Package 'propagate'

November 10, 2026

Type Package

LazyData no

Title Propagation of Uncertainty

Version 1.3-0

Date 2026-11-09

Maintainer Andrej-Nikolai Spiess <draspiess@gmail.com>

Description Propagation of uncertainty using higher-order Taylor expansion and Monte Carlo simulation. Calculations of propagated uncertainties are based on matrix calculus including covariance structure according to Arras 1998 <doi:10.3929/ethz-a-010113668> (first order), Wang & Iyer 2005 <doi:10.1088/0026-1394/42/5/011> (second order) and the Monte Carlo method of GUM Supplement 1 (2008) <doi:10.59161/JCGM101-2008>.

License GPL (>= 2)

Depends R (>= 4.0.0)

Imports Rcpp, minpack.lm, hdf5r, crayon, nortest

LinkingTo Rcpp

NeedsCompilation yes

Suggests readxl, knitr, rmarkdown

VignetteBuilder knitr

Author Andrej-Nikolai Spiess [aut, cre]

Repository CRAN

Date/Publication 2026-11-10 07:30:02 UTC

Contents

bigcor 2
conformity 4
cor2cov 8
corDist 9
datasets 12
doGauge 16

fitDistr 18
interval 23
makeDat 26
makeDerivs 29
matrixStats 31
mixCov 32
moments 34
numDerivs 35
plot.propagate 36
predictNLS 37
propagate 41
propagateMV 54
rDistr 57
report 60
sobol 65
statVec 69
stochContr 70
summary.propagate 71
WelchSatter 77

Index **82**

`bigcor` *Creating very large correlation/covariance matrices*

Description

The storage of a value in double format needs 8 bytes. When creating large correlation matrices, the amount of RAM might not suffice, giving rise to the dreaded *"cannot allocate vector of size ..."* error. For example, an input matrix with 50000 columns/100 rows will result in a correlation matrix with a size of 50000 x 50000 x 8 Byte / (1024 x 1024 x 1024) = 18.63 GByte, which still challenges most standard PCs. `bigcor` uses the framework of the 'hdf5r' package to store the correlation/covariance matrix in a file. The complete matrix is created by filling a large preallocated empty matrix with sub-matrices at the corresponding positions. See 'Details'.

Usage

```
bigcor(x, y = NULL, fun = c("cor", "cov"), size = 2000,
       verbose = TRUE, file = tempfile(fileext = ".h5"), ...)
```

Arguments

`x`	the input matrix.
`y`	`NULL` (default) or a vector, matrix or data frame with compatible dimensions to `x`.
`fun`	create either a correlation (`cor`) or covariance (`cov`) matrix.
`size`	the n x n block size of the submatrices. 2000 has proven time-efficient.

verbose	logical. If TRUE, information is printed in the console when running.
file	name (and possibly the path) of the created h5-file on disk, *e.g.* 'c:/temp/test.h5'.
...	other parameters to be passed to cor or cov.

Details

Calculates a correlation matrix $\mathbf{R}$ or covariance matrix $\mathbf{\Sigma}$ using the following steps:
1) An input matrix $\mathbf{X}$ with N columns is split into k equal size blocks (+ a possible remainder block) $A_1, A_2, \ldots, A_k$ of size n. The block size can be defined by the user, `size = 2000` is a good value because cor can handle this quite quickly. For example, if the matrix has 13796 columns, the split will be $A_1 = 1 \ldots 2000$; $A_2 = 2001 \ldots 4000$; $A_3 = 4001 \ldots 6000$; $A_4 = 6001 \ldots 8000$; $A_5 = 8001 \ldots 10000$; $A_6 = 10001 \ldots 12000$; $A_7 = 12001 \ldots 13796$.
2) For all $k(k+1)/2$ pairwise combinations of blocks (each block with itself and with every other block), the $n \times n$ correlation sub-matrix is calculated. If `y = NULL`, $\mathrm{cor}(A_1, A_1), \mathrm{cor}(A_1, A_2), \ldots, \mathrm{cor}(A_k, A_k)$, otherwise $\mathrm{cor}(A_1, y), \mathrm{cor}(A_2, y), \ldots, \mathrm{cor}(A_k, y)$.
3) The sub-matrices are transferred into a preallocated $N \times N$ empty matrix at the corresponding position (where the correlations would usually reside). To ensure symmetry around the diagonal, this is done twice in the upper and lower triangle. If y was supplied, a $N \times M$ matrix is filled, with M = number of columns in y.

Since the resulting matrix is in 'hdf5r' format, one has to subset to extract regions into normal matrix-like objects. See 'Examples'.

Value

Invisibly, the path of the HDF5 file containing the correlation/covariance matrix (dataset `"matrix"`).

Author(s)

Andrej-Nikolai Spiess

References

https://rmazing.wordpress.com/2013/02/22/bigcor-large-correlation-matrices-in-r/

Examples

```
## Small example to prove similarity
## to standard 'cor'. We create a matrix
## by subsetting the complete 'hdf5r' matrix.
MAT <- matrix(rnorm(70000), ncol = 700)
h5file <- tempfile(fileext = ".h5")
FILE <- bigcor(MAT, size = 500, fun = "cor", file = h5file)
h5 <- hdf5r::H5File$new(FILE, mode = "r")
COR1 <- h5[["matrix"]][,]  # Read entire matrix
## COR1 <- h5[["matrix"]][1:100, 1:100]  # would be submatrix
h5$close_all()
all.equal(COR1, cor(MAT)) # => TRUE
file.remove(h5file)
```

`conformity` *Conformity assessment with measurement uncertainty (JCGM 106:2012)*

Description

Conformity assessment of a measurand against tolerance limits, taking the measurement uncertainty into account, according to JCGM 106:2012. For a single measured item, `conformity` calculates the conformance probability and acceptance limits; for a production process inspected by a measuring system, it calculates the global consumer's and producer's risks, acceptance limits for a target consumer's risk and the capability indices of the process. Optionally, decision rules with the expanded uncertainty (ISO 14253-1, ASME B89.7.3.1, VDA 5) are applied, and `plot` draws the probability density function with the conforming and non-conforming parts and the decision zones (see 'Details').

Usage

```
conformity(object, lower = -Inf, upper = Inf, method = c("MC", "GUF"), prob = NULL,
           um = NULL, accept = NULL, risk = NULL, support = c(-Inf, Inf),
           rule = NULL, p = 0.95, interval = c("symmetric", "shortest"))

## S3 method for class 'conformity'
print(x, digits = 4, ...)

## S3 method for class 'conformity'
plot(x, density = TRUE,
     col = c(accept = "#2E9D48", reject = "#C8102E", zone = "grey70"), ...)
```

Arguments

`object`	for a single measured item (`um = NULL`): a `propagate` object, a vector `c(y, u)` (normal PDF) or `c(y, u, nu)` (scaled and shifted *t*-distribution with `nu` degrees of freedom), or a sample of the measurand. For a production process (`um` given): the density function of the process (*e.g.* `function(x) dnorm(x, 1500, 0.12)`), a sample of it, or a `propagate` object whose Monte Carlo output describes the process (see 'Details').
`lower, upper`	the tolerance limits T_L and T_U; one of them may be infinite (one-sided tolerance interval).
`method`	for a `propagate` object: `"MC"` uses the Monte Carlo sample, `"GUF"` the estimate, standard uncertainty and effective degrees of freedom of the GUM uncertainty framework (normal or scaled and shifted *t*-distribution).
`prob`	single item: the required conformance probability; acceptance limits are calculated such that a measured value at the limit has this conformance probability (guarded acceptance, JCGM 106:2012, 8.3).
`um`	production process: the standard uncertainty of the inspection measurement (normal PDF $h(\eta_m\|\eta)$).

accept	production process: the acceptance limits c(AL, AU); default: the tolerance limits (simple acceptance), or the limits found for risk.
risk	production process: a target global consumer's risk; the acceptance limits are found as the tolerance limits shifted inwards by a guard band w (outwards if $w < 0$) such that R_C equals risk.
support	the support of the process density function (*e.g.* c(0, Inf) for a gamma density), used for the numerical integration.
rule	NULL (default) or one or several decision rules with the expanded uncertainty, or "all": "stringent" (stringent acceptance and stringent rejection, ISO 14253-1), "simple", "stringent.relaxed", "relaxed.stringent", "stringent.simple", "simple.stringent" (acceptance criterion first, rejection criterion second). For a production process, rule defines the acceptance limits and cannot be combined with accept or risk. See 'Details'.
p	the coverage probability of the expanded uncertainty used by rule.
interval	for a single item with method = "MC": the probabilistically symmetric or the shortest Monte Carlo coverage interval (JCGM 101:2008, 7.7), whose (asymmetric) half-widths are the expanded uncertainties U_- and U_+.
x	an object of class conformity.
digits	the number of significant digits for printing.
density	logical. If TRUE, the probability density function of the measurand (single item) or of the process (production process) is drawn above the zone bars, with the parts inside and outside the tolerance interval shaded. Without rule, only the density is drawn.
col	the colours of the acceptance, rejection and uncertainty zones.
...	not used.

Details

Overview. For a single measured item, the conformance probability is calculated from the Monte Carlo output or the GUM uncertainty framework of a propagate object (or from an estimate and its standard uncertainty), together with acceptance limits with guard bands for a required conformance probability. For a production process, the global risks, the acceptance limits for a target consumer's risk and the capability indices are calculated. With rule, a single item gets the acceptance, rejection and uncertainty zones and the decision of each rule; a production process gets the accepted, rejected and undecided fractions and the consumer's and producer's risks of each rule. plot draws the probability density function with the conforming and non-conforming parts and the zones of the decision rules.

Single measured item (JCGM 106:2012, 7 and 8). The conformance probability is $p_c = \Pr(T_L \leq Y \leq T_U)$. With method = "MC" it is the fraction of the Monte Carlo output within the tolerance interval, which accounts for an asymmetric or non-normal distribution of the measurand. With method = "GUF" it is $p_c = F((T_U - y)/u) - F((T_L - y)/u)$, with F the standard normal distribution function or, for $\nu < 1000$, the *t*-distribution function (JCGM 106:2012, 7.3, 7.4; JCGM 101:2008, 6.4.9). With prob, the acceptance limits are $A_U = T_U - q_p$ and $A_L = T_L - q_{1-p}$, where q are the quantiles of $Y - y$ (one-sided at each limit), assuming that the distribution shifts with the measured value.

Production process (JCGM 106:2012, 9.5). The process distribution $g_0(\eta)$ plays the role of the prior distribution of the property of the items, and a measured value has the normal distribution $h(\eta_m|\eta)$ with standard deviation `um`. With the conformance region C and the acceptance interval A,

$$R_C = \int_{\eta \notin C} \Pr(\eta_m \in A|\eta)\, g_0(\eta)\, d\eta, \qquad R_P = \int_{\eta \in C} \Pr(\eta_m \notin A|\eta)\, g_0(\eta)\, d\eta$$

are the global consumer's and producer's risks, computed by numerical integration for a density function and as Monte Carlo averages for a sample. The output also contains the conforming fraction of the process, the accepted fraction, the non-conforming fraction among the accepted items, and the process mean, standard deviation, capability indices $C_p = (T_U - T_L)/(6s)$ and C_{pk} (normal-theory definitions) and the number of defects per million (DPMO).

Decision rules (`rule`; ISO 14253-1, ASME B89.7.3.1, VDA 5). With the expanded uncertainty given by the coverage interval $[y - U_-, y + U_+]$ for coverage probability `p` ($U_- = U_+ = k\,u$ for the GUF, with k from the normal or *t*-distribution; asymmetric for the Monte Carlo method; $U = k\,u_m$ with $k = \Phi^{-1}((1+p)/2)$ for a production process), the acceptance and rejection criteria are

criterion	acceptance: measured value in	rejection: measured value below / above
stringent	$[T_L + U_-, T_U - U_+]$	$T_L - U_+$ / $T_U + U_-$
simple	$[T_L, T_U]$	T_L / T_U
relaxed	$[T_L - U_+, T_U + U_-]$	$T_L + U_-$ / $T_U - U_+$

Stringent acceptance accepts only if the whole coverage interval lies within the tolerance interval (protection of the consumer; guarded acceptance with guard band $w = U$ in the terms of JCGM 106:2012, 8.3); stringent rejection rejects only if the whole coverage interval lies outside (protection of the producer; guarded rejection). A rule combines one acceptance and one rejection criterion; measured values that are neither accepted nor rejected lie in the uncertainty zone (`"no decision"`). `"stringent"` (stringent acceptance and stringent rejection) is the default rule of ISO 14253-1.

For a single item, the decision of each rule is returned. For a production process, each rule gives, by the integrals above with the acceptance interval A of the rule and the rejection limits R_L, R_U: the accepted fraction, the rejected fraction $\Pr(\eta_m < R_L \text{ or } \eta_m > R_U)$, the undecided fraction (items in the uncertainty zone, *e.g.* to be measured again), the consumer's risk R_C (accepted and not conforming) and the producer's risk R_P (rejected and conforming). This shows directly what a rule costs the producer and gains the consumer. The rule `"simple"` gives the risks of simple acceptance (`accept` = tolerance limits), and the acceptance criterion `"stringent"` those of guarded acceptance with $w = U$.

Process or measurand? For a single item, the `propagate` object describes the knowledge of one measurand (for Type A inputs, the uncertainty of a mean). For a production process, the sample must describe the scatter of the individual items, *e.g.* a `propagate` object of a transfer function whose inputs are the item-to-item distributions of the components (assigned with `makeDat` or as `type = "sim"`), not the uncertainties of their means.

Value

An object of class `conformity`, a list. For a single item: `type = "specific"`, `method`, `estimate`, `u`, `nu`, the tolerance limits, the conformance probability `pc`, the non-conformance probability `pnc`, `prob`, the acceptance limits `accept` and the standardized distances `z` of the estimate from the tolerance limits, `p`, the expanded uncertainties `U` (a vector `c(U-, U+)`) and the coverage `interval`.

For a production process: `type = "global"`, `method`, the tolerance limits, `um`, the acceptance limits `accept`, the guard band `w` and $r = w/(2u_m)$ (if `risk` was given), `pc`, `pnc`, the risks `RC` and `RP`, the accepted fraction `accepted`, the non-conforming fraction among accepted items `RC.accepted` and `process` (mean, sd, Cp, Cpk, DPMO), `p` and the expanded uncertainty `U`. In both cases `rules` (`NULL` without `rule`) is a data frame with one row per rule: `rule`, `description`, the acceptance limits `AL`, `AU`, the rejection limits `RL`, `RU` and, for a single item, the `decision`, or, for a production process, the fractions `accepted`, `rejected`, `undecided` and the risks `RC`, `RP`; `density` contains the density of the measurand or of the process for `plot`.

Author(s)

Andrej-Nikolai Spiess

References

Conformity assessment and measurement uncertainty:
Evaluation of measurement data - The role of measurement uncertainty in conformity assessment.
JCGM 106:2012. Joint Committee for Guides in Metrology (2012).
doi:10.59161/JCGM1062012

Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

Decision rules:
Geometrical product specifications (GPS) - Inspection by measurement of workpieces and measuring equipment - Part 1: Decision rules for verifying conformity or nonconformity with specifications.
ISO 14253-1:2017. International Organization for Standardization, Geneva (2017).

Guidelines for Decision Rules: Considering Measurement Uncertainty in Determining Conformance to Specifications.
ASME B89.7.3.1-2001. American Society of Mechanical Engineers, New York (2001).

Capability of Measurement Processes.
VDA Volume 5, 2nd edition. Verband der Automobilindustrie (VDA), Berlin (2010).

See Also

`propagate`, `makeDat`; all examples of JCGM 106:2012 with comparisons against the published values: `demo("JCGM106", package = "propagate")`.

