata, "U" mcdata
## "V" mcdata and "U" mcdata with correct recycling
mcstoc(rnorm, type="VU", mean=x0, sd=xVU)
mcstoc(rnorm, type="VU", mean=xV, sd=xU)

## any function giving a set (vector/matrix) of value of length 'size' works
f <- function(popi) 1:popi
mcstoc(f, type="V", nsample="popi")

##Multivariates

ndvar(2)
ndunc(5)
##Build a multivariate node with univariate distribution
mcstoc(rnorm, "0", nvariates=3)
mcstoc(rnorm, "V", nvariates=3)
mcstoc(rnorm, "U", nvariates=3)
mcstoc(rnorm, "VU", nvariates=3)

##Build a multivariate node with multivariates distribution
alpha <- mcdata(c(1, 1000, 10, 100, 100, 10, 1000, 1), "V", nvariates=4)
(p <- mcstoc(rdirichlet, "V", alpha=alpha, nvariates=4))
mcstoc(rmultinomial, "VU", size=10, p, nvariates=4)

##Build a univariates node with "multivariates" distribution
size <- mcdata(c(1:5), "U")
mcstoc(rmultinomial, "VU", size, p, nvariates=1) #since a multinomial return one value

##Build a multivariates node with "multivariates" distribution
mcstoc(rmultinomial, "VU", size, p, nvariates=4) #sent 4 times to fill the array

##Use of rempiricalD with nodes
##A bootstrap
ndunc(5)
ndvar(5)
dataset <- c(1:9)
(b <- mcstoc(rempiricalD, "U", nvariates=9, values=dataset))
unclass(b)
##Then we build a VU node by sampling in each set of bootstrap
(node <- mcstoc(rempiricalD, "VU", values=b))
unclass(node)

## truncated
ndvar(2)
ndunc(5)
linf <- mcdata(-1:3, "U")
x <- mcstoc(rnorm, "VU", rtrunc=TRUE, linf=linf)
unclass(round(x))
linf <- mcdata(1:5, "U")
mcstoc(rnorm, "VU", nsv=100, rtrunc=TRUE, linf=linf, lhs=TRUE)
```

```
ndvar(Oldnvar)
ndunc(Oldnunc)
```

`multinormal` *The Vectorized Multivariate Random Deviates*

Description

This function is the vectorized version of `rmvnorm`. It provides a random number generator for the multivariate normal distribution with varying vectors of means and varying covariance matrixes.

Usage

```
rmultinormal(n, mean, sigma, method=c("eigen", "svd", "chol"))
```

Arguments

`n`	Number of observations.
`mean`	Vector of means (if unique for all `n`) or array of means (if varying according to `n`).
`sigma`	Covariance vector corresponding to the coercion of the covariance matrix into a vector (if unique for all `n`) or array of covariance vectors (if varying according to `n`).
`method`	Matrix decomposition used to determine the matrix root of sigma, possible methods are eigenvalue decomposition ("eigen", default), singular value decomposition ("svd"), and Cholesky decomposition ("chol").

Details

`rmvnorm(n, m, s)` is equivalent to `rmultinormal(n, m, as.vector(s))`.

If `mean` and/or `sigma` is a matrix, the first random deviate will use the first row of `mean` and/or `sigma`, the second random deviate will use the second row of `mean` and/or `sigma`, ... recycling being permitted by raw. If `mean` is a vector of length `l` or is a matrix with `l` columns, `sigma` should be a vector of length `l^2` or a matrix of number of `l^2` columns.

Note

The use of a varying sigma may be very time consumming.

Examples

```
(mean <- c(10, 0))
(sigma <- matrix(c(1, 2, 2, 10), ncol=2))
(sigma <- as.vector(sigma))
round(rmultinormal(10, mean, sigma))

(mean <- matrix(c(10, 0, 0, 10), ncol=2))
round(rmultinormal(10, mean, sigma))

(mean <- c(10, 0))
(sigma <- matrix(c(1, 2, 2, 10, 10, 2, 2, 1), nrow=2, byrow=TRUE))
```

```
round(rmultinormal(10, mean, sigma))

(mean <- matrix(c(10, 0, 0, 10), ncol=2))
(sigma <- matrix(c(1, 2, 2, 10, 10, 2, 2, 1), nrow=2, byrow=TRUE))
round(rmultinormal(10, mean, sigma))

