Package 'mc2d'

June 1, 2026

Type Package

Title Tools for Two-Dimensional Monte-Carlo Simulations

Version 0.2.2

Date 2026-05-12

Suggests fitdistrplus, survival, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 2.10.0), mvtnorm

Imports stats, grDevices, graphics, utils, ggplot2, ggpubr

Description A complete framework to build and study Two-Dimensional Monte-Carlo simulations, aka Second-Order Monte-Carlo simulations. Also includes various distributions (pert, triangular, Bernoulli, empirical discrete and continuous).

License GPL (>= 2)

NeedsCompilation no

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Author Regis Pouillot [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6107-5212>),
Marie-Laure Delignette-Muller [ctb] (ORCID:
<https://orcid.org/0000-0001-5453-3994>),
Jean-Baptiste Denis [ctb],
Yu Chen [ctb],
Arie Havelaar [ctb] (ORCID: <https://orcid.org/0000-0002-6456-5460>)

Maintainer Regis Pouillot <rpouillot@yahoo.fr>

Repository CRAN

Date/Publication 2026-06-01 19:10:08 UTC

Contents

`mc2d-package` *mc2d: Tools for Two-Dimensional Monte-Carlo Simulations*

Description

A complete framework to build and study Two-Dimensional Monte-Carlo simulations, aka Second-Order Monte-Carlo simulations. Also includes various distributions (pert, triangular, Bernoulli, empirical discrete and continuous).

Author(s)

Maintainer: Regis Pouillot <rpouillot@yahoo.fr> (ORCID)

Authors:

- Regis Pouillot <rpouillot@yahoo.fr> (ORCID)

Other contributors:

- Marie-Laure Delignette-Muller <ml.delignette@vetagro-sup.fr> (ORCID) [contributor]
- Jean-Baptiste Denis <jbdenis@jouy.inra.fr> [contributor]
- Yu Chen <chen0099yu@gmail.com> [contributor]
- Arie Havelaar <ariehavelaar@ufl.edu> (ORCID) [contributor]

`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 = 0.5, log = FALSE)

pbern(q, prob = 0.5, lower.tail = TRUE, log.p = FALSE)

qbern(p, prob = 0.5, lower.tail = TRUE, log.p = FALSE)

rbern(n, prob = 0.5)
```

Arguments

x, q	vector of quantiles.
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]'.
p	vector of probabilities.
n	number of observations. If 'length(n) > 1', the length is taken to be the number required.

Details

These functions 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.

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.
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]'.
p	vector of probabilities.
n	number of observations. If 'length(n) > 1', the length is taken to be the number required.

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 parameterization.

Value

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

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)
```

BetaSubjective *The BetaSubjective Distribution*

Description

Density, distribution function, quantile function and random generation for the "Beta Subjective" distribution

Usage

```
dbetasubj(x,
  min,
  mode,
  mean,
  max,
  log = FALSE)

pbetasubj(q,
  min,
  mode,
  mean,
  max,
  lower.tail = TRUE,
  log.p = FALSE
)

qbetasubj(p,
  min,
  mode,
  mean,
  max,
  lower.tail = TRUE,
  log.p = FALSE
)

rbetasubj(n,
  min,
  mode,
```

```
  mean,
  max
)

pbetasubj(q, min, mode, mean, max, lower.tail = TRUE, log.p = FALSE)

qbetasubj(p, min, mode, mean, max, lower.tail = TRUE, log.p = FALSE)

rbetasubj(n, min, mode, mean, max)
```

Arguments

x, q	Vector of quantiles.
min	continuous boundary parameter min < max
mode	continuous parameter $min < mode < max$ and $mode \neq mean$.
mean	continuous parameter min < mean < max
max	continuous boundary parameter
log, log.p	Logical; if TRUE, probabilities p are given as log(p).
lower.tail	Logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
p	Vector of probabilities.
n	Number of observations.

Details

The Subjective beta distribution specifies a [stats::dbeta()] distribution defined by the minimum, most likely (mode), mean and maximum values and can be used for fitting data for a variable that is bounded to the interval $[min, max]$. The shape parameters are calculated from the mode value and mean parameters. It can also be used to represent uncertainty in subjective expert estimates.

Define

$$mid = (min + max)/2$$

$$a_1 = 2 * \frac{(mean - min) * (mid - mode)}{((mean - mode) * (max - min))}$$

$$a_2 = a_1 * \frac{(max - mean)}{(mean - min)}$$

The subject beta distribution is a [stats::dbeta()] distribution defined on the $[min, max]$ domain with parameter $shape1 = a_1$ and $shape2 = a_2$.

Hence, it has density

$$f(x) = (x - min)^{(a_1 - 1)} * (max - x)^{(a_2 - 1)} / (B(a_1, a_2) * (max - min)^{(a_1 + a_2 - 1)})$$

The cumulative distribution function is

$$F(x) = B_z(a_1, a_2)/B(a_1, a_2) = I_z(a_1, a_2)$$

where $z = (x - min)/(max - min)$. Here B is the beta function and B_z is the incomplete beta function.

The parameter restrictions are:

$$min <= mode <= max$$

$$min <= mean <= max$$

If $mode > mean$ then $mode > mid$, else $mode < mid$.

Author(s)

Yu Chen

Examples

```
curve(dbetasubj(x, min=0, mode=1, mean=2, max=5), from=-1,to=6)
pbetasubj(q = seq(0,5,0.01), 0, 1, 2, 5)
qbetasubj(p = seq(0,1,0.01), 0, 1, 2, 5)
rbetasubj(n = 1e7, 0, 1, 2, 5)
```

---

converg *Graph of Running Statistics for Convergence Diagnostics*

Description

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, mc or mccut object.
node	the node to be considered in a 'mc' or 'mccut' object, either as order number or name. By default: the last node. Cannot be of type '"0"'.
margin	the margin used to plot the graph. Used only if the node is a '"VU"' node.

nvariates	the variate to be considered. Used only for multivariate nodes.
iter	if 'margin=="var"' and the node is '"VU"', specifies which iteration in the uncertainty dimension to use.
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 for the grid size. If 'NULL', calculated automatically.
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 is provided for each of the mean, median and 'probs' quantiles calculated in the variability dimension.

Value

'invisible(NULL)' (called for its side effect of drawing a plot).

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' in 'converg' should have been used in that 'summary.mc'.

Examples

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

---

cornode *Build a Rank Correlation Using the Iman and Conover Method*

Description

Builds a rank correlation structure between columns of a matrix or between 'mcnode' objects using the Iman and Conover (1982) method.

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 element but the first will be reordered). Arguments should be named.
target	a scalar (only if 'n=2') or a '(n x n)' matrix of correlations.
outrank	should the order be returned?
result	should the correlation eventually obtained be printed?
seed	the random seed for building the correlation. If 'NULL' the seed is unchanged.

Details

The function accepts for 'data':

- some '"V" mcnode' objects separated by a comma;
- some '"U" mcnode' objects separated by a comma;
- some '"VU" mcnode' objects, building the structure column by column;
- one '"V" mcnode' as first element and some '"VU" mcnode' objects.