Examples

```
## JCGM 106, 7.3: Zener diode, V_b <= -5.40 V, y = -5.47 V, u = 0.05 V: pc = 0.92
conformity(c(-5.47, 0.05), upper = -5.40)

## JCGM 106, 7.4: motor oil, 12.5 <= Y <= 16.3 mm^2/s: pc = 0.66
conformity(c(13.6, 1.8), lower = 12.5, upper = 16.3)

## JCGM 106, 8.3.3: nandrolone, t-distribution with 9 degrees of freedom, decision limit 2.37 ug/L
conformity(c(2.00, 0.20, 9), lower = 2.00, prob = 0.95)
```

```
## JCGM 106, 9.5.3: precision resistors, R_C = 0.01, R_P = 0.07
conformity(function(x) dnorm(x, 1500, 0.12), lower = 1499.8, upper = 1500.2,
           um = 0.04, accept = c(1499.82, 1500.18))

## JCGM 106, 9.5.4: ball bearings, acceptance limit for R_C = 0.1 %: A = 1.7 um
conformity(function(x) dgamma(x, shape = 4, rate = 4), upper = 2, um = 0.25,
           risk = 0.001, support = c(0, Inf))

## Monte Carlo output of 'propagate' (asymmetric, JCGM 101:2008, 9.4):
## the conformance probability differs from the GUF value
DAT <- cbind(X1 = c(0.010, 0.005, Inf), X2 = c(0, 0.005, Inf))
RES <- propagate(expression(1E6 * (X1^2 + X2^2)), DAT, type = "stat", nsim = 1E5, check = FALSE)
conformity(RES, upper = 300)
conformity(RES, upper = 300, method = "GUF")

## Decision rules (ISO 14253-1) for the Zener diode: zones and decisions of all six rules
C1 <- conformity(c(-5.47, 0.05), upper = -5.40, rule = "all")
C1
plot(C1)

## ... and for the production of precision resistors: risks of each rule
R1 <- conformity(function(x) dnorm(x, 1500, 0.12), lower = 1499.8, upper = 1500.2,
                 um = 0.04, rule = "all")
R1$rules[, c("rule", "accepted", "undecided", "RC", "RP")]
plot(R1)
```

`cor2cov` *Converting a correlation matrix into a covariance matrix*

Description

Converts a correlation matrix into a covariance matrix using variance information. It is therefore the inverse of `cov2cor`.

Usage

```
cor2cov(C, var = NULL)
```

Arguments

`C`	a symmetric numeric correlation matrix $\mathbf{C}$.
`var`	a vector of variances s_i^2, one per row/column of `C`. Required: the default `NULL` gives an error.

Details

Calculates the covariance matrix $\mathbf{\Sigma}$ from a correlation matrix $\mathbf{C}$ and the outer product of the vector of standard deviations $\mathbf{s} = (s_1, \ldots, s_N)^T$, with s_i^2 the elements of `var`, by the elementwise (Hadamard) product $\circ$:

$$\mathbf{\Sigma} = \mathbf{s}\mathbf{s}^T \circ \mathbf{C}, \qquad \Sigma_{ij} = s_i \, s_j \, C_{ij}$$

This is equivalent to $\mathbf{\Sigma} = \mathbf{DCD}$ with $\mathbf{D} = \mathrm{diag}(\mathbf{s})$, the inverse of the relation $r(x_i, x_j) = u(x_i, x_j)/(u(x_i)\, u(x_j))$ of JCGM 100:2008, 5.2.2.

Value

The corresponding covariance matrix.

Author(s)

Andrej-Nikolai Spiess

References

Evaluation of measurement data - Guide to the expression of uncertainty in measurement. JCGM 100:2008. Joint Committee for Guides in Metrology (2008). doi:10.59161/JCGM1002008E

Examples

```
## JCGM 100:2008, Annex H.2
## (see 'References'), simultaneous resistance
## and reactance measurement.
data(H.2)

## Original covariance matrix.
COV <- cov(H.2)
## extract variances
VAR <- diag(COV)

## cor2cov covariance matrix.
COV2 <- cor2cov(cor(H.2), VAR)

## Equal to original covariance matrix.
all.equal(COV2, COV)
```

`corDist` *Correlated Monte Carlo samples of input quantities with arbitrary distributions*

Description

Creates Monte Carlo samples of correlated input quantities, each with its own distribution, joined by a Gaussian copula: Gaussian, rectangular, triangular, arcsine, curvilinear trapezoidal or scaled and shifted *t* given by estimate and standard uncertainty, or any distribution given by a sampler (*e.g.* the generators of `rDistr` or of R's `distributions`) or by a sample of values. The result is used as input of `propagate` with `type = "sim"` and `cov = "full"`.

Usage

```
corDist(mean, u, dist = "normal", cor = NULL, df = Inf, d = NA, nsim = 1E6,
        cor.type = c("pearson", "copula"))
```

Arguments

`mean`	a named vector of the estimates x_i of the input quantities; the names are the variables of the expression passed to `propagate`. `NA` for inputs given by a sampler or a sample.
`u`	the standard uncertainties u_i, recycled to the length of `mean` (`NA` for samplers and samples).
`dist`	the distributions, a character vector or list (recycled): `"normal"`, `"rect"`, `"tri"` (symmetric triangular), `"arcsine"` (U-shaped), `"ctrap"` (curvilinear trapezoidal, JCGM 101:2008, 6.4.3, with `d`), `"t"` (scaled and shifted *t*-distribution with scale u_i and `df` degrees of freedom, JCGM 101:2008, 6.4.9), a sampler `function(n)` returning `n` random values, *e.g.* `function(n) propagate:::rtrap(n, 0, 1, 2, 3)` (the generators of `rDistr` are internal functions, hence `:::`) or `function(n) rgamma(n, 2, 1)`, or a numeric vector of sampled or measured values (resampled with replacement if its length differs from `nsim`).
`cor`	the correlation matrix $\mathbf{R}$ of the input quantities, in the order of `mean`; `NULL` for independent inputs.
`df`	the degrees of freedom of `"t"` margins (recycled). They are also passed to the GUM uncertainty framework of `propagate`.
`d`	the inexactness of the limits of `"ctrap"` margins (recycled).
`nsim`	the number of Monte Carlo samples.
`cor.type`	`"pearson"` (default): the copula correlation is chosen such that the Pearson correlation of the samples equals `cor`; `"copula"`: `cor` is used as the parameter of the Gaussian copula. See 'Details'.

Details

Margins. Each input quantity has expectation x_i and standard deviation u_i, with the half-widths $a = \sqrt{3}\,u_i$ (rectangular), $\sqrt{6}\,u_i$ (triangular), $\sqrt{2}\,u_i$ (arcsine) and $u_i^2 = a^2/3 + d^2/9$ (curvilinear trapezoidal). The *t*-margin has scale u_i and variance $\frac{\nu_i}{\nu_i - 2}u_i^2$, as `mc.margins = "S1"` in `propagate`. For a sampler or a sample, the distribution is that of its values, and x_i and u_i are their mean and standard deviation (`df = Inf`).

Sampling. $\mathbf{Z} \sim N(\mathbf{0}, \mathbf{R}_Z)$ is drawn with the copula correlation matrix $\mathbf{R}_Z$, and $X_i = Q_i(\Phi(Z_i))$ with the quantile function Q_i of the margin (Gaussian copula, Sklar's theorem; Possolo 2010).

The rank correlations of the samples are those of the copula, $\rho_S = \frac{6}{\pi} \arcsin \frac{\rho_Z}{2}$, for any margins. For samplers and samples, the `nsim` values are sorted and assigned in the rank order of Z_i (rank reordering), which is the same construction with the empirical quantile function: each input keeps exactly the values drawn from its distribution, and only their order is changed. All 34 generators of `rDistr`, all random number generators of R and the distributions of measured data can thus be correlated.

Correlation. With `cor.type = "copula"`, $\mathbf{R}_Z = \mathbf{R}$; the Pearson correlation of non-Gaussian margins is then slightly smaller (*e.g.* 0.786 instead of 0.80 for two rectangular inputs). With `cor.type = "pearson"`, each element of $\mathbf{R}_Z$ is found by root finding such that $\mathrm{cor}(X_i, X_j) = R_{ij}$, with the expectation evaluated by Gauss-Hermite quadrature (NORTA, Cario & Nelson 1997); for two rectangular margins this is $\rho_Z = 2\sin(\pi R_{ij}/6)$. The GUM uncertainty framework and the Monte Carlo method then use the same input covariance matrix, as required for the validation of the GUF by the Monte Carlo method (JCGM 101:2008, 8). Correlations that cannot be attained with the given margins give an error; for *t*-margins with $\nu_i \leq 4$ (the sample correlation has no finite variance), the copula value $\mathbf{R}_Z = \mathbf{R}$ is kept.

Use with propagate. The samples carry the attribute `"budget"` with the stated estimates, standard uncertainties, degrees of freedom and distributions. `propagate(..., type = "sim", cov = "full")` then uses these values for the GUM uncertainty framework, together with the correlation matrix of the samples, and the samples themselves for the Monte Carlo simulation. The adaptive Monte Carlo procedure (`adaptive = TRUE`) draws new samples from the budget without correlation and must not be used with `corDist` samples; increase `nsim` instead.

Value

A matrix with `nsim` rows and one named column per input quantity, with the attributes `"budget"` (data frame: `name`, `mean`, `u`, `df`, `dist`) and `"corMat.copula"` (the copula correlation matrix $\mathbf{R}_Z$ used).

Author(s)

Andrej-Nikolai Spiess

References

Copulas:
Copulas for uncertainty analysis.
Possolo A.
Metrologia (2010), **47**: 262-271.
doi:10.1088/00261394/47/3/017

Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix.
Cario MC, Nelson BL.
Technical report, Northwestern University (1997).

Probability distributions of input quantities:
Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

See Also

propagate, makeDat, rDistr

Examples

```
## Two correlated rectangular inputs (r = 0.8), Y = X1 + X2:
## u(y) = sqrt(1 + 1 + 2 * 0.8) = 1.897 for GUF and Monte Carlo
R <- matrix(c(1, 0.8, 0.8, 1), 2)
X <- corDist(mean = c(X1 = 10, X2 = 5), u = c(1, 1), dist = "rect", cor = R, nsim = 1E5)
cor(X)[1, 2]
RES <- propagate(expression(X1 + X2), X, type = "sim", cov = "full", check = FALSE)
c(RES$prop["u.1"], RES$sim["u.MC"])

## Mixed margins: t (10 degrees of freedom), curvilinear trapezoidal
## and a lognormal given by a sampler
R3 <- matrix(c(1, 0.5, -0.3, 0.5, 1, 0, -0.3, 0, 1), 3)
X3 <- corDist(mean = c(a = 1, b = 2, z = NA), u = c(0.1, 0.2, NA),
              dist = list("t", "ctrap", function(n) rlnorm(n, 0.5, 0.3)),
              df = c(10, Inf, Inf), d = c(NA, 0.05, NA), cor = R3, nsim = 1E5)
round(cor(X3), 2)
propagate(expression(a * b + z), X3, type = "sim", cov = "full", check = FALSE)

## Internal samplers of the package (trapezoidal, 4P beta) and a measured sample
meas <- c(50.3, 49.1, 51.2, 50.8, 48.7, 50.1, 49.9, 51.5, 50.4, 49.6)
X4 <- corDist(mean = c(A = NA, B = NA, M = NA), u = NA,
              dist = list(function(n) propagate:::rtrap(n, 0, 1, 2, 3),
                          function(n) propagate:::rbeta2(n, 2, 5, 10, 12), meas),
              cor = matrix(c(1, 0.7, 0.4, 0.7, 1, 0.2, 0.4, 0.2, 1), 3), nsim = 1E5)
round(cor(X4), 2)
attr(X4, "budget")
```

---

datasets *Datasets from the GUM (JCGM 100:2008), its Supplements 1 and 2 (JCGM 101:2008, JCGM 102:2011), NIST TN 1900 and EURACHEM QUAM*

Description

Input data of the examples of Annex H of the GUM (JCGM 100:2008), of clause 9 of its supplements (JCGM 101:2008, JCGM 102:2011), of NIST TN 1900 and of the EURACHEM/CITAC Guide QUAM, in the formats used by propagate, propagateMV and makeDat. The examples are evaluated in the demos demo("JCGM100") (H.1 to H.6), demo("JCGM101") (S1.9.3, S1.9.5), demo("JCGM102") (S2.9.4, S2.9.5), demo("NIST1900") (TN.E2, TN.E11) and demo("QUAM") (QUAM.A4, QUAM.A5); H.1 is also used in demo("JCGM106") for a conformity assessment according to JCGM 106:2012 (see conformity).

Usage

```
data(H.1); data(H.2); data(H.3); data(H.4); data(H.5); data(H.6)
data(S1.9.3); data(S1.9.5)
data(S2.9.4); data(S2.9.5)
data(TN.E2); data(TN.E11); data(QUAM.A4); data(QUAM.A5)
```

Format

`H.1` GUM H.1, end-gauge calibration, Table H.1: a matrix with the rows `estimate`, `u` and `nu` and the columns `Ls` (length of the standard, nm), `d` (difference in length, nm), `da` (difference of the thermal expansion coefficients, 1/degC), `theta` (deviation from 20 degC, degC), `as` (thermal expansion coefficient of the standard, 1/degC) and `dt` (temperature difference, degC); for `propagate(type = "stat")`. Model (GUM H.3): $l = l_S + d - l_S(\delta\alpha\,\theta + \alpha_S\,\delta\theta)$.

`H.2` GUM H.2, simultaneous resistance and reactance measurement, Table H.2: a data frame with five sets of simultaneous indications of the amplitude `V` (V) of a sinusoidally-alternating potential difference, the amplitude `I` (A) of the alternating current and the phase-shift angle `phi` (rad). The measurands are related to the inputs by Ohm's law (GUM H.7): $R = (V/I)\cos\phi$, $X = (V/I)\sin\phi$, $Z = V/I$; since $Z^2 = R^2 + X^2$, there are only two independent output quantities. The indications are correlated and evaluated with `type = "raw"`, `cov = "full"`.

`H.3` GUM H.3, calibration of a thermometer, Table H.6: a data frame with $n = 11$ thermometer readings `tk` (degC, negligible uncertainty) and the measured corrections `bk` = $t_{R,k} - t_k$ (degC) against reference temperatures in the range 21 degC to 27 degC. A linear calibration curve $b(t) = y_1 + y_2(t - t_0)$, t_0 = 20 degC (GUM H.12), is fitted by least squares; intercept and slope, with their variances and covariance, predict the correction and its uncertainty at any temperature t, *e.g.* with `predictNLS`.

`H.4` GUM H.4, measurement of activity, Table H.7: a data frame with six cycles of counting of a standard, a blank and a sample: the times `ts`, `tb`, `tx` (min) from the reference time to the midpoints of the dead-time-corrected counting intervals T_0 = 60 min, and the numbers of counts `Cs`, `Cb`, `Cx`. The decay- and background-corrected count rates are $R = (C - C_B)\,e^{\lambda t}/T_0$ with λ = 1.25894e-4 1/min, and the activity concentration of the sample is $A_x = A_S(m_S/m_x)(\bar{R}_x/\bar{R}_S)$ (GUM H.19, H.21) with A_S = 0.1368 Bq/g, m_S = 5.0192 g, m_x = 5.0571 g (GUM H.4.3).

`H.5` GUM H.5, analysis of variance, Table H.9: a data frame with the columns `day` (day number), `Vj` (daily mean, V) of $K = 5$ observations of a Zener voltage standard and `s` (their standard deviation, microvolt).

`H.6` GUM H.6, Rockwell C hardness, Table H.10: an uncertainty budget for `makeDat` (columns `name`, `value`, `u`, `a`, `dist`) with the mean indentation depth `d`, the machine difference `Dc`, the block variation `Db` (triangular) and the national standard (column `DS`, Δ_S in the model), in Rockwell units (0.002 mm). Model (GUM H.33a): $h = 100 - d - \Delta_c - \Delta_b - \Delta_S$.

`S1.9.3` JCGM 101:2008, 9.3, mass calibration, Table 5: an uncertainty budget for `makeDat` with the conventional mass `mR` and the mass difference `dmR` of the reference weight (mg, Gaussian), and the densities of air `ra`, of the weight `rW` and of the reference weight `rR` (kg/m^3, rectangular). Model (JCGM 101 Eq. 24): $\delta m = (m_{R,c} + \delta m_{R,c})[1 + (\rho_a - \rho_{a0})(1/\rho_W - 1/\rho_R)] - m_{nom}$, with ρ_{a0} = 1.2 kg/m^3 and m_{nom} = 100 g.

`S1.9.5` JCGM 101:2008, 9.5, gauge block calibration, Table 10: an uncertainty budget for `makeDat` with the columns `name`, `value`, `u`, `df`, `a`, `d` and `dist` (t, rectangular, Gaussian, arcsine and

curvilinear trapezoidal distributions; lengths in nm). Model (JCGM 101 Eq. 37): $\delta L = L_s + D + d_1 + d_2 - L_s[\delta\alpha(\theta_0 + \Delta) + \alpha_s\delta\theta] - L_{nom}$, L_{nom} = 50 mm.

`S2.9.4` JCGM 102:2011, 9.4, simultaneous resistance and reactance measurement, Table 8: the data of GUM H.2 (`H.2`) with a sixth set of indications, a data frame with `V` (V), `I` (A) and `phi` (rad).