(mean <- c(10, 0))
(sigma <- matrix(c(1, 2, 2, 10, 10, 2, 2, 1), nrow=2, byrow=TRUE))
round(x <- rmultinormal(1000, mean, sigma))
plot(x)
```

`outm` *Output of Nodes*

Description

Changes the output of Nodes

Usage

```
outm(x, value="each", which.node=1)
```

Arguments

`x`	A `mcnode` or a `mc` object.
`value`	The output of the `mcnode` for multivariates nodes. May be "each" (default) if output should be provided for each variates considered independently, "none" for no output or a vector of name of function(s) (as a character string) that will be applied on the variates dimension before any output (ex: `"mean"`, `"median"`, `c("min","max")`). The function should have no other arguments and send one value per vector of values (ex. do not use `"range"`).
`which.node`	which node should be changed in a `mc` object

Value

`x` with a modified `outm` attribute.

Examples

```
data(total)
total$xVUM2
## since outm = NULL
summary(total$xVUM2)
x <- outm(total$xVUM2, c("min"))
summary(x)
```

`pert` *The Pert Distribution*

Description

Density, distribution function, quantile function and random generation for the pert distribution with minimum equal to `min`, mode equal to `mode` and maximum equal to `max`.

Usage

```
dpert(x, min=-1, mode=0, max=1, shape=4, log=FALSE)
ppert(q, min=-1, mode=0, max=1, shape=4, lower.tail=TRUE, log.p=FALSE)
qpert(p, min=-1, mode=0, max=1, shape=4, lower.tail=TRUE, log.p=FALSE)
rpert(n, min=-1, mode=0, max=1, shape=4)
```

Arguments

`x,q`	Vector of quantiles.
`p`	Vector of probabilities.
`n`	Number of observations. If length(n) > 1, the length is taken to be the number required.
`min`	Vector of minima.
`mode`	Vector of modes.
`max`	Vector of maxima.
`shape`	Vector of scaling parameters.
`log, log.p`	Logical; if `TRUE`, probabilities `p` are given as `log(p)`.
`lower.tail`	Logical; if `TRUE` (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

Details

The Pert distribution is a special case of the Beta distribution specified by the following parameters. Given:

$$\mu = \frac{min + max + shape \times mode}{shape + 2}$$

the values of α_1 and α_2 are

$$\alpha_1 = \frac{(\mu - min)(2 \times mode - min - max)}{(mode - \mu)(max - min)}$$

$$\alpha_2 = \frac{\alpha_1 \times (max - \mu)}{mu - min}$$

on the domain `[min, max]`.

If $\mu = mode$, α_1 is set to $1 + \nu/2$.

Value

`dpert` gives the density, `ppert` gives the distribution function, `qpert` gives the quantile function, and `rpert` generates random deviates.

Author(s)

Regis Pouillot

References

Vose D. Risk Analysis - A Quantitative Guide (John Wiley & Sons, 2000).

See Also

`Beta`

Examples

```
curve(dpert(x, min=3, mode=5, max=10, shape=6), from = 2, to = 11, lty=3)
curve(dpert(x, min=3, mode=5, max=10), from = 2, to = 11, add=TRUE)
curve(dpert(x, min=3, mode=5, max=10, shape=2), from = 2, to = 11, add=TRUE, lty=2)
legend(x = 8, y = 2, c("Default", "shape:2", "shape:6"), lty=1:3)
```

`plot.mc` *Plots Results of a Monte Carlo Simulation*

Description

Plots the empirical cumulative distribution function of a `mcnode` or a `mc` object ("0" and "V" nodes) or the empirical cumulative distribution function of the estimate of a `mcnode` or `mc` object ("U" and "VU" nodes).

Usage