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

The final correlation structure should be checked, as it is not always possible to achieve the target exactly. The order of values within each 'mcnode' (except the first) will be changed by this function. The 'outrank' option may help to rebuild prior links.

Value

If 'outrank=FALSE': the matrix or a list of rearranged 'mcnode's. If 'outrank=TRUE': the order to be used to rearrange to build the desired correlation.

References

Iman, R. L., & Conover, W. J. (1982). A distribution-free approach to inducing rank correlation among input variables. *Communication in Statistics - Simulation and Computation*, 11(3), 311-334.

Examples

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

dimmcnode *Dimension of mcnode and mc Objects*

Description

Provides the dimension of a 'mcnode' or a 'mc' object: the number of simulations in the variability dimension, in the uncertainty dimension, and the maximum number of variates.

Usage

```
dimmcnode(x)

dimmc(x)

typemcnode(x, index = FALSE)

is.mc(x)

is.mcnode(x)
```

Arguments

x	a 'mcnode' or a 'mc' object.
index	if 'TRUE', give the index of the type rather than the type (for 'typemcnode' only).

Value

'dimmcnode': a vector of three scalars (nsv, nsu, nvariates). 'dimmc': a vector of three scalars (max nsv, max nsu, max variates). 'typemcnode': '"0"', '"V"', '"U"' or '"VU"', or the corresponding index if 'index=TRUE'. 'NULL' if none found. 'is.mc': 'TRUE' or 'FALSE'. 'is.mcnode': 'TRUE' or 'FALSE'.

Note

These functions do not test whether the object is correctly built. See is.mcnode and is.mc.

See Also

mcnode, mc.

Examples

```
data(total)
dimmcnode(xVUM2)
dimmc(total)
typemcnode(total$xVUM2)
is.mcnode(xVU)
is.mc(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 multivariate '`mcnode`' objects.

Value

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

Note

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)
```

`dmultinomial` *The Vectorized Multinomial Distribution*

Description

Generate multinomially distributed random number vectors and compute multinomial probabilities. These functions are the vectorized versions of `rmultinom` and `dmultinom`. Recycling is permitted.

Usage

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

rmultinomial(n, size, prob)
```

Arguments

`x`	vector or matrix of length (or ncol) K of integers in '`0:size`'.
`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 '`dmultinomial`', it defaults to '`sum(x)`'. The first element corresponds to the vector '`prob`' or the first row of '`prob`', ...
`prob`	numeric non-negative vector of length K, or matrix of size '`(n x K)`' specifying the probability for the K classes; is internally normalized to sum 1.
`log`	logical; if '`TRUE`', log probabilities are computed.
`n`	number of random vectors to draw.

Examples

```
x <- c(100, 200, 700)
x1 <- matrix(c(100,200,700, 200,100,700, 700,200,100), byrow=TRUE, ncol=3)
p <- c(1, 2, 7)
p1 <- matrix(c(1,2,7, 2,1,7, 7,2,1), byrow=TRUE, ncol=3)
dmultinomial(x1, prob=p)
## is equivalent to
c(dmultinom(x1[1,], prob=p),
  dmultinom(x1[2,], prob=p),
  dmultinom(x1[3,], prob=p))

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

ec *An example 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

```
data(ec)
```

Format

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

Source

Fictive example

References

None

empiricalC *The Continuous Empirical Distribution*

Description

Density, distribution function, quantile function and random generation for a continuous empirical distribution.

Usage

```
dempiricalC(x, min, max, values, prob = NULL, log = FALSE)

pempiricalC(q, min, max, values, prob = NULL, lower.tail = TRUE, log.p = FALSE)

qempiricalC(p, min, max, values, prob = NULL, lower.tail = TRUE, log.p = FALSE)

rempiricalC(n, min, max, values, prob = NULL)
```

Arguments

x, q	vector of quantiles.
min	a finite minimal value.
max	a finite maximal value.
values	vector of numerical values.
prob	optional vector of counts or probabilities.
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]'.
p	vector of probabilities.
n	number of random values. If 'length(n) > 1', the length is taken to be the number required.

Details

Given p_i, the distribution value for x_i with 'i' the rank $i = 0, 1, 2, \ldots, N+1$, $x_0 = min$ and $x_{N+1} = max$, the density is:

$$f(x) = p_i + \frac{(p_{i+1} - p_i)(x - x_i)}{x_{i+1} - x_i}$$

The 'p' values are normalized to give the distribution a unit area.

'min' and/or 'max' and/or 'values' and/or 'prob' may vary: in that case, 'min' and/or 'max' should be vector(s). 'values' and/or 'prob' should be matrices, 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, ... Recycling is permitted if the number of elements of 'min' or 'max' or the number of rows of 'prob' and 'values' are equal or equal to one.

Value

'dempiricalC' gives the density, 'pempiricalC' gives the distribution function, 'qempiricalC' gives the quantile function and 'rempiricalC' generates random deviates.

See Also

empiricalD

Examples

```
prob <- c(2, 3, 1, 6, 1)
values <- 1:5
par(mfrow = c(1, 2))
curve(dempiricalC(x, min=0, max=6, values, prob), from=-1, to=7, n=1001)
curve(pempiricalC(x, min=0, max=6, values, prob), from=-1, to=7, n=1001)

## Varying values
(values <- matrix(1:10, ncol=5))
dempiricalC(c(1, 1), values, min=0, max=11)
```

---

empiricalD *The Discrete Empirical Distribution*

Description

Density, distribution function, quantile 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.
values	vector or matrix of numerical values. See details.
prob	optional vector or matrix of counts 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]'.
p	vector of probabilities.
n	number of random values. If 'length(n) > 1', the length is taken to be the number required.

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 matrices, 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, ... 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' is 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 functions should be written for non numerical values.

See Also

sample, empiricalC.

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))
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 variability dimension.
nsu	the number of simulations in the uncertainty dimension.
seed	the random seed. If 'NULL' the seed is unchanged.

Details

The model is evaluated. Intermediate variables used to build the 'mc' object are not stored.

Value

The result of the evaluation — should be a 'mc' object.

Note

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

See Also

mcmodel, evalmccut to evaluate high-dimension models.

Examples

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

---

extractvar *Utilities for Multivariate Nodes*

Description

'extractvar' extracts one or more variates from a multivariate 'mcnode'. 'addvar' combines consistent 'mcnode's into a single multivariate 'mcnode'.

Usage

```
extractvar(x, which = 1)

addvar(...)
```

Arguments

x	a multivariate 'mcnode'.
which	a vector specifying which variate(s) to extract.
...	'mcnode's to be combined in a multivariate 'mcnode'. These 'mcnode's should be of the same type and dimension.

Details

The 'outm' attribute of the output of 'addvar' is taken from the first element.

Value

The new 'mcnode'.

See Also

mcnode for 'mcnode' objects.

Examples

```
x <- mcdata(0:3, "0", nvariates = 4)
y <- extractvar(x, c(1, 3))
y
addvar(x, y)
```

---

gghist *Histogram of a Monte Carlo Simulation (ggplot version)*

Description

Shows histogram of a 'mcnode' or a 'mc' object by ggplot framework.