`S2.9.5` JCGM 102:2011, 9.5, Celsius temperature with a resistance thermometer: a list with `inputs` (Table 12: `R0` in Ohm, `A` in 1/degC, `B` in 1/degC^2, `RS` in Ohm, with standard uncertainties), `cor` (Table 13: correlation matrix of `R0`, `A`, `B`) and `ratios` (Table 14: ten measured resistance ratios r_j with their standard uncertainty). Implicit model (JCGM 102 Eq. 31): $R_0(1 + A\theta + B\theta^2) - rR_S = 0$.

`TN.E2` NIST TN 1900, E2, surface temperature (Exhibit 2): a data frame with the daily maximum temperatures `t` (degC) read on 22 days of May 2012. Mean 25.59 degC, $u = s/\sqrt{22}$ = 0.872 degC.

`TN.E11` NIST TN 1900, E11, microwave step attenuator (EA-4/02, S7): a list with `LS`, four observations of the attenuation difference (dB; mean 30.0402 dB, $u = s/\sqrt{4}$ = 0.0091 dB), and `budget`, the input quantities of Exhibit 16 with estimates, standard uncertainties `u` (dB) and distributions (`"t3"`: Student t with 3 degrees of freedom, rescaled to standard deviation `u`; arcsine and rectangular inputs with standard deviation `u`). Model: $L_X = L_S + \delta L_S + \delta L_D + \delta L_M + \delta L_K + \delta L_{ib} - \delta L_{ia} + \delta L_{0b} - \delta L_{0a}$.

`QUAM.A4` EURACHEM QUAM, A4, organophosphorus pesticides in bread, Table A4.2: a data frame with 15 duplicate analyses `D1`, `D2` (mg/kg) of `residue`, taken in different runs. The standard deviation of the relative differences, 0.382, divided by $\sqrt{2}$, gives the relative standard uncertainty 0.27 of a single result (intermediate precision). In Table A4.2, the means of rows 13 and 14 are misprinted (0.75 and 0.10 instead of 0.075 and 0.01); the dataset contains the duplicate values only.

`QUAM.A5` EURACHEM QUAM, A5, cadmium release from ceramic ware, Table A5.2: a data frame with the AAS calibration: three replicate absorbances (column `absorbance`) at each of five cadmium concentrations `conc` (mg/L). The least-squares line has slope 0.2410 and intercept 0.0087 with residual standard deviation 0.005486; for the leach solution, measured twice, c_0 = 0.26 mg/L with $u(c_0)$ = 0.018 mg/L (QUAM, A5 and E.4).

Note

GAUGING: Complete worked examples of JCGM 100:2008 (GUM, Annex H), JCGM 101:2008, JCGM 102:2011, JCGM 106:2012, the EMUE Compendium, NIST TN 1900, EURACHEM QUAM and correlated non-Gaussian inputs, with comparisons against the published values: `demo("JCGM100")`, `demo("JCGM101")`, `demo("JCGM102")`, `demo("JCGM106")`, `demo("EMUE")`, `demo("NIST1900")`, `demo("QUAM")`, `demo("COPULA")`; see doGauge.

References

GUM:
Evaluation of measurement data - Guide to the expression of uncertainty in measurement.
JCGM 100:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1002008E

GUM Supplements and conformity assessment:
Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in

measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

Evaluation of measurement data - Supplement 2 to the "Guide to the expression of uncertainty in measurement" - Extension to any number of output quantities.
JCGM 102:2011. Joint Committee for Guides in Metrology (2011).
doi:10.59161/JCGM1022011

Evaluation of measurement data - The role of measurement uncertainty in conformity assessment.
JCGM 106:2012. Joint Committee for Guides in Metrology (2012).
doi:10.59161/JCGM1062012

Collections of worked examples:
Compendium of examples: Good practice in evaluating measurement uncertainty (EMUE), version 2.
van der Veen AMH, Cox MG (Eds.).
Zenodo (2021).
doi:10.5281/zenodo.5142180

Quantifying Uncertainty in Analytical Measurement (QUAM:2012.P1), 3rd edition.
Ellison SLR, Williams A (Eds.).
EURACHEM/CITAC Guide CG 4 (2012). ISBN 978-0-948926-30-3.
https://www.eurachem.org/index.php/publications/guides/quam

Simple Guide for Evaluating and Expressing the Uncertainty of NIST Measurement Results.
Possolo A.
NIST Technical Note 1900, National Institute of Standards and Technology (2015).
doi:10.6028/NIST.TN.1900

See Also

propagate, propagateMV, makeDat; demo(package = "propagate")

Examples

```
## GUM H.1: l - 50 mm = 838 nm, u = 32 nm
data(H.1)
propagate(expression(Ls + d - Ls * (da * theta + as * dt) - 50000000), H.1,
          type = "stat", alpha = 0.01, nsim = 1E4, check = FALSE)

## NIST TN 1900, E2: mean 25.59 degC, u = 0.872 degC, 95 % coverage interval (23.8, 27.4) degC
data(TN.E2)
propagate(expression(t), as.matrix(TN.E2), type = "raw", nsim = 1E4, check = FALSE)

## EURACHEM QUAM, A5: calibration line and u(c0) = 0.018 mg/L for c0 = 0.26 mg/L (two readings)
data(QUAM.A5)
FIT <- lm(absorbance ~ conc, data = QUAM.A5); S <- summary(FIT)$sigma
Sxx <- sum((QUAM.A5$conc - mean(QUAM.A5$conc))^2)
S/coef(FIT)[2] * sqrt(1/2 + 1/15 + (0.26 - mean(QUAM.A5$conc))^2/Sxx)
```

doGauge *Gauge propagate against published examples of measurement uncertainty evaluation*

Description

Runs the demos of the package that reproduce the worked examples of the GUM (JCGM 100:2008, Annex H), its Supplements 1 and 2 (JCGM 101:2008, JCGM 102:2011), JCGM 106:2012 (conformity assessment), the EMUE Compendium, NIST Technical Note 1900 and the EURACHEM/CITAC Guide QUAM and, in demo `COPULA`, correlated non-Gaussian inputs (JCGM 101:2008, 9.4.3; Possolo 2010) checked against exact results. It writes their complete console output (code and results) to one HTML or PDF file per demo. In each demo, every recomputable published value is compared with the result of **propagate**, and the result of each comparison (`PASS`, `NOTE` or `DIFF`) is highlighted in the output. A summary table of all comparisons is printed and returned.

Usage

```
doGauge(demo = c("JCGM100", "JCGM101", "JCGM102", "JCGM106", "EMUE", "NIST1900",
               "QUAM", "COPULA"), output = c("html", "pdf"), outdir = tempdir(),
                 fontsize = 20, silent = TRUE,
                 dir = system.file("demo", package = "propagate"))
```

Arguments

`demo`	one or several of the demos: `"JCGM100"` (GUM, Annex H), `"JCGM101"` (GUM Supplement 1, clause 9), `"JCGM102"` (GUM Supplement 2, clause 9), `"JCGM106"` (JCGM 106:2012, conformity assessment), `"EMUE"` (EMUE Compendium), `"NIST1900"` (NIST TN 1900), `"QUAM"` (EURACHEM/CITAC Guide CG 4, Appendix A) and `"COPULA"` (correlated non-Gaussian inputs with `corDist`: JCGM 101:2008, 9.4.3, Possolo 2010, and exact results). Default: all.
`output`	`"html"` (default) or `"pdf"`: the format of the files written.
`outdir`	the directory in which the files `<demo>.html` or `<demo>.pdf` are written. The default is a temporary directory; use, *e.g.*, `getwd()` to keep the files.
`fontsize`	the font size of the output in pixels (HTML); for PDF, the size is scaled accordingly and reduced if necessary, so that the longest line fits the page width.
`silent`	logical. If `TRUE`, messages and warnings issued while the demos run are suppressed. The progress of the demos and the summary table are always shown.
`dir`	the directory of the demo files; only needed to run modified copies of the demos.

Details

The published value is given in the demos as a character string, so that its number of decimals defines the rounding tolerance (half a unit of the last decimal). For Monte Carlo results, four Monte Carlo standard errors of both calculations (**propagate** and the published run) are added; where a published value was computed from rounded intermediate values that are printed in the same document, their propagated rounding is added as well (argument `extra` of the internal function gauge of the demos).

- PASS: the value of **propagate** agrees with the published value within the tolerance.
- NOTE: a documented discrepancy of the published value, *e.g.* a truncated instead of a rounded value, a value carried over from another calculation, or an inconsistency between a printed equation and its numerical result. The cause is explained in a comment in the demo.
- DIFF: a deviation that is neither within the tolerance nor documented.

The demos use `nsim = 1E6` Monte Carlo trials; running all demos takes several minutes. The expected result is `DIFF = 0` for all demos. They can also be run individually, *e.g.* `demo("JCGM101", package = "propagate")`.

Value

Invisibly, a list with

`files`	the paths of the files written.
`summary`	a data frame with the number of PASS, NOTE and DIFF results of each demo.

Author(s)

Andrej-Nikolai Spiess

References

GUM:
Evaluation of measurement data - Guide to the expression of uncertainty in measurement.
JCGM 100:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1002008E

GUM Supplements and conformity assessment:
Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

Evaluation of measurement data - Supplement 2 to the "Guide to the expression of uncertainty in measurement" - Extension to any number of output quantities.
JCGM 102:2011. Joint Committee for Guides in Metrology (2011).
doi:10.59161/JCGM1022011

Evaluation of measurement data - The role of measurement uncertainty in conformity assessment.
JCGM 106:2012. Joint Committee for Guides in Metrology (2012).
doi:10.59161/JCGM1062012

Collections of worked examples:
Compendium of examples: Good practice in evaluating measurement uncertainty (EMUE), version 2.
van der Veen AMH, Cox MG (Eds.).
Zenodo (2021).
doi:10.5281/zenodo.5142180

Quantifying Uncertainty in Analytical Measurement (QUAM:2012.P1), 3rd edition.
Ellison SLR, Williams A (Eds.).

EURACHEM/CITAC Guide CG 4 (2012). ISBN 978-0-948926-30-3.
https://www.eurachem.org/index.php/publications/guides/quam

Simple Guide for Evaluating and Expressing the Uncertainty of NIST Measurement Results.
Possolo A.
NIST Technical Note 1900, National Institute of Standards and Technology (2015).
doi:10.6028/NIST.TN.1900

Copulas:
Copulas for uncertainty analysis.
Possolo A.
Metrologia (2010), **47**: 262-271.
doi:10.1088/00261394/47/3/017

See Also

propagate, propagateMV, conformity, corDist, demo(package = "propagate")

Examples

```
## one demo, file written to tempdir()
# res <- doGauge("JCGM106")
# res$summary
## all demos (several minutes):
# doGauge()
```

---

fitDistr *Fitting distributions to observations/Monte Carlo simulations*

Description

This function fits 43 different continuous distributions to the Monte Carlo simulation results obtained by propagate, or to any other vector of observations, and ranks them by ascending BIC. By default, the distributions are fitted by maximum likelihood to the observations; the (weighted) least-squares histogram fit used up to version 1.2-0 is available with method = "nls".

Usage

```
fitDistr(object, method = c("mle", "nls"), nbin = 100, nmax = 1E5,
         weights = FALSE, verbose = TRUE, brute = c("fast", "slow"),
         plot = c("hist", "qq", "none"), distsel = NULL, ...)
```

Arguments

object	an object of class 'propagate' or a vector containing observations.
method	"mle" (default) for maximum likelihood estimation on the observations, "nls" for the least-squares fit of the density to the histogram. See 'Details'.

nbin	the number of bins in the histogram. For method = "mle", the histogram is only used for the plots and for fitted/residuals.
nmax	method = "mle" only: the maximum number of observations used for fitting. Larger data sets, such as propagate output with 10^6 values, are reduced to an evenly spaced, deterministic subsample of nmax values.
weights	method = "nls" only: numeric or logical. Either a numeric vector of weights with one weight per histogram bin, or if TRUE, the distributions are fitted with weights = 1/(counts per bin); bins with zero counts get weight 1. Ignored, with a message, for method = "mle".
verbose	logical. If TRUE, steps of the analysis are printed to the console.
brute	complexity of the brute force approach. See 'Details'.
plot	if "hist", a plot with the "best" distribution (in terms of lowest BIC) on top of the histogram is displayed. If "qq", the fitted density values of the best distribution are plotted against the histogram densities, bin by bin, with the identity line (a density-density plot, not a quantile-quantile plot).
distsel	a vector of distribution numbers to select from the complete cohort as listed below, e.g. c(1:10, 15). All outputs (fit, par, se, dens) refer to the selected distributions, sorted by BIC.
...	other parameters to be passed to the plots.

Details

Fits the following 43 distributions (definitions in Johnson, Kotz & Balakrishnan 1994, 1995 and Leemis & McQueston 2008, unless another reference is given; parametrisations as in the density functions of the package):
1) Normal distribution (dnorm)
2) Skewed-normal distribution (propagate:::dsn)
3) Generalized normal distribution (propagate:::dgnorm)
4) Log-normal distribution (dlnorm)
5) Scaled and shifted t-distribution (propagate:::dst) => JCGM 101:2008, 6.4.9.2. The parameter sd is the scale, not the standard deviation.
6) Logistic distribution (dlogis)
7) Uniform distribution (dunif)
8) Triangular distribution (propagate:::dtriang)
9) Trapezoidal distribution (propagate:::dtrap)
10) Curvilinear Trapezoidal distribution (propagate:::dctrap) => JCGM 101:2008, 6.4.3
11) Gamma distribution (dgamma)
12) Inverse Gamma distribution (propagate:::dinvgamma)
13) Cauchy distribution (dcauchy)
14) Laplace distribution with standard deviation sigma (propagate:::dlaplace)
15) Gumbel distribution (propagate:::dgumbel)
16) Johnson SU distribution (propagate:::dJSU)
17) Johnson SB distribution (propagate:::dJSB)
18) Three-parameter Weibull distribution (propagate:::dweibull2)
19) Two-parameter beta distribution (dbeta)
20) Four-parameter beta distribution (propagate:::dbeta2)
21) Arcsine distribution (propagate:::darcsin)

22) Von Mises distribution (`propagate:::dmises`)
23) Inverse Gaussian distribution (`propagate:::dinvgauss`)
24) Generalized Extreme Value distribution (`propagate:::dgevd`)
25) Rayleigh distribution (`propagate:::drayleigh`)
26) Chi-square distribution (`dchisq`)
27) Exponential distribution (`dexp`)
28) F-distribution (`df`)
29) Burr type VIII distribution (`propagate:::dburr`) => Johnson, Kotz & Balakrishnan (1994)
30) Chi distribution (`propagate:::dchi`)
31) Inverse Chi-square distribution (`propagate:::dinvchisq`)
32) Raised cosine distribution (`propagate:::dcosine`)
33) Pareto distribution (`propagate:::dpareto`)
34) Levy distribution (`propagate:::dlevy`)
35) Gompertz distribution (`propagate:::dgompertz`)
36) 3P log-normal distribution, Johnson SL (`propagate:::dlnorm3`): $\xi + \exp(\mu + \sigma Z)$, positive skewness with a free lower limit ξ
37) 3P gamma distribution, Pearson type III (`propagate:::dgamma3`): gamma distribution shifted by ξ
38) Rectangular-normal convolution (`propagate:::dunorm`): $g(x) = [\Phi((x-a)/\sigma) - \Phi((x-b)/\sigma)]/(b-a)$, the output of models dominated by one rectangular input
39) Exponential power distribution (`propagate:::dexpow`): $g(x) \propto \exp(-(|x-\mu|/\alpha)^\beta)$; Laplace for $\beta = 1$, normal for $\beta = 2$, rectangular for $\beta \to \infty$
40) Skew-t distribution (`propagate:::dskewt`, Azzalini & Capitanio 2003): skewness and heavy tails, *e.g.* for ratios of *t*-distributed inputs
41) Rice distribution (`propagate:::drice`): magnitude of a complex quantity with Gaussian real and imaginary parts; for large arguments, the exponentially scaled Bessel function I_0 is replaced by its asymptotic expansion
42) Folded normal distribution (`propagate:::dfnorm`): absolute value of a normal variable
43) Two-component normal mixture (`propagate:::dnmix`): bimodal output distributions

All density functions return 0 outside their support (and `-Inf` for `log = TRUE`). Distributions whose support cannot contain the data (*e.g.* the log-normal for negative values, or the 2P beta outside $(0, 1)$) are skipped. The density functions `propagate:::d...` are internal functions (hence `:::`). The code for the density functions, and the parametrisations used for maximum likelihood fitting (internal function `.mleSpec`), are in file "distr-densities.R".