```
## S3 method for class 'mc':
plot(x, prec=0.01, stat=c("median", "mean"), lim=c(0.025, 0.975),
          na.rm=TRUE, griddim=NULL, xlab=NULL, ylab="Fn(x)", main="",
          draw=TRUE, ...)
## S3 method for class 'mcnode':
plot(x, ...)
## S3 method for class 'plotmc':
plot(x, ...)
## S3 method for class 'mccut':
plot(x, stat=c("median", "mean"), lim=c(0.025, 0.975), griddim=NULL,
          xlab=names(x), ylab="Fn(x)", main="", draw=TRUE, ...)
```

Arguments

`x`	a `mcnode` or a `mc` objects
`prec`	the precision of the plot. 0.01 will provide an ecdf from the 0.00, 0.01, .02, ..., 1.00 quantiles, 0.001 will provide a 0.000, 0.001, 0.002, ..., 1.000 quantiles,...
`stat`	the function used for estimates (2D `mc` or `mcnode`). By default the median.
`lim`	a vector of numbers (between 0 and 1) indicating the enveloppe (2D `mc` or `mcnode`) . Maybe `NULL` or empty.
`na.rm`	Should NA values be discarded
`griddim`	a vector of two integers, indicating the size of the grid of the graph. If `NULL`, the grid is calculated to produce a "nice" graph.
`xlab`	vector of labels for the x-axis. If `NULL`, use the name of the node.
`ylab`	vector of labels for the y-axis.
`main`	vector of main titles of the graph.
`draw`	Should the plot be drawn?
`...`	further arguments to be passed to `plot.ecdf`.

Details

`plot.mcnode` is a user-friendly function that send the `mcnode` to `plot.mc`.

For `"VU"` and `"U"` `mcnodes`, quantiles are calculated using `quantile.mc` within each of the `nsu` simulations (i.e. by columns of each `mcnode`). The medians (but may be the means using `stat="mean"`) calculated from the `nsu` values are plotted. The 0.025 and 0.975 quantiles (default values of `lim`) of these quantiles are used as the enveloppe.

Value

A `plot.mc` object, list of the quantiles used to plot the draw.

Author(s)

Regis Pouillot

References

Cullen AC and Frey HC (1999) Probabilistic techniques in exposure assessment. Plenum Press, USA, pp. 81-155.

See Also

`ecdf`, `plot`, `quantile.mc`

Examples

```
data(total)
plot(xVUM3)
plot(total)
```

plot.tornado *Draws a Tornado chart.*

Description

Draws a Tornado chart as provided by `tornado`.

Usage

```
## S3 method for class 'tornado':
plot(x, which=1, name=NULL, stat=c("median", "mean"), xlab="method",
          ylab="", ...)
## S3 method for class 'tornadounc':
plot(x, which=1, stat=2, name=NULL, xlab="method", ylab="", ...)
```

Arguments

`x`	A `tornado` object or a `tornadounc` object.
`which`	Which output to print -for multivariates output-.
`name`	Vector of name of input variables. If NULL, the name will be given from the name of the elements.
`stat`	The name (or the number of column) of the statistics of the output to be considered.
`xlab`	Label of the x axis. if "method", use the correlation method used in the `tornado` object.
`ylab`	Label of the y axis.
`...`	Further arguments to be passed to the `plot` function.

Details

A point is drawn at the estimate and the segment reflects the uncertainty around this estimate.

Value

NULL

Author(s)

Regis Pouillot

See Also

`tornado`

print.mc *Prints a mcnode or a mc Object*

Description

Print a description of the structure of the `mc` or the `mcnode` object.

Usage

```
## S3 method for class 'mc':
print(x, digits=3, ...)
## S3 method for class 'mcnode':
print(x, ...)
```

Arguments

`x`	a `mcnode` or a `mc` object.
`digits`	Number of digits to be used.
`...`	Further arguments to be passed to the print function.

Value

An invisible data frame.

Author(s)

Regis Pouillot

See Also

`mcnode` for `mcnode` objects. `mc` for `mc` objects.

quantile.mc *Quantiles of a mc Object*

Description

Evaluates quantiles of a `mc` object. This function is used by `plot.mc`

Usage

```
## S3 method for class 'mc':
quantile(x, probs=seq(0, 1, 0.01), lim=c(0.025, 0.975), na.rm=TRUE, ...)
## S3 method for class 'mcnode':
quantile(x, ...)
```

Arguments

`x`	a `mc` objects
`probs`	the quantiles to be calculated
`na.rm`	TRUE or FALSE
`lim`	a vector of numbers (between 0 and 1) indicating the enveloppe. Maybe `NULL` or empty.
`...`	For generic method consistancy.

Details

The quantiles are evaluated in the variability dimension. Then, the median, the mean and the `lim` quantiles are evaluated for each of these quantiles.

Value

A list of quantiles.

Author(s)

Regis Pouillot

See Also

`plot.mc`, `quantile`.

Examples