Usage

```
gghist(x, ...)

## S3 method for class 'mcnode'
gghist(
  x,
  griddim = NULL,
  xlab = names(x),
  ylab = "Frequency",
  main = "",
  bins = 30,
  which = NULL,
  ...
)
```

```
## S3 method for class 'mc'
gghist(
  x,
  griddim = NULL,
  xlab = names(x),
  ylab = "Frequency",
  main = "",
  bins = 30,
  ...
)
```

Arguments

x	an ‘mc‘ or an ‘mcnode‘ object
...	Further arguments to be passed to geom_histogram()
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
bins	Number of bins. Defaults to 30.
which	An argument used for a multivariate ‘mcnode‘. Can specify which variate plot to display. When variates are more than one, the output will be saved in a plot list by default or use the number of which variate to display.

Value

a ggplot object.

Author(s)

Yu Chen and Regis Pouillot

See Also

[hist.mc()]

Examples

```
data(total)
# When mcnode has one variate
gghist(xV)
# When mcnode has two variates, the two plots will be saved in a list
# if affected to a variable
gplots <- gghist(xVUM)
# show the first variate plot of xVUM mcnode
gplots[[1]]
```

```
# directly show the first variate plot of xVUM mcnode
gghist(xVUM, which = 1) #directly show the first variate plot of xVUM mcnode
# Post process
gplots[[1]] + ggplot2::geom_histogram(color = "red",fill="blue")
```

---

ggplotmc *ggplotmc*

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) based on [ggplot2::ggplot] package.

Usage

```
ggplotmc(x, ...)

## S3 method for class 'mcnode'
ggplotmc(
  x,
  prec = 0.001,
  stat = c("median", "mean"),
  lim = c(0.025, 0.25, 0.75, 0.975),
  na.rm = TRUE,
  griddim = NULL,
  xlab = NULL,
  ylab = "Fn(x)",
  main = "",
  paint = TRUE,
  xlim = NULL,
  ylim = NULL,
  which = NULL,
  ...
)

## S3 method for class 'mc'
ggplotmc(
  x,
  prec = 0.001,
  stat = c("median", "mean"),
  lim = c(0.025, 0.25, 0.75, 0.975),
  na.rm = TRUE,
  griddim = NULL,
  xlab = NULL,
  ylab = "Fn(x)",
```

```
  main = "",
  paint = TRUE,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

x	and ‘mc‘ or an ‘mcnode‘ object
...	further arguments to be passed to [ggplot2::stat_ecdf()]
prec	the precision of the plot. 0.001 will provide an ecdf using the 0.000, 0.001, .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 envelope (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‘, the name of the node is used.
ylab	vector of labels for the y-axis.
main	vector of main titles of the graph
paint	Should the envelopes be filled?
xlim	x coordinate range. ‘xlim‘ is either a vector of length 2, used for each graph, or a list of vectors of length 2, whose ith element is used for the ith graph. By default, the data range is used as xlim.
ylim	y coordinate range. ‘ylim‘ is either a vector of length 2, used for each graph, or a list of vectors of length 2, whose ith element is used for the ith graph. By default, the data range is 0-1.
which	An argument used for an ‘mcnode‘ with multivariates. Can specify which variate plot to display. When variates are more than one, the output will be saved in a plot list by default or use the number of which variate to display.

Value

a ggplot object.

Author(s)

Yu Chen and Regis Pouillot

See Also

[plot.mc()]

Examples

```
data(total)
# When mcnode has one variate
ggplotmc(xV)
# Post process
ggplotmc(xV) + ggplot2::ggtitle("post processed")
# When mcnode has two variates
gplots <- ggplotmc(xVUM) #will save two plots in a list
gplots[[1]] # show the first variate plot of xVUM mcnode
ggplotmc(xVUM, which = 1) #directly show the first variate plot of xVUM mcnode
```

---

ggspaghetti *Spaghetti Plot of 'mc' or 'mcnode' Object*

Description

Use ggplot to draw spaghetti plots for the [mc] or [mcnode] objects.

Usage

```
ggspaghetti(x, ...)

## S3 method for class 'mc'
ggspaghetti(
  x,
  griddim = NULL,
  xlab = names(x),
  ylab = "F(n)",
  main = "",
  maxlines = 100,
  ...
)

## S3 method for class 'mcnode'
ggspaghetti(
  x,
  griddim = NULL,
  xlab = names(x),
  ylab = "F(n)",
  main = "",
  which = NULL,
  maxlines = 100,
  ...
)
```

Arguments

x	an ‘mc‘ or an ‘mcnode‘ object
...	further arguments to be passed to [ggplot2::stat_ecdf()]
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
maxlines	the maximum number of ecdf to draw.
which	An argument used for an ‘mcnode‘ with multivariates. Can specify which variate plot to display. When variates are more than one, the output will be saved in a plot list by default or use the number of which variate to display.

Author(s)

Yu Chen and Regis Pouillot

Examples

```
data(total)
# mcnode has one variate
```

---

ggtornado *Draws a Tornado chart as provided by tornado (ggplot version).*

Description

Draws a Tornado chart as provided by tornado.

Usage

```
## For class 'tornado'
ggtornado(x,
  which=1,
  name=NULL,
  stat=c("median","mean"),
  xlab="method",
  ylab=""
)

## For class 'tornadounc'
ggtornadounc(x,
  which=1,
  stat="median",
```

```
  name=NULL,
  xlab="method",
  ylab=""
)

ggtornadounc(
  x,
  which = 1,
  stat = "median",
  name = NULL,
  xlab = "method",
  ylab = ""
)
```

Arguments

x	A tornado object as provided by the tornado function.
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 of the statistics of the output to be considered. For a tornado object: "median" or "mean". For a tornadounc object: the value should match one row name of the tornadounc object. Alternatively, for a tornadounc object, the number of the row may be used.
xlab	Label of the x axis. Default is to use the correlation method used in the tornado object.
ylab	Label of the y axis. Default is empty.

See Also

tornado

Examples

```
data(ec)
x <- evalmcmod(ec$modEC2, nsv=100, nsu=100, seed=666)
tor <- tornado(x, 7)
ggtornado(tor)
data(total)
ggtornado(tornadounc(total, 10, use="complete.obs"), which=1)
```

`hist.mc` *Histogram of a Monte Carlo Simulation*

Description

Shows a 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. May be recycled.
`ylab`	a vector of labels for the y-axis. May be recycled.
`main`	a vector of main titles of histograms. May be recycled.
`...`	other arguments to be passed to all calls of 'hist'.

Value

'`invisible()`' (called for side effects).

Note

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

See Also

`gghist` for a ggplot2 version.

Examples

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

`lhs` *Random Latin Hypercube Sampling*

Description

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

Usage

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

Arguments

`distr`	the function for generating random samples, or its name as a character string. If '`distr`' is '`"rdist"`', the quantile function '`"qdist"`' must exist with argument '`p`' as a vector of probabilities.
`nsv`	the number of rows 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 a latin hypersquare sampling: the LHS structure is provided only in the first 2 dimensions. It is not possible to use truncated distributions directly with `rtrunc`. Use `mcstoc` with '`lhs=TRUE`' and '`rtrunc=TRUE`' for that purpose. The ... arguments will be recycled.

Adapted from a code by Rob Carnell (package **lhs**).

See Also

`mcstoc`

Examples

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

Lognormalb *The Log Normal Distribution parameterized through its mean and standard deviation.*

Description

Density, distribution function, quantile function and random generation for a log normal distribution whose arithmetic mean equals to 'mean' and standard deviation equals to 'sd'.

Usage