Maximum likelihood (`method = "mle"`, default).
For each distribution with density $g(x \mid \theta)$, the parameters are estimated by maximising the log-likelihood of the n observations,

$$\ln L(\theta) = \sum_{i=1}^{n} \ln g(x_i \mid \theta),$$

and the distributions are ranked by

$$\mathrm{BIC} = -2 \ln L(\hat{\theta}) + k \ln n, \qquad \mathrm{AIC} = -2 \ln L(\hat{\theta}) + 2k,$$

with k the number of distribution parameters. Unlike the histogram fit, the BIC is then a proper likelihood-based criterion that does not depend on `nbin`. `stat` also reports ΔBIC to the best distribution and the Schwarz weights $w_j = \exp(-\Delta\mathrm{BIC}_j/2) / \sum_l \exp(-\Delta\mathrm{BIC}_l/2)$, which approximate the posterior probabilities of the candidates: a weight close to 1 indicates a clear winner,

several similar weights indicate that the data cannot distinguish between the candidates (*e.g.* nested families such as the normal and the *t*-distribution with large ν). With large n, the BIC discriminates strongly, and the best distribution is the best *approximation* among the candidates, which need not contain the true distribution.

Parametrisation. The likelihood is maximised over unconstrained internal parameters: scale and shape parameters on the log scale, modes and other interior proportions on the logit scale, and support limits as $\min(x) - s - e^{\vartheta}$ and $\max(x) + s + e^{\vartheta}$, so that the support always contains the data. Here $s = (\max(x) - \min(x))/n$ is the mean spacing of the observations: keeping support limits at least one spacing outside the data bounds the likelihood of densities that are infinite at a support limit (arcsine, 3P Weibull with shape < 1), whose maximum likelihood would otherwise be infinite. For the uniform distribution, this gives practically the minimum-variance unbiased estimates $\min(x) - (\max(x) - \min(x))/(n-1)$ and $\max(x) + (\max(x) - \min(x))/(n-1)$ of the limits. Dependent parameters are mapped internally, *e.g.* the mode of the triangular distribution as $b = a + (c-a)\,p$ with $0 < p < 1$, or the upper limit $\xi + \lambda$ of the Johnson SB distribution.

Optimisation. A grid of starting values around moment-based estimates is evaluated (`brute = "fast"`: 3 values per parameter, 3^k points; `brute = "slow"`: 5 values, 5^k points), with steps of a factor 10 for scale/shape parameters and of one standard deviation for location parameters. The likelihood is maximised by Nelder-Mead from the best grid point (`"fast"`) or the best three (`"slow"`), followed by a BFGS refinement.

For the normal mixture 43), the component standard deviations are bounded below by 1 % of the sample standard deviation, because the likelihood of a mixture is unbounded when one component collapses onto a single value.

Standard errors. The standard errors are obtained from the inverse Hessian of $-\ln L$ with respect to the internal parameters, transformed to the parameters of the density by the delta method. Parameters that are limits of the support are non-regular: their estimates converge to the sample extremes at rate $1/n$ rather than $1/\sqrt{n}$, and the Hessian does not describe their uncertainty. They are held fixed for the Hessian, and their standard errors are `NA`. This applies to the limits of the uniform, triangular, trapezoidal, curvilinear trapezoidal, 4P beta, arcsine and Johnson SB distributions, to the mode of the triangular and the inner corners of the trapezoidal distribution (kinks of the density), the location of the 3P Weibull and Rayleigh distributions, the scale of the Pareto distribution and σ of the cosine distribution.

Sample size. The likelihood is evaluated on all observations, or on an evenly spaced, deterministic subsample of `nmax` values (no random numbers are drawn). 10^5 values are ample to identify a distribution, and the run time is then similar to that of the histogram fit.

Histogram fit (`method = "nls"`**, versions up to 1.2-0).**
The density is fitted by (weighted) least squares to the histogram densities at the bin midpoints, with `minpack.lm::nls.lm`. The parameter space is searched with a brute force approach, extended over three orders of magnitude $(0.1, 1, 10)\times\beta_i$ when `brute = "fast"`, or five orders $(0.01, 0.1, 1, 10, 100)\times\beta_i$ when `brute = "slow"`. The goodness-of-fit is calculated with `BIC` from the (weighted) Gaussian log-likelihood of the NLS fit, as in `stats:::logLik.nls`:

$$\ln(L) = \frac{1}{2}\left(\sum_{i=1}^{N}\ln(w_i) - N\left(\ln(2\pi) + 1 - \ln(N) + \ln\sum_{i=1}^{N} r_i^2\right)\right)$$

$$\mathrm{BIC} = -2\ln(L) + k\ln(N)$$

with $r_i = \sqrt{w_i}\,(\hat{g}_i - g(x_i))$ the weighted residuals between the histogram density $\hat{g}_i$ and the fitted density at the bin midpoints x_i, w_i the weights, N the number of bins with non-zero weight, and

k the number of distribution parameters plus one (for the residual variance). Because N is the number of bins, not of observations, these BIC values depend on `nbin` and are a heuristic criterion for comparing distributions on the same histogram. Setting `weights = TRUE` gives more weight to bins with low counts; also check whether the ranking remains stable when `nbin` is changed.

Use. Besides characterising the output distribution of `propagate`, `fitDistr` can identify a distribution for raw data prior to Monte Carlo simulation. A decent number of observations is needed to distinguish between similar distributions. In contrast to software that ranks by residual sum-of-squares or Kolmogorov-Smirnov/Anderson-Darling statistics, the BIC penalises the number of parameters and thus compensates for overfitting; the fits are maximum likelihood fits as in `MASS::fitdistr`.

Value

A list with the following items:

`stat`: a data frame of the fitted distributions, sorted by ascending BIC. For `method = "mle"`: the number of parameters `k`, `logLik`, `AIC`, `BIC`, `dBIC` (difference to the lowest BIC) and the Schwarz `weight`; distributions that could not be fitted, or whose support excludes the data, have `NA`. For `method = "nls"`: BIC, the (weighted) residual sum-of-squares RSS and the mean squared residual MSE.
`fit`: a list with the fit for each distribution, sorted as in `stat`. For `"mle"`: a list with `par`, `se`, `logLik`, `k`, `n`, `convergence` (of `optim`) and the internal parameters `theta`; for `"nls"`: the `minpack.lm::nls.lm` object.
`par`: a list of the estimated parameters, named as the arguments of the density functions.
`se`: a list of the parameters' standard errors (`NA` for non-regular parameters with `"mle"`).
`dens`: a list with the density functions of the fitted distributions, sorted and named as in `fit`.
`bestfit`: the best model in terms of lowest BIC.
`bestpar`: the parameters of `bestfit`.
`bestse`: the parameters' standard errors of `bestfit`.
`fitted`: the density of `bestfit` at the histogram bin midpoints.
`residuals`: the histogram density minus `fitted`.
`method`: the fitting method.
`n`: the number of observations used (for `"mle"`, at most `nmax`).

Author(s)

Andrej-Nikolai Spiess

References

Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

Distributions generated by perturbation of symmetry with emphasis on a multivariate skew t-distribution.
Azzalini A, Capitanio A.
Journal of the Royal Statistical Society B (2003), **65**: 367-389.
doi:10.1111/14679868.00391

Continuous Univariate Distributions, Volume 1, 2nd edition.
Johnson NL, Kotz S, Balakrishnan N.
Wiley, New York (1994). ISBN 0-471-58495-9.

Continuous Univariate Distributions, Volume 2, 2nd edition.
Johnson NL, Kotz S, Balakrishnan N.
Wiley, New York (1995). ISBN 0-471-58494-0.

Univariate distribution relationships.
Leemis LM, McQueston JT.
The American Statistician (2008), **62**: 45-53.
doi:10.1198/000313008X270448

Estimating the dimension of a model.
Schwarz G.
The Annals of Statistics (1978), **6**: 461-464.
doi:10.1214/aos/1176344136

Model Selection and Multimodel Inference.
Burnham KP, Anderson DR.
Springer, New York, 2nd edition (2002).
doi:10.1007/b97636

Examples

```
## Linear example, small error
## => Normal distribution.
## Don't set 'distsel' for full test
EXPR1 <- expression(x + y)
x <- c(5, 0.1)
y <- c(1, 0.01)
DF1 <- cbind(x, y)
RES1 <- propagate(expr = EXPR1, data = DF1, type = "stat",
                  nsim = 10000, check = FALSE)
FIT1 <- fitDistr(RES1, distsel = 1:10)
FIT1$stat


## Identify the distribution of raw data: t-distributed sample.
set.seed(1)
X <- propagate:::rst(10000, mean = 3, sd = 2, df = 5)  # internal generator, see ?rDistr
FIT2 <- fitDistr(X, distsel = 1:10)
FIT2$stat          # the t-distribution wins, weight close to 1
FIT2$bestpar; FIT2$bestse
```

`interval` *Range of a function over intervals of its input variables*

Description

Calculates the range $[\min f, \max f]$ of a model $f(x_1, \ldots, x_N)$ when each input variable is only known to lie in an interval $[A_i, B_i]$. The function is evaluated on a grid over the box of input intervals ("vertex" or "grid" method), and the grid minimum and maximum are refined by box-constrained optimisation. Unlike naive interval arithmetic, the result is not inflated by the dependency problem.

Usage

```
interval(df, expr, seq = 10, plot = TRUE, refine = TRUE, maxgrid = 1E6)

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

Arguments

`df`	a data frame or matrix with two rows, the lower limits A_i in the first row and the upper limits B_i in the second row. Column names must correspond to the variable names in `expr`; further columns are ignored.
`expr`	an expression, such as `expression(x/y)`.
`seq`	the number of grid points per variable from A_i to B_i (at least 2, *i.e.* the interval limits); 0 is added to intervals containing zero.
`plot`	logical. If `TRUE`, the grid evaluations are plotted, with the range as blue lines.
`refine`	logical. If `TRUE`, the grid minimum and maximum are refined by `optim` (method `"L-BFGS-B"`, bounded by the input intervals), which finds extrema between grid points.
`maxgrid`	the maximal number of grid points $\prod_i n_i$, with n_i = `seq` (+1 if 0 is added) the number of grid points of variable i; the grid grows as `seq`^{N} with the number N of variables.
`x`	an object of class `interval`.
`...`	not used.

Details

Interval arithmetic and the dependency problem. For two intervals $[x_1, x_2]$ and $[y_1, y_2]$, the basic operations $\circ \in \{+, -, \cdot, /\}$ of interval arithmetic are

$$[x_1, x_2] \circ [y_1, y_2] = [\min(x_i \circ y_j), \max(x_i \circ y_j)], \quad i, j \in \{1, 2\}$$

(for division, $0 \notin [y_1, y_2]$). Applied operation by operation, this overestimates the range when a variable occurs more than once in an expression, because each occurrence is treated as an independent variable (dependency problem): *e.g.*, $x - x^2$ with $x \in [0, 1]$ gives $[0, 1] - [0, 1] = [-1, 1]$, while the true range is $[0, 0.25]$.

Vertex and grid method. `interval` avoids this by evaluating f as a whole. With `seq` = 2, f is evaluated at the 2^N vertices of the box (all combinations of the interval limits); this gives the exact range if f is monotone in each variable. If not, the extrema may lie inside the box: a larger `seq`

evaluates f on a finer grid (`seq`^{N} points, with 0 added to intervals containing zero, *e.g.* for even powers), and `refine = TRUE` refines the grid minimum and maximum by bounded optimisation.

Inner approximation. The result is the range of the evaluated points and optimisation results, which can only be narrower than (or equal to) the true range: it is an inner approximation, not a guaranteed enclosure as in rigorous interval arithmetic. For functions with several local extrema, use a finer grid. Grid points where f is not finite (outside its domain) are ignored with a warning.

Value

An object of class `interval`: a vector with the elements `lower` and `upper`, and the attributes `argmin` and `argmax` (the input values at the extrema) and `ngrid` (the number of grid points).

Author(s)

Andrej-Nikolai Spiess

References

Interval arithmetic and the dependency problem:
Introduction to Interval Analysis.
Moore RE, Kearfott RB, Cloud MJ.
SIAM, Philadelphia (2009).
doi:10.1137/1.9780898717716

Comparison to Monte Carlo and error propagation:
Interval arithmetic in power flow analysis.
Wang Z, Alvarado FL.
IEEE Transactions on Power Systems (1992), **7**: 1341-1349.
doi:10.1109/59.207353

Computer implementation:
Interval arithmetic: From principles to implementation.
Hickey T, Ju Q, Van Emden MH.
Journal of the ACM (2001), **48**: 1038-1068.
doi:10.1145/502102.502106

Complete interval arithmetic and its implementation on the computer.
Kulisch UW.
In: Numerical Validation in Current Hardware Architectures. Cuyt A, Kraemer W, Luther W, Markstein P (Eds.). Springer (2009), *Lecture Notes in Computer Science* **5492**: 7-26.
doi:10.1007/9783642015915

See Also

`propagate`, `makeDat`

Examples

```
## Example 1: even power of an interval containing zero: [0, 1], not [-1, 1].
interval(data.frame(x = c(-1, 1)), expression(x^2))

## Example 2: a nonlinear model, monotone in each variable:
```

```
## the vertices (seq = 2) give the exact range.
EXPR2 <- expression(Cf * sqrt((520 * H * P)/(M * (Temp + 460))))
H <- c(64, 65); M <- c(16, 16.2); P <- c(361, 365); Temp <- c(165, 170); Cf <- c(38.4, 38.5)
DAT2 <- makeDat(EXPR2)
interval(DAT2, EXPR2, seq = 2)

## Example 3: body mass index.
EXPR3 <- expression(m/h^2)
m <- c(79.5, 80.5); h <- c(1.795, 1.805)
DAT3 <- makeDat(EXPR3)
interval(DAT3, EXPR3)

## Example 4: minimum inside the interval. For x in [-2, 1], naive interval arithmetic gives
## [0, 4] - [-2, 1] + 1 = [0, 7] (or [-2, 7] if x^2 is computed as x * x);
## the true range is [0.75, 7].
EXPR4 <- expression(x^2 - x + 1)
x <- c(-2, 1)
DAT4 <- makeDat(EXPR4)
interval(DAT4, EXPR4, seq = 2, refine = FALSE)  ## vertices and 0 only: [1, 7]
interval(DAT4, EXPR4, seq = 2)                  ## refined: [0.75, 7]
curve(x^2 - x + 1, -2, 1)

## Example 5: dependency problem. Naive interval arithmetic gives
## [0, 1] - [0, 1] = [-1, 1]; the true range is [0, 0.25].
EXPR5 <- expression(x - x^2)
x <- c(0, 1)
DAT5 <- makeDat(EXPR5)
interval(DAT5, EXPR5, seq = 2, refine = FALSE)  ## vertices only: [0, 0]
interval(DAT5, EXPR5)                           ## [0, 0.25]
curve(x - x^2, 0, 1)
```

---

makeDat *Create the input data for 'propagate' from the workspace, a file or an uncertainty budget*

Description

Creates the data matrix for propagate from (a) variables in the workspace, (b) a file or data frame with one column per input quantity (repeated observations, mean/u/degrees of freedom, or samples), or (c) a file or data frame in the form of an uncertainty budget, with one row per input quantity. Files can be .csv, .txt, .tsv, .dat, .xlsx or .xls.

Usage