```
data(total)
quantile(total$xVUM3)
quantile(total)
```

`rtrunc` *Random Truncated Distributions*

Description

Provides samples from classical R distributions and `mc2d` specific distributions truncated between `linf` and `lsup`.

Usage

```
rtrunc(distr=runif, n, linf=-Inf, lsup=Inf, ...)
```

Arguments

`distr`	A function providing random data or its name as character. The function 'rdistr' should have a 'qdistr' form (with argument 'p') and a 'pdistr' form (with argument 'q'). Example : 'rnorm' (has a 'qnorm' and a 'pnorm' form), 'rbeta', 'rbinom', 'rgamma', ...
`n`	The size of the sample.
`linf`	A vector of lower bounds.
`lsup`	A vector of upper bounds.
`...`	All arguments to be passed to `pdistr` and `qdistr`.

Details

The function 1) evaluates the `p` values corresponding to `linf` and `lsup` using `pdistr`; 2) samples `n` values using `runif(n, min=pinf, max=psup)`, and 3) takes the `n` corresponding quantiles from the specified distribution using `qdistr`.

All distributions (but sample) implemented in the stats library could be used. The arguments in ... should be named. Do not use 'log' or 'log.p' or 'lower.tail'.

Value

A vector of `n` values.

Note

The inversion of the quantile function leads to time consuming functions for some distributions.

Author(s)

Regis Pouillot

Examples

```
rtrunc("rnorm", n=10, linf=0)
range(rtrunc(rnorm, n=1000, linf=3, lsup=5, sd=10))
```

`summary.mc` *Summary of mcnode and mc Object*

Description

Provides a summary of a `mcnode`, a `mc` or a `mccut` object.

Usage

```
## S3 method for class 'mc':
summary(object, probs=c(0, 0.025, 0.25, 0.5, 0.75, 0.975, 1), lim=c(0.025,
          0.975), ...)
## S3 method for class 'mcnode':
summary(object, probs=c(0, 0.025, 0.25, 0.5, 0.75, 0.975, 1), lim=c(0.025,
          0.975), digits=3, ...)
## S3 method for class 'mc':
print.summary(x, digits=3, ...)
## S3 method for class 'mccut':
summary(object, lim=c(0.025, 0.975), ...)
```

Arguments

`object`	a `mcnode` or a `mc` object or a `mccut` object.
`x`	A `summary.mc` object as provided by the `summary.mc` function.
`probs`	A vector of values used for the quantile function (variability dimension).
`digits`	Number of digits in the print.
`lim`	A vector of values used for the quantile function (uncertainty dimension).
`...`	For generic functions consistancy.

Details

The mean, the standard deviation and the `probs` quantiles will be evaluated in the variability dimension. The median, the mean and the `lim` quantiles will then be evaluated on these statistics in the uncertainty dimension.

Multivariate nodes:

If the `"outm"` attributes of the mcnode is "none", the node is not evaluated, if it is "each" the variates are evaluated one by one, if it is a function (e.g. "mean"), the function is applied on the `nvariates` dimension before providing a classical output.

Value

a list.

Author(s)

Regis Pouillot

See Also

`mcnode` for mcnode objects, `mc` for mc objects, `mccut` for mccut objects, `quantile`

Examples

```
data(total)
summary(xVUM3)
summary(total)
```

`tornado` *Computes Correlation between Inputs and Output in a mc Object (tornado) in the Variability Dimension;*

Description

Provides statistics for a tornado chart. Evaluates correlations between output and inputs of a `mc` object.

Usage