```
dlnormb(x, mean = exp(0.5), sd = sqrt(exp(2) - exp(1)), log = FALSE)

plnormb(
  q,
  mean = exp(0.5),
  sd = sqrt(exp(2) - exp(1)),
  lower.tail = TRUE,
  log.p = FALSE
)

qlnormb(
  p,
  mean = exp(0.5),
  sd = sqrt(exp(2) - exp(1)),
  lower.tail = TRUE,
  log.p = FALSE
)

rlnormb(n, mean = exp(0.5), sd = sqrt(exp(2) - exp(1)))
```

Arguments

x, q	vector of quantiles.
mean	the mean of the distribution.
sd	the standard deviation of the distribution.
log, log.p	logical. if 'TRUE' probabilities 'p' are given as 'log(p)'.
lower.tail	logical. if 'TRUE', probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
p	vector of probabilities.
n	number of observations. If 'length(n) > 1', the length is taken to be the number required.

Details

This function calls the corresponding density, distribution function, quantile function and random generation from the log normal (see Lognormal) after evaluation of $meanlog = log(mean^2/\sqrt{sd^2 + mean^2})$ and $sdlog = \sqrt{\log(1 + sd^2/mean^2)}$

Value

'dlnormb' gives the density, 'plnormb' gives the distribution function, 'qlnormb' gives the quantile function, and 'rlnormb' generates random deviates. The length of the result is determined by 'n' for 'rlnorm', and is the maximum of the lengths of the numerical arguments for the other functions. The numerical arguments other than 'n' are recycled to the length of the result. Only the first elements of the logical arguments are used.

The default 'mean' and 'sd' are chosen to provide a distribution close to a lognormal with 'meanlog = 0' and 'sdlog = 1'.

See Also

Lognormal

Examples

```
x <- rlnormb(1E5, 3, 6)
mean(x)
sd(x)
dlnormb(1) == dnorm(0)   # TRUE: default params give meanlog=0, sdlog=1
dlnormb(1) == dlnorm(1)  # TRUE
```

---

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 comma.
name	vector of characters of the same length as the final 'mc' object. If 'NULL', names are taken from the names 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 names of its elements are used. 'devname = TRUE' develops the name using the name of the 'mc' object as prefix. Finally, names are made unique.

Value

An object of class 'mc'.

See Also

mcnode, the basic element of a 'mc' object. To evaluate 'mc' objects: mcmodel, evalmcmod, evalmccut. To study 'mc' objects: print.mc, summary.mc, plot.mc, converg, hist.mc, tornado, tornadounc.

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 in the variability and uncertainty dimensions.

Usage

```
ndvar(n)

ndunc(n)
```

Arguments

n	number of simulations.

Details

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

'ndunc()' gets and 'ndunc(n)' sets the number of simulations in the uncertainty dimension in 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 (unlike options).

See Also

mcstoc, mcdata

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' objects 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 to apply the function. May be '"all"' (default) to apply the function on all values, '"var"' to apply on the variability dimension, '"unc"' to apply on the uncertainty dimension, or '"variates"' to apply on the variates. Note: do not use '"var"' for '"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'.
...	optional arguments to 'fun'.

Value

If 'fun' returns a vector of length 'n' or if 'margin="all"', the returned 'mcnode's have the same type and dimension as 'x'. Otherwise, the type of 'mcnode' is changed accordingly.

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 evaluation of very high dimension models using time instead of memory.

Builds a 'mcmodelcut' object that can be sent to evalmccut.

Usage

```
evalmccut(model, nsv = ndvar(), nsu = ndunc(), seed = NULL, ind = "index")

mcmodelcut(x, is.expr = FALSE)

## S3 method for class 'mccut'
plot(
  x,
  stat = c("median", "mean"),
  lim = c(0.025, 0.25, 0.75, 0.975),
  griddim = NULL,
  xlab = names(x),
  ylab = "Fn(x)",
  main = "",
  draw = TRUE,
  ...
)

## S3 method for class 'mccut'
print(x, lim = c(0.025, 0.975), digits = 3, ...)
```

Arguments

model	a 'mcmodelcut' object or a valid call including three blocks.
nsv	the number of simulations for variability.
nsu	the number of simulations for uncertainty.

seed	the random seed. If 'NULL' the seed is unchanged.
ind	the variable name used in 'model' to refer to the uncertainty loop index.
x	for mcmodelcut: a call or an expression (if is.expr=TRUE) including three blocks (see Details and Examples for the required structure). For print.mccut and plot.mccut: an 'mccut' object.
is.expr	FALSE to send a call, TRUE to send an expression (see mcmodel examples).
stat	a character string: "median" (default) or "mean", the statistic used to display the uncertainty in the plot.
lim	a vector of quantiles for the uncertainty dimension. For print.mccut defaults to c(0.025, 0.975); for plot.mccut defaults to c(0.025, 0.25, 0.75, 0.975).
griddim	a vector of two integers specifying the grid layout (rows, columns) for the plot. If NULL, the grid is computed automatically.
xlab	a vector of x-axis labels, one per node. Defaults to the names of the 'mccut' object.
ylab	the y-axis label.
main	the plot title.
draw	logical. If TRUE (default), the plot is drawn. Set to FALSE when calling from within evalmccut to store the plot data.
...	additional arguments passed to the print or plot function.
digits	number of digits in the print.

Details

'x' (or 'model') should be built as three blocks, separated by '{':

1. The first block is evaluated once (and only once) before the loop. It should evaluate all '"V"', '"0"' and '"U"' nodes.
2. The second block, which should lead to an 'mc' object, is evaluated using 'nsu=1'. It must end with an expression like 'mymc <- mc(...)'.
3. The third block is evaluated on the 'mc' object. All resulting statistics are stored. It should build a list of statistics such as 'summary', 'plot', 'tornado' or any function leading to a vector, matrix or list of those.

Steps 2 and 3 are repeated 'nsu' times. At each iteration, the loop index (from 1 to 'nsu') is assigned to the variable specified in 'ind'.

Value

An object of class 'mccut'.

Note

The seed is set at the beginning of the evaluation. Complete similarity of two evaluations with the same seed is not certain depending on the model structure. In particular, the results will differ from evalmcmod with the same seed. Do not use upper case variables in your model to avoid conflicts with internal variables.

See Also

evalmcmod, mcmodelcut.

Examples