```
makeDat(expr = NULL, file = NULL, data = NULL, layout = c("auto", "columns", "budget"),
      output = c("stat", "sim"), nsim = 1E6, sheet = 1, envir = parent.frame(), ...)
```

Arguments

`expr`	an expression to be used in `propagate`. Its variables select and order the columns of the result. Required for the workspace mode; optional otherwise.
`file`	the name of a file to import (`.csv`, `.txt`, `.tsv`, `.dat`, `.xlsx`, `.xls`). Excel files require package 'readxl'.
`data`	alternatively to `file`, a data frame or matrix already in R.
`layout`	`"columns"`: one column per input quantity; `"budget"`: one row per input quantity; `"auto"`: `"budget"` if there is a column `name` and a column `value` (or `mean`, `x`, `estimate`), otherwise `"columns"`.
`output`	for `layout = "budget"` only: `"stat"` returns means, standard uncertainties and degrees of freedom (`type = "stat"`); `"sim"` returns `nsim` samples from the stated distributions (`type = "sim"`).
`nsim`	the number of samples for `output = "sim"`.
`sheet`	the sheet of an Excel file, by number or name.
`envir`	the environment in which the variables of `expr` are looked up in workspace mode (default: the calling environment).
`...`	further arguments to `read.table` or `read_excel` of package 'readxl'.

Details

(a) Workspace. Without `file` and `data`, the variables of `expr` are taken from `envir` and combined column-wise. Variables of different length (*e.g.* repeated observations with different numbers of observations) are padded with `NA`.

(b) Columns layout. Each column is one input quantity, named as in `expr`; additional columns are dropped. The rows contain either repeated observations (`type = "raw"`), the mean, standard uncertainty and optional degrees of freedom (`type = "stat"`), or samples (`type = "sim"`). Missing observations may be left empty (`NA`).

(c) Budget layout. Each row is one input quantity, as in an uncertainty budget. Recognised columns (case-insensitive, except `u`/`U`):

`name`	name of the quantity, as in `expr` (required)
`value`	estimate (aliases `mean`, `x`, `estimate`; required)
`u`	standard uncertainty
`U`, `k`	expanded uncertainty and coverage factor: $u = U/k$
`a`	half-width of a rectangular, triangular, arcsine or curvilinear trapezoidal distribution
`d`	curvilinear trapezoidal distribution only: half-width of the inexactness of the limits
`dist`	`normal`, `t`, `rect`, `tri`, `arcsine` or `ctrap` (aliases `gaussian`, `norm`; `rectangular`, `uniform`, `unif`; `triangular`, `tria`
`df`	degrees of freedom (aliases `nu`, `dof`); empty = `Inf`

The standard uncertainty is taken from `u`, else from `U/k`, else from the half-width: $u = a/\sqrt{3}$ (rectangular), $a/\sqrt{6}$ (triangular), $a/\sqrt{2}$ (arcsine), $\sqrt{a^2/3 + d^2/9}$ (curvilinear trapezoidal). Degrees of freedom given for Type B inputs (*e.g.* from the reliability of the limits, JCGM 100:2008, G.4.2) are kept; they enter only the effective degrees of freedom of the GUF. If `dist` is empty, an input is `normal` for `df = Inf` and `t` for finite `df`. For a calibration certificate stating U_p, k_p and ν_{eff}, enter `U`, `k` and `df`: this is the *t*-distribution $t_{\nu_{\mathrm{eff}}}(x, (U_p/k_p)^2)$ of JCGM 101:2008, 6.4.9.7.

With `output = "stat"`, the result has the three rows mean, u and ν of `type = "stat"`. `propagate` then samples every input from a t or Gaussian margin with this mean and u, so that rectangular, triangular and arcsine inputs are represented only by their first two moments (a message is given). With `output = "sim"`, every input is sampled from its stated distribution (JCGM 101:2008, 6.4): Gaussian, scaled and shifted t with scale u (6.4.9), rectangular, triangular, arcsine or curvilinear trapezoidal (6.4.3). The samples carry the interpreted budget as attribute `"budget"`; `propagate` with `type = "sim"` then uses its estimates, standard uncertainties and degrees of freedom for the GUM uncertainty framework, so that one call gives both the GUF and the Monte Carlo results of JCGM 101:2008 (for t-distributed inputs, the sample variance would be $\nu/(\nu-2)\,u^2$ instead of u^2). The samples are independent; correlations between inputs can be supplied to `propagate` only for `output = "stat"`, *e.g.* with `cov = cor2cov(R, u^2)` using the `u` of the attribute `"budget"`.

CSV formats. The separator is detected from the header line: `";"` (with decimal comma, as written by spreadsheet programs in many European locales), tab, or `","` (with decimal point).

Type. The result carries an attribute `"type"` with the suggested `type` for `propagate` (`"stat"` for at most 3 rows, `"raw"` for up to 999 rows, `"sim"` otherwise; `"stat"`/`"sim"` for budgets), which is also reported in a message. Check it: raw data with at most three repeated observations also have at most 3 rows and are then reported as `"stat"`.

Value

A matrix with one named column per input quantity and the attribute `"type"`. For `layout = "budget"`, the attribute `"budget"` contains the interpreted budget (name, mean, `u`, `df`, distribution and half-width).

Author(s)

Andrej-Nikolai Spiess

See Also

`propagate`, `cor2cov`

Examples

```
## (a) Workspace: simulated inputs
EXPR1 <- expression(a + b^d)
a <- rnorm(10000, 12, 1)
b <- rnorm(10000, 5, 0.1)
d <- runif(10000, 6, 7)
DAT1 <- makeDat(EXPR1)

## (b) Repeated observations from a CSV file (JCGM 100:2008, H.2)
f <- tempfile(fileext = ".csv")
write.csv(data.frame(V = c(5.007, 4.994, 5.005, 4.990, 4.999),
                     I = c(19.663, 19.639, 19.640, 19.685, 19.678),
                   phi = c(1.0456, 1.0438, 1.0468, 1.0428, 1.0433)), f, row.names = FALSE)
EXPR2 <- expression((V/I) * cos(phi))
DAT2 <- makeDat(EXPR2, file = f)
RES2 <- propagate(EXPR2, DAT2, type = "raw", cov = "full", nsim = 1E5, check = FALSE)
```

```
## (c) Uncertainty budget, mass calibration of EA 4/02
## (EMUE Compendium, section 3.6)
BUD <- data.frame(name  = c("mS", "dmD", "dm", "dmC", "dB"),
                  value = c(10000.005, 0, 0.020, 0, 0),
                  ## dm: standard uncertainty of the mean of 3 observations, s = 0.025 g
                  u     = c(NA, NA, 0.025/sqrt(3), NA, NA),
                  U     = c(0.045, NA, NA, NA, NA),
                  k     = c(2, NA, NA, NA, NA),
                  a     = c(NA, 0.015, NA, 0.010, 0.010),
                  dist  = c("normal", "rect", "normal", "rect", "rect"))
EXPR3 <- expression(mS + dmD + dm + dmC + dB)
## means, u, df for type = "stat" ...
DAT3 <- makeDat(EXPR3, data = BUD)
attr(DAT3, "budget")
## ... or samples from the stated distributions for type = "sim"
DAT4 <- makeDat(EXPR3, data = BUD, output = "sim", nsim = 1E5)
RES4 <- propagate(EXPR3, DAT4, type = "sim", check = FALSE)
RES4$sim   ## u = 0.029 g, 95 % coverage interval [9999.967, 10000.082] g
```

`makeDerivs` *Utility functions for creating gradient and Hessian matrices with symbolic derivatives and evaluating them in an environment*

Description

`makeGrad` and `makeHess` create matrices of the symbolic first- and second-order partial derivatives of an expression; `evalDerivs` evaluates such a matrix in an environment. The evaluations of the symbolic derivatives are used within the `propagate` function to calculate the propagated uncertainty, but these functions may also be useful for other applications.

Usage

```
makeGrad(expr, order = NULL)
makeHess(expr, order = NULL)
evalDerivs(deriv, envir)
```

Arguments

`expr`	an expression, such as `expression(x/y)`.
`order`	order of creating partial derivatives, *e.g.* `c(2, 1)`: a permutation of the variable order of `all.vars(expr)`. See 'Examples'.
`deriv`	a matrix returned from `makeGrad` or `makeHess`.
`envir`	an environment (or a list) to evaluate in. If missing, `.GlobalEnv` is used.

Details

Given a function $f(x_1, x_2, \ldots, x_n)$, the following matrices containing symbolic derivatives of f are returned:

makeGrad:

$$\nabla(f) = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right]$$

makeHess:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Value

The symbolic or evaluated Gradient/Hessian matrices.

Author(s)

Andrej-Nikolai Spiess

References

The Matrix Cookbook (Version November 15, 2012).
Petersen KB & Pedersen MS.
Technical University of Denmark (2012).
https://www2.compute.dtu.dk/pubdb/pubs/3274-full.html

Examples

```
EXPR <- expression(a^b + sin(d))
ENVIR <- list(a = 2, b = 3, d = 4)

## First-order partial derivatives: Gradient.
GRAD <- makeGrad(EXPR)

## This will evaluate the Gradient.
evalDerivs(GRAD, ENVIR)

## Second-order partial derivatives: Hessian.
HESS <- makeHess(EXPR)

## This will evaluate the Hessian.
evalDerivs(HESS, ENVIR)
```

```
## Change derivatives order.
GRAD <- makeGrad(EXPR, order = c(2,1,3))
evalDerivs(GRAD, ENVIR)
```

---

matrixStats *Fast column- and row-wise versions of variance coded in C++*

Description

These two functions are fast C++ versions for column- and row-wise variance calculation on matrices/data.frames and are meant to substitute the classical `apply(mat, 1, var)` approach.

Usage

```
colVarsC(x)
rowVarsC(x)
```

Arguments

x	a matrix or data.frame

Details

They are coded in a way that they automatically remove `NA` values, so they behave like `na.rm = TRUE`; a column/row with fewer than two non-missing values gives `NaN`. Both functions require `x` to have at least two columns: for a single column, `colVarsC` stops with "'x' should have more than one column!" and `rowVarsC` with "'x' should have more than one column!".

Value

A (named) vector of column/row variances, with the column/row names of `x`.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Speed comparison on a large matrix:
## rowVarsC is much faster than apply().
MAT <- matrix(rnorm(10 * 100000), ncol = 10)
system.time(RES1 <- apply(MAT, 1, var))
system.time(RES2 <- rowVarsC(MAT))
all.equal(RES1, RES2)
```

`mixCov` *Aggregating covariance matrices and/or variance vectors into a single covariance matrix*

Description

This function aggregates covariance matrices, single variance values or vectors of variance values into one block-diagonal covariance matrix with dimnames, suitable for the `cov` argument of `propagate`.

Usage

```
mixCov(...)
```

Arguments

`...`	any number of covariance matrices (square, at least 2 x 2), single variance values or vectors of variance values, in the order in which the variables should appear in the result.

Details

'Mixes' (aggregates) data of the following types into a final covariance matrix:
1) covariance matrices $\mathbf{\Sigma}$ that are already available.
2) single variance values σ^2.
3) a vector of variance values $\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2$.

Each covariance matrix is used as it is; a single variance or a vector of variances becomes a diagonal matrix. If the k-th argument has m_k variables and $S_k = m_1 + \ldots + m_k$ ($S_0 = 0$), the result is the $S_K \times S_K$ matrix $\mathbf{C}$ (K = number of arguments) that is filled successively: the k-th argument fills rows and columns $S_{k-1} + 1 \ldots S_k$ of $\mathbf{C}$, and all elements outside these diagonal blocks are 0 (no covariances between the blocks).

The dimnames of the result are taken, for each argument, from the column names of a covariance matrix or the names of a variance vector. If these are missing, the (first) variable in the argument's expression is used, e.g. `"As"` for `As[2]^2`, with a number appended if the argument holds more than one variance (e.g. `"x1"`, `"x2"` for an unnamed vector `x` of length 2). If the expression contains no variable (e.g. a number), the names are `"V1"`, `"V2"` and so on. Finally, `make.unique` makes all names unique. These names matter because `propagate` matches the rows/columns of `cov` to the columns of `data` by name.

Value

The aggregated (block-diagonal) covariance matrix with dimnames.

Author(s)

Andrej-Nikolai Spiess

References

Evaluation of measurement data - Guide to the expression of uncertainty in measurement. JCGM 100:2008. Joint Committee for Guides in Metrology (2008). doi:10.59161/JCGM1002008E

Examples

```
## Names of the result: column names of a matrix,
## names of a vector, else the variable in the call.
S <- matrix(c(4, 1, 1, 9), 2, 2, dimnames = list(c("a", "b"), c("a", "b")))
u <- 0.5
mixCov(S, c(d = 0.01, e = 0.04), u^2, 2)

########################################################
## Example in Annex H.4.1 of JCGM 100:2008,
## measurement of activity.
## This will give exactly the same values as Table H.8.
data(H.4)
T0 <- 60   ## counting time in min (JCGM 100:2008, H.4)
lambda <- 1.25894E-4
Rx <- with(H.4, ((Cx - Cb)/T0) * exp(lambda * tx))
Rs <- with(H.4, ((Cs - Cb)/T0) * exp(lambda * ts))

mRx <- mean(Rx)
sRx <- sd(Rx)/sqrt(6)
mRx
sRx

mRs <- mean(Rs)
sRs <- sd(Rs)/sqrt(6)
mRs
sRs

R <- Rx/Rs
mR <- mean(R)
sR <- sd(R)/sqrt(6)
mR
sR

cor(Rx, Rs)

## Definition as in H.4.3.
As <- c(0.1368, 0.0018)
ms <- c(5.0192, 0.005)
mx <- c(5.0571, 0.001)

## We have to scale Rs/Rx by sqrt(6) to get the
## covariances of the means.
Rs <- Rs/sqrt(6)
Rx <- Rx/sqrt(6)

## Here we create an aggregated covariance matrix
```

```
## from the raw and summary data; its names are
## "Rs", "Rx", "As", "ms", "mx".
COV1 <- cov(cbind(Rs, Rx))
COV <- mixCov(COV1, As[2]^2, ms[2]^2, mx[2]^2)
COV

## Prepare the data for 'propagate'.
MEANS <- c(mRs, mRx, As[1], ms[1], mx[1])
SDS <- c(sRs, sRx, As[2], ms[2], mx[2])
DAT <- rbind(MEANS, SDS)
colnames(DAT) <- c("Rs", "Rx", "As", "ms", "mx")

## This will give exactly the same values as
## in H.4.3/H.4.3.1.
EXPR <- expression(As * (ms/mx) * (Rx/Rs))
propagate(EXPR, data = DAT, cov = COV, nsim = 10000, check = FALSE)
```

`moments` *Skewness and (excess) Kurtosis of a vector of values or of the columns of a matrix/data frame*

Description

These functions calculate skewness and excess kurtosis of a vector of values, or column-wise for a matrix or data frame. They were adapted from the package 'moments'.

Usage

```
skewness(x, na.rm = FALSE)
kurtosis(x, na.rm = FALSE)
```

Arguments

`x`	a numeric vector, matrix or data frame.
`na.rm`	logical. Should missing values be removed?

Details

The formulas use the biased $(1/n)$ sample moments, without small-sample correction, as in package 'moments' (other software may use bias-corrected versions):
Skewness:

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right)^{3/2}}$$

(excess) Kurtosis:

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right)^{2}} - 3$$

Value

The skewness/kurtosis value of a vector, or a vector with one value per column of a matrix or data frame.

Author(s)

Andrej-Nikolai Spiess

Examples

```
X <- rnorm(100, 20, 2)
skewness(X)
kurtosis(X)
```

`numDerivs` *Functions for creating Gradient and Hessian matrices by numerical differentiation (Richardson's method) of the partial derivatives*

Description

These two functions create Gradient and Hessian matrices by Richardson's central finite difference method of the partial derivatives for any expression.

Usage

```
numGrad(expr, envir = parent.frame())
numHess(expr, envir = parent.frame())
```

Arguments

`expr`	an expression, such as `expression(x/y)`.
`envir`	the `environment` in which the variables of `expr` are found (default: the calling environment, so that it also works inside functions).

Details

Calculates first- and second-order numerical approximations of the partial derivatives at the current values $x = (x_1, \ldots, x_N)$ of the variables, using the **central difference formulas**

$$f_i'(x) \approx \frac{f(x_1, \ldots, x_i + d_i, \ldots, x_N) - f(x_1, \ldots, x_i - d_i, \ldots, x_N)}{2d_i}$$

$$f_{ii}''(x) \approx \frac{f(x_1, \ldots, x_i + d_i, \ldots, x_N) - 2f(x_1, \ldots, x_N) + f(x_1, \ldots, x_i - d_i, \ldots, x_N)}{d_i^2}$$

and, for the mixed partial derivatives ($i \neq j$), with $x^{\pm} = x$ with $x_i \pm d_i$ and $x_j \pm d_j$,

$$f_{ij}''(x) \approx \frac{f(x^+) - 2f(x) + f(x^-) - f_{ii}''(x)\, d_i^2 - f_{jj}''(x)\, d_j^2}{2d_i d_j}.$$

These approximations are refined by **Richardson extrapolation**, as in genD of the 'numDeriv' package: each is computed for $r = 4$ step sizes, starting with $d_i = 0.1|x_i|$ (plus 10^{-4} if x_i is close to 0) and halving the step each time, and the r values $D_1, \ldots, D_r$ are combined successively for $m = 1, \ldots, r-1$ by $D_k \leftarrow (4^m D_{k+1} - D_k)/(4^m - 1)$.

Value

The numeric Gradient/Hessian matrices.