```
tornado(mc, output=length(mc), use="all.obs", method=c("spearman",
           "kendall", "pearson"), lim=c(0.025, 0.975))
## S3 method for class 'tornado':
print(x, ...)
```

Arguments

`mc`	a `mc` object or a `mccut` object.
`x`	A `tornado` object as provided by the `tornado` function.
`output`	(for `mc` objects only). The rank or the name of the output to be considered. By default: the last element of the `mc`.
`use`	(for `mc` objects only). An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "all.obs", "complete.obs" or "pairwise.complete.obs" (see `cor`).
`method`	(for `mc` objects only). A character string indicating which correlation coefficient (or covariance) is to be computed. One of "spearman" (default), "kendall" or "pearson", can be abbreviated (see `cor`). Warning : the default is not the same in `cor`.
`lim`	A vector of quantiles used to compute the credible interval in two-dimensional models.
`...`	Further arguments to be passed to the final print function.

Details

The tornado function computes the spearman's rho statistic. It is used to estimate a rank-based measure of association between one set of random variable of a `mc` object (the output) and the others (the inputs).

`tornado` may be applied on a `mccut` object if a `tornado` function was used in the third block of the `evalmccut` call.

If "output" refers to a `"0" mcnode`, it is an error. If "output" refers to a `"V" mcnode`, correlations are only provided for other `"V" mcnodes`. If "output" refers to a `"U" mcnode`, correlations are only provided for other `"U" mcnodes`. If "output" refers to a `"VU" mcnode`, correlations are only provided for other `"VU" mcnodes` and `"V" mcnodes`.

If use is "all.obs", then the presence of missing observations will produce an error. If use is "complete.obs" then missing values are handled by casewise deletion. Finally, if use has the value "pairwise.complete.obs" then the correlation between each pair of variables is computed using all complete pairs of observations on those variables.

Value

An invisible object of class tornado. A tornado object is a list of objects containing the following objects:

`value`	the value of correlation coefficients
`output`	the name of the output
`method`	the method used
`use`	the use parameter

Author(s)

Regis Pouillot

See Also

`cor`.

`plot.tornado` to draw the results.

Examples

```
data(total)
tornado(total, 2, "complete.obs", "spearman", c(0.025, 0.975))
tornado(total, 4, "pairwise.complete.obs", "spearman", c(0.025, 0.975))
tornado(total, 6, "complete.obs", "kendall", c(0.025, 0.975))
tornado(total, 8, "complete.obs", "spearman", c(0.025, 0.975))
(y <- tornado(total, 10, "complete.obs", "spearman", c(0.025, 0.975)))
plot(y)
```

`tornadounc` *Computes Correlation between Inputs and Output in a mc Object (tornado) in the Uncertainty Dimension*

Description

Provides statistics for a tornado chart. Evaluates correlations between output and inputs of a `mc` object in the uncertainty dimension.

Usage

```
## S3 method for class 'mc':
tornadounc(mc, output=length(mc), quant=c(0.5, 0.75, 0.975), use="all.obs",
          method=c("spearman", "kendall", "pearson"), ...)
## S3 method for class 'tornadounc':
print(x, ...)
## S3 method for class 'mccut':
tornadounc(mc, output=length(mc), quant=c(0.5, 0.75, 0.975), use="all.obs",
          method=c("spearman", "kendall", "pearson"), ...)
```

Arguments

`mc`	a `mc` object.
`x`	a `tornadounc` object.
`output`	The rank or the name of the output to be considered. Should be a `"VU"` or a `"U" type mcnode`. By default: the last element of `mc`.
`quant`	The vector of quantiles used in the variability dimension.
`use`	An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "all.obs", "complete.obs" or "pairwise.complete.obs" (see `cor`).
`method`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "spearman" (default), "kendall" or "pearson", can be abbreviated (see `cor`). Warning : "pearson" is the default for `cor`).
`...`	Further arguments to be passed to the final print function.

Details

The `tornadounc.mc` function computes the spearman's rho statistic between

- values (`"U" type mcnode`) or statistics calculated in the variability dimension (`"VU" type mcnode`) of inputs and
- values (`"U" type mcnode`) or statistics calculated in the variability dimension (`"VU" type mcnode`) of one output.

The statistics are the mean, the median and the quantiles specified by `quant`.

It is useful to estimate a rank-based measure of association between one set of random variable of a `mc` object (the output) and the others in the uncertainty dimension.

`tornadounc.mccut` may be applied on a `mccut` object if a `summary.mc` function was used in the third block of the `evalmccut` call.

If output refers to a `"0"` or `"V" mcnode`, it is an error.

If use is "all.obs", then the presence of missing observations will produce an error. If use is "complete.obs" then missing values are handled by casewise deletion. Finally, if use has the value "pairwise.complete.obs" then the correlation between each pair of variables is computed using all complete pairs of observations on those variables.

Value

An invisible object of class `tornadounc`. A `tornadounc` object is a list of objects containing the following objects:

`value`	a matrix of values of correlation coefficients. Each row are the value or the statistics of inputs, each columns the value or the statistics of outputs.
`output`	the name of the output
`method`	the method used
`use`	the `use` parameter

Author(s)

Regis Pouillot

See Also

cor.

tornado for tornado in the variability dimension.

plot.tornadounc to draw the results.

Examples