```
modEC3 <- mcmodelcut({
## First block: evaluates all 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 VU nodes, leads to 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: statistics
  {
    list(
      sum = summary(res),
      plot = plot(res, draw=FALSE),
      minmax = lapply(res, range)
    )
  }
})

x <- evalmccut(modEC3, nsv=101, nsu=101, seed=666)
summary(x)
```

---

mcmodel *Specify a Monte Carlo Model*

Description

Specifies a 'mcmodel' without evaluating it, for subsequent 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.

Details

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

Value

an R expression of class 'mcmodel'.

See Also

expression, evalmcmod to evaluate the model, mcmodelcut to evaluate 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*

Description

Creates a 'mcnode' object from a vector, an array or another 'mcnode'.

'mcdatanocontrol' is a fast but unchecked version of mcdata which forces the dimension of data to be '(nsv x nsu x nvariates)' and sets the attributes and class without any validation. Useful when your model is tested and speed is important.

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 a '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 multivariate nodes. May be '"each"' (default), '"none"', or a vector of function names applied on the variates dimension before any output.

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 exist:

'"V" mcnode' for "Variability", arrays of dimension '(nsv x 1 x nvariates)'. The variability of the data should reflect parameter variability.

'"U" mcnode' for "Uncertainty", arrays of dimension '(1 x nsu x nvariates)'. The variability reflects parameter uncertainty.

'"VU" mcnode' for "Variability and Uncertainty", arrays of dimension '(nsv x nsu x nvariates)'.

'"0" mcnode' for "Neither Variability nor Uncertainty", arrays of dimension '(1 x 1 x nvariates)'.

Multivariate nodes ('nvariates != 1') should be used for multivariate distributions (rmultinomial, rmultinormal, rempiricalD, rdirichlet).

Recycling rules are limited: recycling is only permitted to fill a dimension from 1 to the final size of the dimension.

Value

An 'mcnode' object.

See Also

mcstoc to build a stochastic 'mcnode' object. mc to build a Monte-Carlo object. is.mcnode, dimmcnode, typemcnode.

Examples

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

(x0 <- mcdata(100, type = "0"))
(xV <- mcdata(1:ndvar(), type = "V"))
(xU <- mcdata(10 * 1:ndunc(), type = "U"))
(xVU <- mcdata(1:(ndvar() * ndunc()), type = "VU"))

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

ndvar(oldvar)
ndunc(oldunc)
```

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

Description

Builds an 'mcnode' as a mixture of '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 an 'mcnode'.
`mcvalues`	a named list of 'mcnode's, 'mcdata' calls, or 'mcstoc' calls. Each element should be or lead to a compatible 'mcnode'.
`type`	the type of 'mcnode' to be built. By default, a '"V"' node.
`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 final 'mcnode'.
outm	the default output for multivariate nodes. See outm.
seed	the random seed. If 'NULL' the seed is unchanged.

Details

'mcswitch' may be either:

- a vector of weights (need not sum to one, must be nonneg and not all zero). Length must equal the number of elements in 'mcvalues'.
- a '"0 mcnode"' to build any type of node.
- a '"V mcnode"' to build a '"V"' or '"VU"' node.
- a '"U mcnode"' to build a '"U"' or '"VU"' node.
- a '"VU mcnode"' to build a '"VU"' node.

Names of elements in 'mcvalues' should correspond to values in 'mcswitch', specified as character. Elements are evaluated only if needed.

Value

An 'mcnode' object.

See Also

mcdata, mcstoc, switch.

Examples

```
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")
## Equivalent using weights
mcprobtree(c(.75, .20, .05), list("1"=conc1, "2"=conc2, "3"=conc3), type="VU")
```

---

mcratio *Ratio of Uncertainty to Variability*

Description

Provides measures of variability, uncertainty, and both combined for an 'mc' or an 'mcnode' object.

Usage

```
mcratio(x, pcentral = 0.5, pvar = 0.975, punc = 0.975, na.rm = FALSE)
```

Arguments

x	an 'mc' or an 'mcnode' object.
pcentral	the quantile for the central tendency.
pvar	the quantile for the measure of variability.
punc	the quantile for the measure of uncertainty.
na.rm	logical; should NA values be stripped before computation?

Details

Given quantiles A, B, C, D defined as:

A the '(100*pcentral)'th percentile of uncertainty for the '(100*pcentral)'th percentile of variability

B the '(100*pcentral)'th percentile of uncertainty for the '(100*pvar)'th percentile of variability

C the '(100*punc)'th percentile of uncertainty for the '(100*pcentral)'th percentile of variability

D the '(100*punc)'th percentile of uncertainty for the '(100*pvar)'th percentile of variability

the following ratios are estimated: Variability Ratio: B/A; Uncertainty Ratio: C/A; Overall Uncertainty Ratio: D/A.

Value

A matrix.

References

Ozkaynak, H., Frey, H.C., Burke, J., Pinder, R.W. (2009) "Analysis of coupled model uncertainties in source-to-dose modeling of human exposures to ambient air pollution: A PM2.5 case study", Atmospheric Environment, 43(9), 1641-1649.

Examples

```
data(total)
mcratio(total, na.rm=TRUE)
```

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 multivariate nodes. May be '"each"' (default), '"none"', or a vector of function names (as character strings) applied on the variates dimension before output (e.g. '"mean"', 'c("min","max")').
nsample	the name of the parameter of the function giving the size of the vector. By default '"n"', as in most distributions of the **stats** library.
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'. 'lsup' must be pairwise strictly greater than 'linf'.
lhs	should a Latin Hypercube Sampling be used? See lhs.

Details

Any function that accepts vectors/matrices as arguments may be used (notably all current random generators of the **stats** package). Arguments may be sent classically, but it is strongly recommended to use consistent 'mcnode's if arguments should be recycled, since complex recycling is handled for 'mcnode' but not for vectors.

Compatibility rules for 'mcnode' arguments:

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

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

'type="VU"' accepts '"0"', '"V"', '"U"' and '"VU"' nodes with appropriate recycling rules.

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

If 'rtrunc=TRUE', the distribution is truncated on '(linf, lsup]'. The function 'func' should have a 'q' form and a 'p' form.

If 'lhs=TRUE', a Latin Hypercube Sampling is used on 'nsv' and 'nsu'. The function 'func' should have a 'q' form. Not allowed with multivariate distributions.

Value

An 'mcnode' object.

See Also

mcnode for a description of 'mcnode' objects. Ops.mcnode for operations on 'mcnode' objects. rtrunc for important warnings on the 'rtrunc' option.