Note

The two functions are modified versions of the genD function in the 'numDeriv' package, but easier to handle because they use expressions: the variables are named as in the expression and need not be written as x[1], x[2], ..., x[n] in the body of a function.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Check for equality of symbolic
## and numerical derivatives.
EXPR <- expression(2^x + sin(2 * y) - cos(z))
x <- 5
y <- 10
z <- 20

symGRAD <- evalDerivs(makeGrad(EXPR))
numGRAD <- numGrad(EXPR)
all.equal(symGRAD, numGRAD)

symHESS <- evalDerivs(makeHess(EXPR))
numHESS <- numHess(EXPR)
all.equal(symHESS, numHESS)
```

---

plot.propagate *Plotting function for 'propagate' objects*

Description

Creates a histogram (blue) of the Monte Carlo output, along with a density curve (red), the $100(1-\alpha)$ % coverage intervals (GUM uncertainty framework (GUF): dashed black; Monte Carlo, probabilistically symmetric: solid black), median (orange) and mean (green) of the Monte Carlo output, with α = alpha of the propagate call.

Usage

```
## S3 method for class 'propagate'
plot(x, logx = FALSE, breaks = 100, ...)
```

Arguments

x	an object returned from propagate.
logx	logical. If TRUE, the histogram and density curve show the log-transformed Monte Carlo output $\log(y)$ on a linear abscissa (labelled "log(y)"), and the vertical lines are drawn at the logarithms of the interval limits, median and mean. Non-positive values give non-finite logarithms and are not shown.
breaks	numeric, passed to hist: a suggestion for the number of bins of the histogram.
...	other parameters to hist.

Details

To prevent a few extreme values from compressing the histogram, the Monte Carlo output is restricted to the values between its 0.1 % and 99.9 % quantiles, i.e. the 0.1 % most extreme values on each side are omitted from the histogram and density curve (not from the intervals, median and mean, which are those of the propagate results).

Value

A plot as described above.

Author(s)

Andrej-Nikolai Spiess

Examples

```
EXPR1 <- expression(x^2 * sin(y))
x <- c(5, 0.01)
y <- c(1, 0.01)
DF1 <- cbind(x, y)
RES1 <- propagate(expr = EXPR1, data = DF1, type = "stat",
                  nsim = 100000, alpha = 0.01, check = FALSE)
plot(RES1)
```

---

predictNLS *Confidence/prediction intervals for (weighted) nonlinear models based on uncertainty propagation*

Description

A function for calculating confidence/prediction intervals of (weighted) nonlinear models for the supplied or new predictor values, by using first-/second-order Taylor expansion and Monte Carlo simulation. This approach can be used to construct more realistic error estimates and confidence/prediction intervals for nonlinear models than what is possible with only a simple linearization (first-order Taylor expansion) approach. Another application is when there is an "error in x" setup with uncertainties in the predictor variable (See 'Examples'). This function will also work in the presence of multiple predictors with/without (correlated) errors.

Usage

```
predictNLS(model, newdata, newerror, interval = c("confidence", "prediction", "none"),
           alpha = 0.05, ...)
```

Arguments

model	a model obtained from nls or nlsLM (package 'minpack.lm').
newdata	a data frame with new predictor values, with one column per predictor variable of model, named as in the model formula (in any order). See predict.nls and 'Examples'. If omitted, the model's predictor values are employed.
newerror	optional uncertainties of the predictor values: a data frame of standard uncertainties with the same column names and number of rows as newdata (independent predictors), or, for correlated predictors, a covariance matrix with row and column names as in newdata (used for all rows), or a list of such matrices, one per row of newdata. See 'Examples'.
interval	a character string: "confidence" for confidence intervals of the fitted curve, "prediction" for prediction intervals of a new observation, or "none". With "none", the results are those of "confidence".
alpha	a numeric value: $1 - \alpha$ is the confidence level of the intervals (the coverage probability of the Monte Carlo coverage interval).
...	other parameters to be supplied to propagate.

Details

Taylor expansion. Calculation of the propagated variance $\sigma_y^2 = \nabla\Sigma\nabla^T$ is called the "Delta Method" and is widely applied in NLS fitting. It is based on a first-order Taylor expansion and thus assumes that f is linear in β around $\hat{\beta}$. The second-order approach of propagate partially corrects for this restriction by using a second-order polynomial.

Confidence and prediction intervals are calculated as

$$\hat{y} \pm t_{n-p}(1-\alpha/2)\,\sigma_y \quad (1), \qquad \hat{y} \pm t_{n-p}(1-\alpha/2)\sqrt{\sigma_y^2 + \sigma_r^2} \quad (2),$$

with n the number of observations, p the number of coefficients and the residual variance

$$\sigma_r^2 = \frac{\sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2}{n-p} \quad (3)$$

($w_i = 1$ for unweighted fits). The coefficients and the residual term share this variance estimate and are therefore one component with $n - p$ degrees of freedom for the effective degrees of freedom (Willink 2007, Result 1; passed to propagate as ws.group). Without predictor errors, the coverage factor is thus exactly $t_{n-p}(1-\alpha/2)$, the classical delta-method interval; uncertainties of the predictors from newerror (Gaussian, $\nu = \infty$) increase the effective degrees of freedom according to the generalized Welch-Satterthwaite formula (see WelchSatter). A df.tot passed in ... replaces these effective degrees of freedom, i.e. it only sets the coverage factor of the Taylor expansion interval; it does not change the Monte Carlo simulation.

Augmented model for prediction intervals. The new observation is $y = f(x, \beta) + \varepsilon$ (4), with a residual term ε of expectation 0 and variance σ_r^2. It enters as an additional variable rv of the

expression, and the covariance matrix $\mathbf{\Sigma}$ of the p coefficients is augmented to a $(p+1) \times (p+1)$ matrix with $\mathbf{\Sigma}_{p+1,p+1} = \sigma_r^2$. For two coefficients with variances σ_1^2, σ_2^2 and covariance σ_{12}:

$$\left[\frac{\partial f}{\partial \beta_1} \; \frac{\partial f}{\partial \beta_2} \; 1\right] \begin{bmatrix} \sigma_1^2 & \sigma_{12} & 0 \\ \sigma_{12} & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_r^2 \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial \beta_1} \\ \frac{\partial f}{\partial \beta_2} \\ 1 \end{bmatrix} = \left(\frac{\partial f}{\partial \beta_1}\right)^2 \sigma_1^2 + 2\frac{\partial f}{\partial \beta_1}\frac{\partial f}{\partial \beta_2}\sigma_{12} + \left(\frac{\partial f}{\partial \beta_2}\right)^2 \sigma_2^2 + \sigma_r^2 \equiv \sigma_y^2 + \sigma_r^2,$$

which then goes into (2). The prediction interval assumes that the new observation has unit weight.

Monte Carlo simulation. The coefficients $\hat{\beta}$ and the residual term ε all depend on the same estimated residual variance σ_r^2. Their joint distribution is therefore a *multivariate* t-distribution with $\nu = n - p$ degrees of freedom and one common χ_ν^2 variable, not a set of independent or copula-joined t-distributions:

$$(\beta, \varepsilon) \sim t_{n-p}\left((\hat{\beta}, 0), \begin{bmatrix} \mathbf{\Sigma} & 0 \\ 0 & \sigma_r^2 \end{bmatrix}\right).$$

This is the posterior distribution for a model that is linear in the coefficients, with Gaussian errors and a non-informative prior, and it has the same structure as the multivariate t-distribution of JCGM 102:2011, 5.3.2. `predictNLS` samples it with `propagate`(..., `type = "stat"`, `mc.margins = "mvt"`), with $\nu = n - p$ in the 3rd row of the input data. The uncertainties of the predictor values from `newerror` are Type B uncertainties independent of the fit (degrees of freedom `Inf`); the predictor values are sampled from a (multivariate) Gaussian distribution with the covariance matrix given in `newerror`, or, if a data frame of standard uncertainties is given, from independent Gaussian distributions.

For a model that is linear in the coefficients and without predictor errors, the Taylor expansion reproduces the classical intervals of `predict.lm` exactly, and the Monte Carlo simulation within the Monte Carlo error: $\hat{y} \pm t_{n-p}(1-\alpha/2)\,\sigma_y$ for confidence and $\hat{y} \pm t_{n-p}(1-\alpha/2)\sqrt{\sigma_y^2 + \sigma_r^2}$ for prediction intervals (see 'Examples'). For nonlinear models, the Monte Carlo intervals additionally account for the nonlinearity of f.

Predictor errors (`newerror`). `newerror` can be given in three forms:
1) a data frame (or a matrix without matching row names) of *standard uncertainties*, with the predictor names as column names and one row per row of `newdata`: the predictor errors are independent.
2) a *covariance matrix* (variances on the diagonal, covariances off the diagonal, not standard uncertainties) whose row and column names are the predictor names, in any order: it is used for all rows of `newdata`.
3) a *list* of such covariance matrices, one per row of `newdata`.
In forms 2) and 3), the matrices must be symmetric. The predictor covariance matrix is added as a separate block to the covariance matrix of the coefficients (see `mixCov`), i.e. the predictor errors are uncorrelated with the coefficients.

Value

A list with the following items:
`summary`: a data frame with one row per predictor value, containing the `prop` and `sim` results of `propagate`: the mean and uncertainty estimates from first-/second-order Taylor expansion with the t_{n-p}-based interval (columns 1-6), and from Monte Carlo simulation with the probabilistically symmetric Monte Carlo coverage interval (columns 7-12).
`prop`: the complete output from `propagate` for each row of `newdata`.

Author(s)

Andrej-Nikolai Spiess

References

Nonlinear Regression.
Seber GAF & Wild CJ.
John Wiley & Sons, New York (1989).

Nonlinear Regression Analysis and Its Applications.
Bates DM & Watts DG.
John Wiley & Sons, New York (1988).

Statistical Error Propagation.
Tellinghuisen J.
J. Phys. Chem. A (2001), **105**: 3917-3921.

Least-squares analysis of data with uncertainty in x and y: A Monte Carlo methods comparison.
Tellinghuisen J.
Chemometrics and Intelligent Laboratory Systems (2010), **103**: 160-169.

A generalization of the Welch-Satterthwaite formula for use with correlated uncertainty components.
Willink R.
Metrologia (2007), **44**: 340-349.

Evaluation of measurement data - Supplement 2 to the "Guide to the expression of uncertainty in measurement" - Extension to any number of output quantities.
JCGM 102:2011. Joint Committee for Guides in Metrology (2011).
doi:10.59161/JCGM1022011

Examples

```
## In these examples, 'nsim' is 1E4 or 1E5 to save
## Rcmd check time (CRAN). It is advocated
## to use at least 'nsim = 1E6' though...

## Example from ?nls.
DNase1 <- subset(DNase, Run == 1)
fm3DNase1 <- nls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
                 data = DNase1, start = list(Asym = 3, xmid = 0, scal = 1))

## Using a single predictor value without error.
PROP1 <- predictNLS(fm3DNase1, newdata = data.frame(conc = 2), nsim = 10000)
PRED1 <- predict(fm3DNase1, newdata = data.frame(conc = 2))
PROP1$summary
PRED1
## => mean.1 equal to PRED1

## Using a single predictor value with error.
PROP2 <- predictNLS(fm3DNase1, newdata = data.frame(conc = 2),
                    newerror = data.frame(conc = 0.5), nsim = 10000)
PROP2$summary
```

```
## Two predictors with correlated errors (r = 0.9)
set.seed(1); x <- runif(30, 1, 10); z <- runif(30, 0, 2)
y <- 5 * exp(-0.3 * x) + 2 * z^1.5 + rnorm(30, 0, 0.05)
fm2 <- nls(y ~ a * exp(-b * x) + g * z^d, start = list(a = 4, b = 0.2, g = 1, d = 1))
S <- matrix(c(0.2^2, 0.9 * 0.2 * 0.05, 0.9 * 0.2 * 0.05, 0.05^2), 2,
            dimnames = list(c("x", "z"), c("x", "z")))
## 'S' is a covariance matrix (variances, not standard uncertainties)
## whose dimnames are the predictor names.
predictNLS(fm2, newdata = data.frame(x = 5, z = 1), newerror = S, nsim = 1E4)$summary
## A list of covariance matrices, one per row of 'newdata'.
predictNLS(fm2, newdata = data.frame(x = c(5, 6), z = c(1, 1)),
           newerror = list(S, 2 * S), nsim = 1E4)$summary


## Using a sequence of predictor values without error.
CONC <- seq(1, 12, by = 1)
PROP3 <- predictNLS(fm3DNase1, newdata = data.frame(conc = CONC), nsim = 10000)
PRED3 <- predict(fm3DNase1, newdata = data.frame(conc = CONC))
PROP3$summary
PRED3
## => mean.1 equal to PRED3

## Plot mean and confidence values from first-/second-order
## Taylor expansion and Monte Carlo simulation.
plot(DNase1$conc, DNase1$density)
lines(DNase1$conc, fitted(fm3DNase1), lwd = 2, col = 1)
points(CONC, PROP3$summary[, 1], col = 2, pch = 16)
lines(CONC, PROP3$summary[, 5], col = 2)
lines(CONC, PROP3$summary[, 6], col = 2)
lines(CONC, PROP3$summary[, 11], col = 4)
lines(CONC, PROP3$summary[, 12], col = 4)


## Gauge: for a model that is linear in the coefficients,
## the Taylor and Monte Carlo intervals equal those of predict.lm.
set.seed(70)
xx <- 1:15
yy <- 2 + 0.5 * xx + rnorm(15, 0, 0.4)
fitN <- nls(yy ~ a + b * xx, start = list(a = 1, b = 1))
fitL <- lm(yy ~ xx)
PN <- predictNLS(fitN, newdata = data.frame(xx = 7.3), interval = "prediction", nsim = 1E5)
PN$summary[, c(5, 6, 11, 12)]
predict(fitL, newdata = data.frame(xx = 7.3), interval = "prediction")
## => Taylor interval (columns 5, 6) equal to lwr/upr (4.806, 6.373),
##    Monte Carlo interval (columns 11, 12) within the Monte Carlo error
```

`propagate` *Propagation of uncertainty using higher-order Taylor expansion and Monte Carlo simulation*

Description

A general function for the calculation of uncertainty propagation by first-/second-order Taylor expansion and Monte Carlo (MC) simulation including covariances. Input data can be any symbolic/numeric differentiable expression and data based on summaries (Mean/SEM/DOF), experimental measurements or sampled from distributions. Uncertainty propagation is based completely on matrix calculus accounting for full covariance structure. Monte Carlo simulation is conducted using either a Gaussian copula with marginal *t*-distributions, each with its own degrees of freedom, so that correlated input quantities with different degrees of freedom can be propagated jointly (scaled and shifted according to JCGM 101:2008, 6.4.9), or the multivariate *t*-distribution of JCGM 102:2011 (see `mc.margins`). Coverage intervals are calculated from the expanded uncertainties by `WelchSatter`'s effective degrees of freedom or from the order statistics of the MC evaluations. The demos JCGM100, JCGM101, JCGM102, JCGM106, EMUE, NIST1900 and QUAM reproduce published examples of these documents, and COPULA checks the Gaussian copula against published and exact results; see `doGauge`.

Usage