```
data(total)
tornadounc(total, 3)
tornadounc(total, 4, use="complete")
tornadounc(total, 7, use="complete.obs")
tornadounc(total, 8, use="complete.obs")
(y <- tornadounc(total, 10, use="complete.obs"))
plot(y, 1, 1)
```

---

total *An Exemple of all Kind of mcnode*

Description

An exemple for each kind of mcnodes. They are used in some mc2d examples. They have been built using the following code:

```
ndvar(101) ndunc(51)

x0 <- mcstoc(type="0")

xV <- mcstoc(type="V")

xU <- mcstoc(type="U")

xVU <- mcstoc(type="VU")

x0M <- mcstoc(type="0",nvariates=2)

xVM <- mcstoc(type="V",nvariates=2)

xUM <- mcstoc(type="U",nvariates=2)

xVUM <- mcstoc(type="VU",nvariates=2)

xVUM[c(1,12,35)] <- NA

xVUM2 <- mcstoc(type="VU",nvariates=2,outm="none")

xVUM3 <- mcstoc(type="VU",nvariates=2,outm=c("mean","min"))

total <- mc(x0,xV,xU,xVU,x0M,xVM,xUM,xVUM,xVUM2,xVUM3)
```

Usage

```
total
```

Format

Some mcnode objects and one mc object.

Source

None

References

None

triangular *The Triangular Distribution*

Description

Density, distribution function, quantile function and random generation for the triangular distribution with minimum equal to `min`, mode equal `mode` and maximum equal to `max`.

Usage

```
dtriang(x, min=-1, mode=0, max=1, log=FALSE)
ptriang(q, min=-1, mode=0, max=1, lower.tail=TRUE, log.p=FALSE)
qtriang(p, min=-1, mode=0, max=1, lower.tail=TRUE, log.p=FALSE)
rtriang(n, min=-1, mode=0, max=1)
```

Arguments

`x,q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If length(n) >1, the length is taken to be the number required.
`min`	vector of minima.
`mode`	vector of modes.
`max`	vector of maxima.
`log, log.p`	logical; if `TRUE`, probabilities `p` are given as `log(p)`.
`lower.tail`	logical; if `TRUE` (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

Value

`dtriang` gives the density, `ptriang` gives the distribution function, `qtriang` gives the quantile function, and `rtriang` generates random deviates.

Author(s)

Regis Pouillot

Examples

```
curve(dtriang(x, min=3, mode=5, max=10), from = 2, to = 11)
```

`typemcnode` *Provides the Type of a mcnode Object*

Description

Provide the type of a `mcnode` object.

Usage

```
typemcnode(x, index=FALSE)
```

Arguments

`x`	a `mcnode` object
`index`	if `TRUE` give the index of the type rather than the type.

Value

`"0", "V","U" or "VU"` or the corresponding index if `index=TRUE`.

`NULL` if none of this element is found.

Note

This function does not test if the object is correct. See `is.mcnode`.

Author(s)

Regis Pouillot

Examples

```
data(total)
typemcnode(total$xVUM2)
```

`unmc` *Unclasses the mc or the mcnode Object*

Description

Unclasses the `mc` object in a list of arrays or the `mcnode` object in an array.

Usage

```
unmc(x, drop=TRUE)
```

Arguments

`x`	A `mc` or a `mcnode` object.
`drop`	Should the dimensions of size 1 be dropped (see `drop`).

Value

if x is an `mc` object: a list of arrays. If `drop=TRUE`, a list of vectors, matrixes and arrays. if x is an `mcnode` object: an array. If `drop=TRUE`, a vector, matrix or array.

Author(s)

Regis Pouillot

Examples

```
data(total)
## A vector
unmc(total$xV, drop=TRUE)
## An array
unmc(total$xV, drop=FALSE)
```

Index