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")

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

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

ndvar(Oldnvar)
ndunc(Oldnunc)
```

MinimumQuantileInformation

Minimum Quantile Information Distribution

Description

Density, distribution function, quantile function and random generation for Minimum Quantile Information distribution.

Usage

```
dmqi(x,
  mqi,
  mqi.quantile = c(0.05, 0.5, 0.95),
  realization = NULL,
  k = 0.1,
  intrinsic = NA,
  log = FALSE)

pmqi(q,
  mqi,
  mqi.quantile = c(0.05, 0.5, 0.95),
  realization = NULL,
  k = 0.1,
  intrinsic = NA,
  lower.tail = TRUE,
  log.p = FALSE
)

qmqi(p,
  mqi,
  mqi.quantile = c(0.05, 0.5, 0.95),
  realization = NULL,
  k = 0.1,
  intrinsic = NA,
  lower.tail = TRUE,
  log.p = FALSE
)

rmqi(n,
  mqi,
  mqi.quantile = c(0.05, 0.5, 0.95),
  realization = NULL,
  k=0.1,
  intrinsic = NA
)
```

```
pmqi(
  q,
  mqi,
  mqi.quantile = c(0.05, 0.5, 0.95),
  realization = NULL,
  k = 0.1,
  intrinsic = NA,
  lower.tail = TRUE,
  log.p = FALSE
)

qmqi(
  p,
  mqi,
  mqi.quantile = c(0.05, 0.5, 0.95),
  realization = NULL,
  k = 0.1,
  intrinsic = NA,
  lower.tail = TRUE,
  log.p = FALSE
)

rmqi(
  n,
  mqi,
  mqi.quantile = c(0.05, 0.5, 0.95),
  realization = NULL,
  k = 0.1,
  intrinsic = NA
)
```

Arguments

x, q	Vector of quantiles
mqi	Minimum quantile information
mqi.quantile	The quantile of ‘mqi‘. It’s a vector of length 3. Default is ‘c(0.05, 0.5, 0.95)‘, that is the 5th, 50th and 95th.
realization	Default is ‘NULL‘. If not ‘NULL‘, used to define ‘L‘ or ‘U‘ (see details).
k	Overshot, default value is 0.1.
intrinsic	Use to specify a prior bounds of the intrinsic range. Default = ‘NA‘.
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]‘.
p	Vector of probabilities.
n	Number of observations.

Details

p_1, p_2, and p_3 are percentiles of a distribution with $p_1 < p_2 < p_3$. The interval $[L, U]$ is given with:

$$L = x_{p_1}$$

$$U = x_{p_3}$$

The support of minimum quantile information distribution is determined by the intrinsic range:

$$[L^*, U^*] = [L - k \times (U - L), U + k \times (U - L)]$$

where k denotes an overshoot and is chosen by the analyst (usually $k = 10\%$, which is the default value).

Given the three values of quantile, x_{p_1}, x_{p_2} and x_{p_3}, and define $p_0 = 0$, $p_4 = 1$, $x_{p_0} = L^*$ and $x_{p_4} = U^*$ the minimum quantile information distribution is given by:

Probability density function

$$f(x) = \frac{p_i - p_{i-1}}{x_{p_i} - x_{p_{i-1}}} \text{ for } x_{p_{i-1}} \leq x < x_{p_i}, i = 1, \ldots, 4$$

$$f(x) = 0, \text{ otherwise}$$

Cumulative distribution function

$$F(x) = 0 \text{ for } x < x_{p_0}$$

$$F(x) = \frac{p_i - p_{i-1}}{x_{p_i} - x_{p_{i-1}}} * (x - x_{p_{i-1}}) + p_{i-1} \text{ for } x_{p_{i-1}} \leq x < x_{p_i}, i = 1, \ldots, 4$$

$$F(x) = 1 \text{ for } x_{p_4} \leq x$$

This distribution is usually used for expert elicitation. If experts have realization information, then the range $[L, U]$ is given by:

$$L = \min(x_{p_1}, realization)$$

$$U = \max(x_{p_3}, realization)$$

For some questions, experts may have information for the intrinsic range and set a prior intrinsic range (L^* and U^*).

NOTE that the function is vectorized only for x, q, p, n. As a consequence, it can't be used for variable other parameters.

Author(s)

Yu Chen and Arie Havelaar

References

Hanea, A. M., & Nane, G. F. (2021). An in-depth perspective on the classical model. In International Series in Operations Research & Management Science (pp. 225–256). Springer International Publishing.

Examples

```
curve(dmqi(x, mqi=c(40,50,60), intrinsic=c(0,100)), from=0, to=100, type = "l", xlab="x",ylab="pdf")
curve(pmqi(x, mqi=c(40,50,60), intrinsic=c(0,100)), from=0, to=100, type = "l", xlab="x",ylab="cdf")
rmqi(n = 10, mqi=c(555, 575, 586))
```

`multinormal` *The Vectorized Multivariate Normal Distribution*

Description

Vectorized versions of `rmvnorm` and `dmvnorm` from the **mvtnorm** package, providing random deviates and density for the multivariate normal distribution with varying vectors of means and varying covariance matrices.

Usage

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

dmultinormal(x, mean, sigma, log = FALSE)
```

Arguments

`n`	number of observations. If '`length(n) > 1`', the length is taken to be the number required.
`mean`	vector or matrix of means. If a matrix, each row is taken to be a mean vector. Default is a vector of 0 of convenient length.
`sigma`	covariance vector corresponding to the coercion of the covariance matrix into a vector (if unique for all '`n`' or '`x`') or array of covariance vectors (if varying). Default is a diagonal matrix of convenient size.
`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"`').
`x`	vector or matrix of quantiles. If x is a matrix, each row is taken to be a quantile.
`log`	logical; if '`TRUE`', densities d are given as '`log(d)`'.

Details

'`rmvnorm(n, m, s)`' is equivalent to '`rmultinormal(n, m, as.vector(s))`'. '`dmvnorm(x, m, s)`' is equivalent to '`dmultinormal(x, 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 will use the second row, ... recycling being permitted by row.

If '`mean`' is a vector of length '`1`' or is a matrix with '`1`' columns, '`sigma`' should be a vector of length '`1 x 1`' or a matrix with '`1^2`' columns.

Note

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

Examples

```
## mean and sigma as vectors
(mean <- c(10, 0))
(sigma <- matrix(c(1, 2, 2, 10), ncol = 2))
sigma <- as.vector(sigma)
(x <- matrix(c(9, 8, 1, -1), ncol = 2))
round(rmultinormal(10, mean, sigma))
dmultinormal(x, mean, sigma)

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

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.

See Also

is.finite, NA

Examples

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

Ops.mcnode *Operations on mcnode Objects*

Description

Alters the way operations are performed on 'mcnode' objects for better consistency. This method is used for any of the Group Ops functions.

Usage

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

Arguments

e1	an 'mcnode' object, a vector or an array.
e2	an optional 'mcnode' object, a vector or an array.

Details

Rules (illustrated with '+', ignoring the 'nvariates' dimension):

- '0 + 0 = 0'; '0 + V = V'; '0 + U = U'; '0 + VU = VU'
- 'V + V = V': if both of the same '(nsv)' dimension
- 'V + U = VU': U recycled "by row", V recycled "by column"
- 'V + VU = VU': V recycled "by column"
- 'U + U = U': if both of the same '(nsu)' dimension
- 'U + VU = VU': U recycled "by row"
- 'VU + VU = VU': if dimensions are '(nsu x nsv)'

The 'outm' attribute is transferred as: 'each + each = each'; 'none + other = other'; 'other1 + other2 = other1'.

Value

the result as a 'mcnode' object.

See Also

mcdata, mcstoc

Examples