```
propagate(expr, data, type = c("stat", "raw", "sim"), cov = c("diag", "full"),
          df.tot = NULL, k = NULL, nsim = 1000000, alpha = 0.05,
         exp.uc = c("1st", "2nd"), mc.margins = c("S1", "mvt"), ws.group = NULL,
          adaptive = FALSE, ndig = 2, max.nsim = 1E7,
          ad.interval = c("symmetric", "shortest"), check = TRUE, ...)
```

Arguments

`expr`	an expression, such as `expression(x/y)`.
`data`	a dataframe or matrix containing either a) the means (1^{st} row), standard uncertainties (2^{nd} row; SEMs for Type A) and optional degrees of freedom (3^{rd} row), b) raw measurement values or c) sampled distribution data generated from any of R's `distributions` or those implemented in this package (`rDistr`).
`type`	type of input data. Either `stat` for summary data, `raw` for experimental measurements or `sim` for distribution data. See 'Details'.
`cov`	use only `"diag"`onal variances, the `"full"` covariance matrix with off-diagonals or an external covariance matrix with the same column names as `data`. Like `data`, an external matrix is reduced to the variables of `expr`. See 'Details'.
`df.tot`	an optional total (effective) degrees of freedom ν_{tot} of the output that overrides the one obtained from `WelchSatter`. Only affects the coverage factor k, not the Monte Carlo sampling. See 'Details'.
`k`	an optional coverage factor that overrides the one obtained from `WelchSatter`.
`nsim`	the number of Monte Carlo trials M, minimum is 10000.
`alpha`	1 - coverage probability p of the coverage intervals.
`exp.uc`	Should the expanded uncertainty and coverage interval be calculated from the `"1st"`-order or `"2nd"`-order uncertainty?
`mc.margins`	the probability distribution assigned to the input quantities for Monte Carlo simulation with `type = "stat"` or `"raw"`: `"S1"` (default) for scaled and shifted *t*-distributions according to JCGM 101:2008, 6.4.9, each with the degrees of

freedom of its own input quantity, joined by a Gaussian copula with the input correlation matrix; `"mvt"` for the multivariate t-distribution of JCGM 102:2011, 5.3.2, for jointly observed quantities (`type = "raw"`, `cov = "full"`), or for summary data whose finite degrees of freedom are equal and share one variance estimate (`type = "stat"`, as in `predictNLS`). Ignored for `type = "sim"`. The variance-matched margins of versions up to 1.2-0 (`"GUF"`) were removed in 1.3-0; see 'Details'.

`ws.group` an optional vector that labels input quantities estimated jointly from the same n observations, passed to `WelchSatter` as `group` (Willink 2007, Result 1). For `type = "raw"` with `cov = "full"` and complete data, all inputs are grouped automatically.

`adaptive` `FALSE` (default), `TRUE` or `"validation"`: adaptive Monte Carlo procedure of JCGM 101:2008, 7.9.4, with the numerical tolerance δ or, for `"validation"`, $\delta/5$ as recommended for the validation of the GUM uncertainty framework (8.2). `nsim` is then ignored. See 'Details'.

`ndig` the number of significant decimal digits of $u(y)$ that defines the numerical tolerance δ of the adaptive procedure (7.9.2).

`max.nsim` the maximum number of Monte Carlo trials of the adaptive procedure.

`ad.interval` the coverage interval whose endpoints enter the stopping rule of the adaptive procedure: probabilistically symmetric (default) or shortest.

`check` logical. If `TRUE`, the Monte Carlo margins are fitted with a scaled/shifted t-distribution to check for proper construction, see `summary.propagate`.

`...` other parameters to be supplied to future methods.

Details

The different `type` definitions are the following:

1) "stat", GUM 4.2: Statistical summary data (Mean, SEM, DOF), Type A evaluation

```
     V1            V2
|------------|------------|
|   Mean_1   |   Mean_2   |  => Means in 1st row
|------------|------------|
|   SEM_1    |   SEM_2    |  => SEMs (uncertainties) in 2nd row
|------------|------------|
|  (DOF_1)   |  (DOF_2)   |  => optional DOFs in 3rd row
|------------|------------|
```

Covariance matrix: from SEMs^2 as diagonals (`cov = "diag"`) or supplied externally.

2) "raw", GUM 4.2: Raw measurement values in rows, Type A evaluation

```
      V1           V2
|------------|------------| --
|    x_1     |    y_1     |   \
```

```
|------------|------------|    \
|    x_2     |    y_2     |     |
|------------|------------|     |- Single raw values in rows 1...n
|    x_3     |    y_3     |     |
|------------|------------|    /
|    x_n     |    y_n     |   /
|------------|------------| --
```

Mean, SEM and DOF are estimated from the data.
Covariance matrix: from estimated SEMs^2 as diagonals (`cov = "diag"`), full covariance matrix $\mathrm{cov}(\mathrm{data})/n$ (`cov = "full"`) or supplied externally.

3) **"sim", GUM 4.3: Random samples from distributions $x_i, y_i, ... \sim F_1, F_2, ...$, Type B evaluation**