```
oldvar <- ndvar()
oldunc <- ndunc()
ndvar(30)
ndunc(20)
x0 <- mcdata(3, type="0")
xV <- mcdata(1:ndvar(), type="V")
xU <- mcdata(1:ndunc(), type="U")
xVU <- mcdata(1:(ndunc()*ndvar()), type="VU")
-x0
x0 + 3
xV * x0
xV + xU
xU + xVU
ndvar(oldvar)
ndunc(oldunc)
```

---

outm *Changes the Output of Nodes*

Description

Changes the 'outm' attribute of an 'mcnode' or a node of an 'mc' object.

Usage

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

Arguments

x	a 'mcnode' or a 'mc' object.
value	the output of the 'mcnode' for multivariate nodes. May be '"each"' (default) if output should be provided for each variate independently, '"none"' for no output, or a vector of function names (as character strings) applied on the variates dimension before any output (e.g. '"mean"', '"median"', 'c("min","max")'). The function should return one value per vector.
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 (Modified) PERT Distribution*

Description

Density, distribution function, quantile function and random generation for the PERT (*aka* Beta PERT) distribution with minimum equals to 'min', mode equals to 'mode' (or, alternatively, mean equals to 'mean') and maximum equals to 'max'.

Usage

```
dpert(x, min = -1, mode = 0, max = 1, shape = 4, log = FALSE, mean = 0)

ppert(
  q,
  min = -1,
  mode = 0,
  max = 1,
  shape = 4,
  lower.tail = TRUE,
  log.p = FALSE,
  mean = 0
)

qpert(
  p,
  min = -1,
  mode = 0,
  max = 1,
  shape = 4,
  lower.tail = TRUE,
  log.p = FALSE,
  mean = 0
)

rpert(n, min = -1, mode = 0, max = 1, shape = 4, mean = 0)
```

Arguments

x, q	Vector of quantiles.
min	Vector of minima.
mode	Vector of modes.
max	Vector of maxima.
shape	Vector of scaling parameters. Default value: 4.
log, log.p	Logical; if 'TRUE', probabilities 'p' are given as 'log(p)'.
mean	Vector of means, can be specified in place of 'mode' as an alternative parametrization.
lower.tail	Logical; if 'TRUE' (default), probabilities are 'P[X <= x]', otherwise, 'P[X > x]'
p	Vector of probabilities
n	Number of observations. If length(n) > 1, the length is taken to be the number required.

Details

The PERT distribution is a Beta distribution extended to the domain '[min, max]' with mean

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

The underlying beta distribution is specified by α_1 and α_2 defined as

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

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

'mode' or 'mean' can be specified, but not both. Note: 'mean' is the last parameter for backcompatibility. A warning will be provided if some combinations of 'min', 'mean' and 'max' leads to impossible mode.

David Vose (See reference) proposed a modified PERT distribution with a shape parameter different from 4.

The PERT distribution is frequently used (with the triangular distribution) to translate expert estimates of the min, max and mode of a random variable in a smooth parametric distribution.

Value

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

Author(s)

Regis Pouillot and Matthew Wiener

References

Vose D. Risk Analysis - A Quantitative Guide (2nd and 3rd editions, John Wiley and Sons, 2000, 2008).

See Also

Beta

Examples

```
curve(dpert(x,min=3,mode=5,max=10,shape=6), from = 2, to = 11, lty=3,ylab="density")
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 = .30, c("Default: 4","shape: 2","shape: 6"), lty=1:3)
## Alternatie parametrization using mean
curve(dpert(x,min=3,mean=5,max=10), from = 2, to = 11, lty=2 ,ylab="density")
curve(dpert(x,min=3,mode=5,max=10), from = 2, to = 11, add=TRUE)
legend(x = 8, y = .30, c("mode: 5","mean: 5"), lty=1:2)
```

---

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 CDF of the estimate of a 'mcnode' or 'mc' object ('"U"' and '"VU"' nodes).

Usage

```
## S3 method for class 'mc'
plot(
  x,
  prec = 0.001,
  stat = c("median", "mean"),
  lim = c(0.025, 0.25, 0.75, 0.975),
  na.rm = TRUE,
  griddim = NULL,
  xlab = NULL,
  ylab = "Fn(x)",
  main = "",
  draw = TRUE,
  paint = TRUE,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

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

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

Arguments

x	a 'mcnode' or a 'mc' object.
prec	the precision of the plot. '0.001' gives an ecdf from the 0.000, 0.001, ..., 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 envelope. May be 'NULL' or empty.
na.rm	should NA values be discarded?
griddim	a vector of two integers for the grid size. If 'NULL', calculated automatically.
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?
paint	should the envelopes be filled?
xlim	x coordinate range: a vector of length 2 or a list of such vectors.
ylim	y coordinate range: a vector of length 2 or a list of such vectors.
...	further arguments to be passed to 'plot.stepfun'.

Details

For '"VU"' and '"U"' nodes, quantiles are calculated using quantile.mc within each of the 'nsu' simulations (by columns). The medians (or means if 'stat="mean"') are plotted, with 'lim' quantiles as the envelope.

Value

a 'plot.mc' object (list of quantiles used to draw the plot), invisibly.

References

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

See Also

ecdf, plot, quantile.mc, ggplotmc for a ggplot2 version.

Examples

```
data(total)
plot(xVUM3)
## Only one envelope for 0.025 and 0.975
plot(xVUM3, lim=c(0.025, 0.975))
```

`plot.tornado` *Draw a Tornado Chart*

Description

Draws a Tornado chart as provided by `tornado` or `tornadounc`.

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 = "median",
  name = NULL,
  xlab = "method",
  ylab = "",
  ...
)
```

Arguments

`x`	a `tornado` or `tornadounc` object.
`which`	which output to plot (for multivariate output).
`name`	vector of names for input variables. If '`NULL`', taken from the object.
`stat`	the statistic of the output to be considered. For a '`tornado`' object: '`"median"`' or '`"mean"`'. For a '`tornadounc`' object: should match a row name, or use the row number.
`xlab`	label of the x axis. If '`"method"`', use the correlation method.

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 it.

Value

'NULL' (called for side effects).

See Also

tornado

Examples

```
data(ec)
x <- evalmcmod(ec$modEC2, nsv=100, nsu=100, seed=666)
tor <- tornado(x, 7)
plot(tor)
```

---

pmin.mcnode *Parallel Maxima and Minima for mcnodes*

Description

Returns the parallel maxima and minima of the input values, extended to 'mcnode' objects. The same rules as in Ops.mcnode are applied for type determination.

Usage

```
pmin(..., na.rm = FALSE)

pmax(..., na.rm = FALSE)

## Default S3 method:
pmin(..., na.rm = FALSE)

## Default S3 method:
pmax(..., na.rm = FALSE)

## S3 method for class 'mcnode'
pmin(..., na.rm = FALSE)

## S3 method for class 'mcnode'
pmax(..., na.rm = FALSE)
```

Arguments

...	one or more 'mcnode's or 'mcnode's and compatible vectors. The first element must be an 'mcnode'.
na.rm	a logical indicating whether missing values should be removed.

Details

'pmax' and 'pmin' take one or more 'mcnode' and/or vectors as arguments and return a 'mcnode' of adequate type and size giving the "parallel" maxima or minima. Note that the first element of '...' should be an 'mcnode'.

Value

an 'mcnode' of adequate type and dimension.

See Also

min, Ops.mcnode

Examples

```
ndvar(10)
ndunc(21)
x <- mcstoc(rnorm, "V")
pmin(x, 0)
y <- mcdata(rep(c(-1, 1), length=ndunc()), "U")
unclass(pmin(x, y))
```

---

print.mc *Print a mcnode or mc Object*

Description

Prints a description of the structure of a 'mc' or '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.

See Also

mcnode for 'mcnode' objects, mc for 'mc' objects.

Examples

```
data(total)
print(xVU)
print(total)
```

---

quantile.mc *Quantiles of a mc or mcnode Object*

Description

Evaluates quantiles of a 'mc' or 'mcnode' object. Used internally 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' or 'mcnode' object.
probs	the quantiles to be calculated.
lim	a vector of numbers (between 0 and 1) indicating the envelope. May be 'NULL' or empty.
na.rm	logical; should NA values be stripped?
...	for generic method consistency.

Details

Quantiles are evaluated in the variability dimension. Then, the median, the mean and the 'lim' quantiles are evaluated for each of these quantiles in the uncertainty dimension.

Value

a list of quantiles.

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' (excluded) and 'lsup' (included).

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, with 'lsup > linf' (strictly).
...	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', 'log.p' or 'lower.tail'.

For discrete distributions, 'rtrunc' samples within '(linf, lsup]'.

Warning: The method is flexible, but can lead to problems linked to rounding errors in some extreme situations. The function checks that the values are in the expected range and returns an error if not.

Value

A vector of 'n' values.

Examples

```
rtrunc("rnorm", n = 10, linf = 0)
range(rtrunc(rnorm, n = 1000, linf = 3, lsup = 5, sd = 10))
## Discrete distributions
range(rtrunc(rpois, 1000, linf = 2, lsup = 4, lambda = 1))
```

---

spaghetti *Spaghetti Plot of mc/mcnode Object*

Description

Use plot to draw spaghetti plots for the mc/mcnode objects.

Usage

```
spaghetti(x, ...)

## S3 method for class 'mc'
spaghetti(
  x,
  griddim = NULL,
  xlab = names(x),
  ylab = "F(n)",
  main = "",
  maxlines = 100,
  ...
)

## S3 method for class 'mcnode'
spaghetti(x, ...)
```

Arguments

x	mc/mcnode object
...	further arguments to be passed to plot.stepfun()
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.
maxlines	the maximum number of ecdf to draw.

Examples

```
data(total)
spaghetti(mc(xVUM))
spaghetti(xVUM)
```

---

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 'summary.mc'
print(x, digits = 3, ...)

## S3 method for class 'mccut'
summary(object, lim = c(0.025, 0.975), ...)
```

Arguments

object	a 'mcnode', 'mc' or 'mccut' object.
probs	a vector of values used for the quantile function (variability dimension).
lim	a vector of values used for the quantile function (uncertainty dimension).
...	for generic function consistency.
digits	number of digits in the print.
x	a 'summary.mc' object as returned by the 'summary.mc' function.

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.

For multivariate nodes: if the '"outm"' attribute is '"none"', the node is not evaluated; if it is '"each"', the variates are evaluated one by one; if it is a function name (e.g. '"mean"'), that function is applied on the variates dimension before output.

Value

a list.

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 the Variability Dimension (Tornado)*

Description

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

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 or mccut object.
output	the rank or name of the output to be considered. By default: the last element.
use	an optional character string for computing covariances in the presence of missing values: '"all.obs"', '"complete.obs"' or '"pairwise.complete.obs"'.
method	a character string for the correlation coefficient: '"spearman"' (default), '"kendall"' or '"pearson"'.
lim	a vector of quantiles used to compute the credible interval.
x	a 'tornado' object.
...	further arguments to be passed to the final print function.

Details

The tornado function computes Spearman's rho statistic to estimate a rank-based measure of association between one set of random variables 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.

Type rules for the output node:

- '"V" mcnode': correlations are only provided for other '"V" mcnode's.
- '"VU" mcnode': correlations are provided for other '"VU"' and '"V"' nodes.

Value

an invisible object of class 'tornado', containing: 'value' (correlation coefficients), 'output' (name of output), 'method', 'use'.

See Also

cor, plot.tornado to draw the results.

Examples

```
data(total)
tornado(total, 2, "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 the Uncertainty Dimension (Tornado)*

Description

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

Usage

```
tornadounc(mc, ...)

## Default S3 method:
tornadounc(mc, ...)

## 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.
...	further arguments to be passed to the final print function.
output	the rank or name of the output. Should be a '"VU"' or '"U"' node. By default: the last element.
quant	the vector of quantiles used in the variability dimension.

use	an optional character string for computing covariances in presence of missing values: '"all.obs"', '"complete.obs"' or '"pairwise.complete.obs"'.
method	a character string for the correlation coefficient: '"spearman"' (default), '"kendall"' or '"pearson"'.
x	a 'tornadounc' object.

Details

The function computes Spearman's rho statistic between values or statistics calculated in the variability dimension of inputs and outputs. The statistics are the mean, the standard deviation and the quantiles specified by 'quant'.

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

Value

an invisible object of class 'tornadounc'.

See Also

cor, tornado for variability dimension, plot.tornadounc to draw the results.

Examples

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

---

total *An Example of all Kind of mcnode*

Description

An example for each kind of 'mcnode's. 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

```
data(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' (alternatively, mean equal 'mean') and maximum equal to 'max'.

Usage

```
dtriang(x, min = -1, mode = 0, max = 1, log = FALSE, mean = 0)

ptriang(
  q,
  min = -1,
  mode = 0,
  max = 1,
  lower.tail = TRUE,
  log.p = FALSE,
  mean = 0
)

qtriang(
```

```
  p,
  min = -1,
  mode = 0,
  max = 1,
  lower.tail = TRUE,
  log.p = FALSE,
  mean = 0
)

rtriang(n, min = -1, mode = 0, max = 1, mean = 0)
```

Arguments

x, q	vector of quantiles.
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)'.
mean	Vector of means, can be specified in place of 'mode' as an alternative parametrization.
lower.tail	logical; if 'TRUE' (default), probabilities are 'P[X <= x]', otherwise, 'P[X > x]'.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.

Details

If 'min == mode == max', there is no density in that case and 'dtriang' will return 'NaN' (the error condition) (Similarity with Uniform).

'mode' or 'mean' can be specified, but not both. Note: 'mean' is the last parameter for back-compatibility. A warning will be provided if some combinations of 'min', 'mean' and 'max' leads to impossible mode.

Value

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

Examples

```
curve(dtriang(x, min=3, mode=6, max=10), from = 2, to = 11, ylab="density")
## Alternative parametrization
curve(dtriang(x, min=3, mean=6, max=10), from = 2, to = 11, add=TRUE, lty=2)
##no density when  min == mode == max
dtriang(c(1,2,3),min=2,mode=2,max=2)
```

unmc *Unclass the mc or mcnode Object*

Description

Unclasses an 'mc' object into a list of arrays, or an 'mcnode' object into an array.

Usage

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

Arguments

x	a 'mc' or a 'mcnode' object.
drop	should 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, matrices and arrays. If 'x' is an 'mcnode' object: an array. If 'drop=TRUE', a vector, matrix or array.

Examples

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

Index