```
      F1           F2
|------------|------------| --
|     x_1    |    y_1     |   \
|------------|------------|    \
|     x_2    |    y_2     |     |
|------------|------------|     |- MC samples in rows 1...M
|     x_3    |    y_3     |     |
|------------|------------|    /
|     x_M    |    y_M     |   /
|------------|------------| --
```

Mean, SD and DOF are estimated from the data. The latter will be high (*M* - 1) and approximate a normal distribution.
Covariance matrix: from estimated SDs^2 as diagonals (`cov = "diag"`), full covariance matrix $\mathrm{cov}(\mathrm{data})$ (`cov = "full"`) or supplied externally.

For all `types`: Expanded uncertainty U and coverage intervals from $U = k \cdot u, k = t_{\mathrm{ws.df}}(1 - \alpha/2)$, CI $= [\hat{y} - U, \hat{y} + U]$, and empirical $[\alpha/2, 1 - \alpha/2]$ quantiles of the MC output distribution.

`propagate` tries "hard" to guess the data `type` and gives a comment (no change in settings!), *i.e.* **Type A summary** => 2/3 rows, **Type A raw measurement** => 3 < rows < 1000, **Type B distributions** => rows >= 1000.

The implemented methods are:
1) **Uncertainty propagation:**
The propagated uncertainties are calculated by first-/second-order Taylor expansion of f around the input means $\bar{\mathbf{x}}$, accounting for the full covariance structure by matrix algebra. With N input quantities, ∇ is the $1 \times N$ gradient with elements $j_i = \partial f/\partial x_i$, $\mathbf{H}$ the symmetric $N \times N$ Hessian with elements $h_{ik} = \partial^2 f/\partial x_i \partial x_k$, and $\mathbf{\Sigma}$ the $N \times N$ covariance matrix with variances σ_i^2 and covariances $\sigma_{ik} = \rho_{ik}\sigma_i\sigma_k$. All derivatives are evaluated at $\bar{\mathbf{x}}$. For two inputs x_1, x_2, the matrix expressions reduce to the classical notation as follows:

$$\nabla = \begin{bmatrix} j_1 & j_2 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{bmatrix}, \quad \mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$$

First-order mean:

$$\mathrm{E}[y] = f(\bar{\mathbf{x}})$$

First-order variance:

$$\sigma_y^2 = \nabla\,\mathbf{\Sigma}\,\nabla^T = {j_1}^2\sigma_1^2 + 2\,j_1 j_2\,\sigma_{12} + {j_2}^2\sigma_2^2 = \underbrace{\sum_{i=1}^{N} {j_i}^2\sigma_i^2 + 2\sum_{i=1}^{N-1}\sum_{k=i+1}^{N} j_i j_k\,\sigma_{ik}}_{\text{classical notation}}$$

This is the law of propagation of uncertainty, JCGM 100:2008, Eq. (13); for uncorrelated inputs it reduces to $\sum_i j_i^2\sigma_i^2$ (Eq. (10)). Only for fully correlated inputs ($\rho_{ik} = +1$ for all pairs) does it become $\sigma_y^2 = \left(\sum_i j_i\sigma_i\right)^2$, the linear sum of uncertainty contributions (JCGM 100:2008, 5.2.2, Note 1).

Second-order mean:

$$\mathrm{E}[y] = f(\bar{\mathbf{x}}) + \frac{1}{2}\,\mathrm{tr}(\mathbf{H\Sigma}) = f(\bar{\mathbf{x}}) + \frac{1}{2}\left(h_{11}\sigma_1^2 + 2\,h_{12}\,\sigma_{12} + h_{22}\sigma_2^2\right) = f(\bar{\mathbf{x}}) + \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} h_{ik}\,\sigma_{ik}$$

with $\sigma_{ii} = \sigma_i^2$. The term $\frac{1}{2}\mathrm{tr}(\mathbf{H\Sigma})$ corrects the bias of $f(\bar{\mathbf{x}})$ for nonlinear f, as the first-order expansion assumes linearity around $\bar{\mathbf{x}}$.

Second-order variance:

$$\sigma_y^2 = \nabla\,\mathbf{\Sigma}\,\nabla^T + \frac{1}{2}\,\mathrm{tr}(\mathbf{H\Sigma H\Sigma})$$

For two inputs, with $\mathbf{A} = \mathbf{H\Sigma}$ and $\mathrm{tr}(\mathbf{A}^2) = a_{11}^2 + 2a_{12}a_{21} + a_{22}^2$:

$$\frac{1}{2}\,\mathrm{tr}(\mathbf{H\Sigma H\Sigma}) = \frac{1}{2}\left(h_{11}\sigma_1^2 + h_{12}\sigma_{12}\right)^2 + \left(h_{11}\sigma_{12} + h_{12}\sigma_2^2\right)\left(h_{12}\sigma_1^2 + h_{22}\sigma_{12}\right) + \frac{1}{2}\left(h_{12}\sigma_{12} + h_{22}\sigma_2^2\right)^2$$

which, for uncorrelated inputs ($\sigma_{12} = 0$), reduces to

$$\frac{1}{2}\,\mathrm{tr}(\mathbf{H\Sigma H\Sigma}) = \frac{1}{2}{h_{11}}^2\sigma_1^4 + {h_{12}}^2\sigma_1^2\sigma_2^2 + \frac{1}{2}{h_{22}}^2\sigma_2^4 = \sum_{i=1}^{N}\sum_{k=1}^{N}\frac{1}{2}\,{h_{ik}}^2\,\sigma_i^2\sigma_k^2$$

In general, $\mathrm{tr}(\mathbf{H\Sigma H\Sigma}) \neq [\mathrm{tr}(\mathbf{H\Sigma})]^2$; equality holds, *e.g.*, if $\mathbf{\Sigma}$ has rank one (all $|\rho_{ik}| = 1$) or if $\mathbf{H}$ has rank one. Likewise, the operator notation $\frac{1}{n!}\left(\sum_i \sigma_i \frac{\partial}{\partial x_i}\right)^n f$ is valid only for fully correlated inputs and is not used here.

Assumptions: The second-order mean and variance are the exact moments of the second-order Taylor polynomial of f if the inputs are Gaussian (zero third central moments, fourth central moments $3\sigma_i^4$), see Wang & Iyer (2005). For non-Gaussian inputs (*e.g.* rectangular or *t*-distributed), `u.2` is an approximation; the moments extension below (`u.2m`) removes this restriction. The Note to JCGM 100:2008, 5.1.2, additionally contains terms with third derivatives $\frac{\partial f}{\partial x_i}\frac{\partial^3 f}{\partial x_i \partial x_k^2}$, which are not part of the second-order Taylor polynomial. They vanish for models that are linear in each input separately, such as the end gauge model of JCGM 100:2008, H.1 (see 'Examples'), but not in general. The Taylor expansion uses $\mathbf{\Sigma}$ as given, *i.e.* the standard uncertainties u_i of the GUM uncertainty framework, independent of `mc.margins`.

Moments extension: second-order variance with the input moments (`u.2m`). `u.2` uses Gaussian input moments (third central moment 0, fourth $3u_i^4$). Since version 1.3-0, `propagate` also returns

`u.2m`, the second-order variance with the actual skewness $\gamma_{1,i}$ and excess kurtosis $\gamma_{2,i}$ of each input (Cox 1979; Mekid & Vaja 2008). For independent inputs, the variance of the second-order Taylor polynomial is

$$u_{2m}^2(y) = u_2^2(y) + \sum_i \left(\frac{1}{4} h_{ii}^2 \gamma_{2,i} u_i^4 + c_i h_{ii} \gamma_{1,i} u_i^3 \right),$$

i.e. the main second-order term of each input becomes $\frac{1}{4}h_{ii}^2(\gamma_{2,i}+2)u_i^4 + c_i h_{ii}\gamma_{1,i}u_i^3$ instead of $\frac{1}{2}h_{ii}^2u_i^4$; the interaction terms $h_{ik}^2u_i^2u_k^2$ are unchanged. γ_1 and γ_2 are taken from the stated distributions of a budget from `makeDat` or `corDist` (skewness and excess kurtosis: normal 0 and 0, rectangular 0 and -1.2, triangular 0 and -0.6, arcsine 0 and -1.5, t_ν 0 and $6/(\nu-4)$), from the sample moments for other samples with `type = "sim"`, and from the t_ν-distribution for `type = "raw"` and `"stat"` with finite degrees of freedom (Gaussian for $\nu = \infty$; for $\nu \le 4$ the kurtosis is infinite and the Gaussian value is used). They are returned in the component `moments`. The second-order mean does not depend on these moments, so there is no `mean.2m`.

`u.2m` is exact for quadratic models with independent inputs; for correlated pairs, the Gaussian joint moments are kept. For models with strong higher-order terms, `u.2m` need not be closer to the Monte Carlo value than `u.2`; the difference `u.2m - u.MC` then measures these terms. With `mc.margins = "S1"`, `u.MC` of Type A inputs also contains the factor $\nu/(\nu-2)$ of the t-distribution (see below). `summary.propagate` prints `u.1`, `u.2`, `u.2m` and `u.MC` side by side.

For setups in which no symbolic derivative is available (*e.g.* `e <- expression(abs(x))` => "Function 'abs' is not in the derivatives table", or `atan2`) the function automatically switches from symbolic (using `makeGrad` or `makeHess`) to numeric (`numGrad` or `numHess`) differentiation and gives a message.

The function will try to evaluate the expression in an environment using `eval` which results in a significant speed enhancement (~ 10-fold). If that fails, evaluation is done over the rows of the simulated data using `apply`.

2) **Monte Carlo simulation:**

For `type = "stat"` and `type = "raw"`, `nsim` input vectors are drawn from a joint distribution constructed from the means $\bar{x}_i$, standard uncertainties u_i (2nd row of `data` or SEMs estimated from raw data), degrees of freedom ν_i and the covariance matrix $\mathbf{\Sigma}$.

For `type = "sim"`, the supplied samples are used directly and `mc.margins` has no effect. The GUM uncertainty framework uses the sample means and standard deviations, unless the samples carry the attribute `"budget"` from `makeDat`; then the estimates, standard uncertainties and degrees of freedom of that budget are used, as in JCGM 101:2008 (a message is given). The expression is evaluated for each input vector. The mean and standard deviation of the M = `nsim` results give `mean.MC` and `u.MC`. Coverage intervals for coverage probability $p = 1-\alpha$ are formed from the order statistics $y_{(1)} \le \ldots \le y_{(M)}$ exactly as in JCGM 101:2008, 7.7: with $q = pM$ if pM is an integer, and the integer part of $pM+1/2$ otherwise, $[y_{(r)}, y_{(r+q)}]$ is a $100p\%$ coverage interval. The probabilistically symmetric interval (in `sim`, 7.7.1) takes $r = (M-q)/2$ if this is an integer, and the integer part of $(M-q+1)/2$ otherwise. The shortest interval (in `ci.short`, 7.7.2) takes the r^* that minimises $y_{(r+q)} - y_{(r)}$; it is preferable for asymmetric output distributions, but its coverage probability is marginally below $100p\%$ for finite M (7.7.2, Note).

Adaptive Monte Carlo procedure. With `adaptive = TRUE`, blocks of $M = \max(\lceil 100/\alpha \rceil, 10^4)$ trials are generated until twice the standard deviations associated with the averages of the h block values of y, $u(y)$ and of both coverage interval endpoints (`ad.interval`) are not larger than the numerical tolerance $\delta = \frac{1}{2}10^l$, with $u(y) = c \times 10^l$ and c having `ndig` digits (JCGM 101:2008, 7.9.2 and 7.9.4). All hM trials are then used for the results. `adaptive = "validation"` replaces

δ by $\delta/5$ (8.2), which needs about 25 times more trials; with this setting, the adaptive procedure reproduces the numbers of trials reported in JCGM 101:2008, 9.2.2, 9.3 and 9.5. For `type = "sim"`, the adaptive procedure needs samples from `makeDat(..., output = "sim")`, whose uncertainty budget is used to draw the further blocks.

The two *t*-distribution models (`mc.margins`):
Both models describe the state of knowledge about an input quantity from a Type A evaluation, *i.e.* from n indications with sample mean $\bar{x}$ and standard uncertainty $u = s/\sqrt{n}$ with $\nu = n-1$ degrees of freedom (JCGM 100:2008, 4.2 and G.3). They differ in how the *t*-distribution is scaled and in how several inputs are joined.

`mc.margins`	**Marginal of** X_i	**Var**(X_i)	**Reference**
`"S1"` (default)	$t_{\nu_i}(\bar{x}_i, u_i^2)$	$\frac{\nu_i}{\nu_i-2}u_i^2$	JCGM 101:2008, 6.4.9
`"mvt"`	$t_\nu\left(\bar{x}_i, \frac{n-1}{n-N}u_i^2\right), \nu = n-N$	$\frac{n-1}{n-N-2}u_i^2$	JCGM 102:2011, 5.3.2

Correlated inputs with individual degrees of freedom (`"S1"`).
A common situation in practice is a set of *correlated* input quantities whose standard uncertainties are based on *different* numbers of observations, *e.g.* two quantities evaluated from 4 and 20 indications with a known correlation. The JCGM documents do not provide a joint input distribution for this case:

- the GUM uncertainty framework uses the degrees of freedom only for the output quantity, through the Welch-Satterthwaite formula, which assumes independent inputs (JCGM 100:2008, G.4.2);
- JCGM 101:2008 assigns a *t*-distribution with its own ν_i to each input (6.4.9), but independently, without a joint distribution for correlated inputs;
- the multivariate *t*-distribution of JCGM 102:2011, 5.3.2, allows correlation, but forces one common $\nu = n - N$ on all inputs, because they are assumed to be observed jointly.

`propagate` closes this gap with a Gaussian copula (Sklar's theorem; Possolo 2010): the dependence structure is that of a multivariate Gaussian with the input correlation matrix $\mathbf{R}$, and each margin is the scaled and shifted *t*-distribution $t_{\nu_i}(\bar{x}_i, u_i^2)$ with **its own** ν_i. Inputs from few observations thus keep their heavy tails, inputs from many observations stay close to Gaussian, and the correlation is preserved, all in one joint distribution. Because the margins are obtained by strictly increasing transformations of the Gaussian copula, the rank correlations equal those of the Gaussian copula exactly (Spearman $\rho_S = \frac{6}{\pi}\arcsin\frac{\rho}{2}$) for any combination of ν_i; the Pearson correlation is somewhat smaller for heavy-tailed margins (see 'Checks'). For uncorrelated inputs, the construction reduces exactly to the independent *t*-distributions of JCGM 101:2008. The individual ν_i used are returned in `df.MC` and are verified by fitting each margin in `summary.propagate` (`check = TRUE`). The margins and correlations can be checked against published examples with `doGauge` or `demo("COPULA", package = "propagate")`.

a) `mc.margins = "S1"`*: scaled and shifted t-distributions as margins (JCGM 101:2008, 6.4.9), joined by a Gaussian copula (Possolo 2010; the copula is not part of JCGM 101).*
Assigning a non-informative prior to the unknown mean and variance of Gaussian indications, Bayes' theorem gives the scaled and shifted *t*-distribution $t_\nu(\bar{x}, s^2/n)$ with $\nu = n-1$ for the input quantity (6.4.9.2). Its scale parameter is the standard uncertainty $u = s/\sqrt{n}$ itself, so that its variance $\frac{\nu}{\nu-2}u^2$ is larger than u^2 and exists only for $\nu > 2$, *i.e.* $n > 3$ (6.4.9.4). A draw is made as $\xi = \bar{x} + u \cdot t$ with t from the standard *t*-distribution t_ν (6.4.9.5). The same distribution is assigned

when a calibration certificate states a best estimate x, an expanded uncertainty U_p, a coverage factor k_p and an effective degrees of freedom ν_{eff}: $t_{\nu_{\text{eff}}}(x, (U_p/k_p)^2)$ (6.4.9.7). For `type = "stat"`, enter x, U_p/k_p and ν_{eff} as the three rows of `data`. For $\nu \to \infty$ the distribution becomes the Gaussian of 6.4.7. For a linear model $Y = X$, the MC coverage interval $\bar{x} \pm t_\nu(1-\alpha/2)\, u$ then coincides with the GUM coverage interval, while `u.MC` exceeds `u.1` by a factor of $\sqrt{\nu/(\nu-2)}$.

Removed in 1.3-0: variance-matched t-distributions (`mc.margins = "GUF"`*, the only behaviour up to version 1.2-0).*
Each margin was the *t*-distribution of a), divided by $\sqrt{\nu_i/(\nu_i-2)}$, so that $\mathrm{Var}(X_i) = u_i^2$. This rescaling removes the additional dispersion that the *t*-distribution assigns to an input estimated from few indications, which the GUM uncertainty framework accounts for by the coverage factor t_ν and JCGM 101:2008 by the scale of the *t*-distribution. The Monte Carlo coverage interval was therefore about $\pm 2u$ for all ν (for a single input from $n = 5$ indications $\pm 1.96u$ instead of $\pm 2.78u$), with an actual coverage probability of about 88 % instead of 95 % for $n = 5$ and 92 % for $n = 10$. Monte Carlo coverage intervals of versions up to 1.2-0 for Type A inputs with few indications should therefore be recomputed; the results of the GUM uncertainty framework (`prop`) are not affected.

b) `mc.margins = "mvt"`*: multivariate t-distribution (JCGM 102:2011, 5.3.2).*
If N input quantities are observed *jointly* n times, with $n > N$, assigning a non-informative prior to their unknown expectation and covariance matrix gives the multivariate *t*-distribution $t_\nu(\bar{\mathbf{x}}, \mathbf{S}/n)$ with $\nu = n - N$ and $\mathbf{S} = \frac{1}{\nu}\sum_{r=1}^{n}(\mathbf{x}_r - \bar{\mathbf{x}})(\mathbf{x}_r - \bar{\mathbf{x}})^T$ (5.3.2.1). Note the divisor $\nu = n - N$ instead of $n - 1$. The expectation $\bar{\mathbf{x}}$ exists for $\nu > 1$ and the covariance matrix $\frac{\nu}{\nu-2}\,\mathbf{S}/n$ for $\nu > 2$ (5.3.2.3). A draw is made as $\boldsymbol{\xi} = \bar{\mathbf{x}} + \mathbf{L}\mathbf{z}\sqrt{\nu/w}$, with $\mathbf{z}$ a vector of N standard Gaussian draws, a single draw w from χ^2_ν and $\mathbf{L}$ the lower Cholesky factor of $\mathbf{S}/n = \mathbf{L}\mathbf{L}^T$ (5.3.2.4). In terms of the covariance matrix of the means used for the Taylor expansion, $\boldsymbol{\Sigma} = \mathrm{cov}(\mathrm{data})/n$, the scale matrix is $\mathbf{S}/n = \frac{n-1}{n-N}\,\boldsymbol{\Sigma}$. For $N = 1$ this reduces exactly to a).
All inputs share one ν, which decreases with the number of jointly observed quantities, and all inputs are coupled through the common variable w: they are dependent even when uncorrelated, and extreme values occur jointly (tail dependence), unlike the Gaussian copula of a). For `type = "raw"`, it requires `cov = "full"` and complete joint observations (no `NA`s). For `type = "stat"`, all inputs with finite degrees of freedom (3rd row of `data`) must have the same ν and share one χ^2_ν variable, with the supplied covariance matrix as scale matrix; inputs with degrees of freedom `Inf` are sampled from independent Gaussian distributions and must be uncorrelated with the others. This is the joint distribution of regression coefficients that share one residual variance estimate (see `predictNLS`). As noted in 5.3.2.1 of JCGM 102:2011, a different prior would change the degrees of freedom or even the distribution; a) and b) are therefore both defensible models for correlated Type A inputs, which can give different results for nonlinear models.

Sampling procedure for `mc.margins = "S1"`**:**

1. Convert the covariance matrix $\boldsymbol{\Sigma}$ to the correlation matrix $\mathbf{R}$: $\mathbf{R}_{ij} = \boldsymbol{\Sigma}_{ij}/\sqrt{\boldsymbol{\Sigma}_{ii}\boldsymbol{\Sigma}_{jj}}$.
2. Generate $\mathbf{v} \sim C_{\mathbf{R}}^{\text{Gauss}}$ from the Gaussian copula (uniform marginals on [0,1]). According to Sklar's theorem, $F(x_1, \ldots, x_N) = C_{\mathbf{R}}^{\text{Gauss}}(F_1(x_1), \ldots, F_N(x_N))$.
3. Transform each margin: $z_i = F^{-1}_{t,\nu_i}(v_i)$, with F_{t,ν_i} the CDF of the standard *t*-distribution with ν_i degrees of freedom.
4. Scale and shift: $X_i = \bar{x}_i + u_i \cdot z_i$, giving the MC-simulated matrix $\mathbf{X}^{\text{MC}}$.

In the univariate case, `rt(nsim, df)` replaces steps 1-3. For `"mvt"`, steps 1-4 are replaced by the draw of 5.3.2.4 given in b).

Degrees of freedom: The margins always use the per-input degrees of freedom: the 3rd row of `data` for `type = "stat"` (`Inf` if absent, *i.e.* Gaussian margins as in JCGM 101:2008, 6.4.7), $n_i - 1$ for `type = "raw"`, and, for `"mvt"`, $n - N$ with `type = "raw"` or the common ν of the 3rd row of `data` with `type = "stat"`. The degrees of freedom actually used are returned in `df.MC`. `df.tot` is an effective degrees of freedom of the *output* and only enters the coverage factor k (JCGM 100:2008, G.4); it is not used for sampling. For $\nu \leq 2$, the variance of the *t*-distribution does not exist: `"S1"` and `"mvt"` proceed with a message that `u.MC` is not meaningful. For $\nu = 1$, the expectation does not exist either, so `median.MC` and the MC coverage interval should be used.

Checks: The function conducts the following three additional checks, visible in `summary.propagate`. They compare the MC input with the covariance matrix it *should* have under the chosen model, $\boldsymbol{\Sigma}^{\text{target}}$ (returned as `covMat.target`): $\mathbf{D}\boldsymbol{\Sigma}\mathbf{D}$ with $\mathbf{D} = \operatorname{diag}\left(\sqrt{\nu_i/(\nu_i - 2)}\right)$ for `"S1"`, $\frac{n-1}{n-N-2}\boldsymbol{\Sigma}$ for `"mvt"` with `type = "raw"`, and $\frac{\nu}{\nu-2}\boldsymbol{\Sigma}$ for `"mvt"` with `type = "stat"`.

1. Relative Frobenius norm of the difference between the MC-derived covariance matrix $\hat{\boldsymbol{\Sigma}}^{\text{MC}}$ and the target covariance matrix $\boldsymbol{\Sigma}^{\text{target}}$:

$$\varepsilon_F = \frac{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\hat{\boldsymbol{\Sigma}}_{ij}^{\text{MC}} - \boldsymbol{\Sigma}_{ij}^{\text{target}}\right)^2}}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\boldsymbol{\Sigma}_{ij}^{\text{target}}\right)^2}}$$

2. Relative Frobenius norm of the difference between the MC-derived correlation matrix $\hat{\mathbf{R}}^{\text{MC}}$ and the input correlation matrix $\mathbf{R}$:

$$\varepsilon_F = \frac{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\hat{\mathbf{R}}_{ij}^{\text{MC}} - \mathbf{R}_{ij}\right)^2}}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{R}_{ij}^2}}$$

3. For each marginal component X_i, a shifted and scaled *t*-distribution is fitted independently by estimating $\hat{\mu}_i$, $\hat{\sigma}_i$ and $\hat{\nu}_i$. $\hat{\sigma}_i$ is reported on the scale of u_i and $\hat{\nu}_i$ should be compared with `df.MC`.

For `"mvt"`, both norms tend to zero with increasing `nsim`. For `"S1"`, the Pearson correlation of *t*-margins is smaller than the copula parameter when ν_i is small (*e.g.* 0.65 instead of 0.70 for $\nu = (3, 10)$, while the rank correlation is preserved exactly), so that both norms keep a small, ν-dependent floor for correlated inputs that does not vanish with `nsim`. The *t*-distribution has heavy tails for small ν, so all norms converge slowly in that case.

Value

A short printed output of the results (for a more detailed output use `summary(object)`) and a list with the following components:

`gradient`	the symbolic gradient vector ∇ of partial first-order derivatives.
`evalGrad`	the evaluated gradient vector ∇ of partial first-order derivatives, also known as "sensitivities". See `summary.propagate`.
`hessian`	the symbolic Hessian matrix $\mathbf{H}$ of partial second-order derivatives.
`evalHess`	the evaluated Hessian matrix $\mathbf{H}$ of partial second-order derivatives.

`rel.contr` the relative contributions, RC_i, see `summary.propagate`.

`covMat` the covariance matrix $\boldsymbol{\Sigma}$ used for Monte Carlo simulation and uncertainty propagation (input).

`covMat.MC` the estimated covariance matrix $\hat{\boldsymbol{\Sigma}}^{\mathrm{MC}}$ derived from the Monte Carlo simulation (output).

`covMat.target` the covariance matrix $\boldsymbol{\Sigma}^{\mathrm{target}}$ that the Monte Carlo input should have under the chosen `mc.margins` model, see 'Details'. Contains `Inf` for $\nu \leq 2$.

`corMat` the $\boldsymbol{\Sigma}$-converted correlation matrix $\mathbf{R}$ used for Monte Carlo simulation (input).

`corMat.MC` the estimated correlation matrix $\hat{\mathbf{R}}^{\mathrm{MC}}$ derived from the Monte Carlo simulation (output).

`frobCOV` the relative Frobenius norm ε_F of the target and MC-derived covariance matrices (`NA` if the target covariance does not exist).

`frobCOR` the relative Frobenius norm ε_F of the input and MC-derived correlation matrices.

`ws.df` the Welch-Satterthwaite degrees of freedom ν_{ws}, as obtained from `WelchSatter`.

`k` the coverage factor k, as calculated by $t(1-(\alpha/2), \nu_{\mathrm{ws}})$.

`u.exp` the expanded uncertainty, $U = k \cdot u$, where u is derived from the first- or second-order propagation, depending on the setting of `exp.uc`.

`resSIM` a vector containing the `nsim` values obtained from the expression evaluations $f(x_{m,i})$ of the simulated data in `datSIM`.

`datSIM` a matrix containing the `nsim` simulated multivariate values for each variable in column format.

`datSIM2` a second, independent MC simulation from the same input distribution, used as matrix $\mathbf{B}$ for Sobol' sensitivity analysis, see `sobol`. `NULL` for `type = "sim"`.

`prop` a summary vector containing first-/second-order expectations and uncertainties from propagation (`mean.1`, `mean.2`, `u.1`, `u.2` and `u.2m`, the second-order uncertainty with the input moments, see 'Details') as well as the coverage interval of the GUM uncertainty framework based on the *t*-distribution.

`sim` a summary vector containing the mean, standard deviation, median and MAD from MC evaluation, as well as the probabilistically symmetric coverage interval (JCGM 101:2008, 7.7.1).

`expr` the original expression `expr`.

`data` a matrix containing the given (`type = "stat"`) or estimated (`type = "raw/sim"`) means, uncertainties and degrees of freedom.

`alpha` the original α-level.

`nsim` the number of Monte Carlo trials M (hM for the adaptive procedure).

`check` a matrix containing the means, uncertainties (on the scale of u_i) and degrees of freedom derived from fitting the MC margins with a scaled/shifted *t*-distribution. The degrees of freedom are to be compared with `df.MC`.

`type` the original `type` definition, important for, *e.g.*, `sobol`.

`mc.margins` the `mc.margins` model used (`NA` for `type = "sim"`).

df.MC	the degrees of freedom used for the Monte Carlo sampling (NULL for type = "sim").
input	a list of the settings used by report: the input data (NULL for type = "sim"), type, cov, df.tot, k, exp.uc, ws.group, the method used by WelchSatter (ws.method) and the uncertainty budget of samples from makeDat (budget).
moments	a data frame with the skewness, excess kurtosis and their source for each input quantity, used for u.2m.
ci.short	the shortest $100(1-\alpha)\%$ coverage interval from MC evaluation (JCGM 101:2008, 7.7.2).
sim.se	the numerical standard uncertainties of mean.MC, u.MC and of the endpoints of both coverage intervals: the standard deviation of these results over h blocks divided by $\sqrt{h}$ ($h = 10$ equal parts of the M results, or the blocks of the adaptive procedure).
adaptive	present only for adaptive = TRUE or "validation": the number of blocks, the block size, the total number of trials, δ, the tolerance used, ndig, the interval type and whether the procedure converged.
call	the matched call.

Note

Why u.MC is larger than u.1 for Type A inputs. With mc.margins = "S1", an input estimated from n indications is assigned the scaled and shifted t-distribution $t_{\nu_i}(\bar{x}_i, u_i^2)$ with $\nu_i = n - 1$ (JCGM 101:2008, 6.4.9), whose variance is

$$\mathrm{Var}(X_i) = \frac{\nu_i}{\nu_i - 2}\, u_i^2.$$

For uncorrelated inputs and a model that is approximately linear over the ranges of the inputs, the Monte Carlo variance of the output is therefore

$$u_{\mathrm{MC}}^2 \approx \sum_i c_i^2\, \mathrm{Var}(X_i) = \sum_i c_i^2\, \frac{\nu_i}{\nu_i - 2}\, u_i^2,$$

with the sensitivity coefficients c_i. If all inputs have the same ν, this is $\frac{\nu}{\nu-2}\, u_1^2$, where $u_1^2 = \sum_i c_i^2 u_i^2$ is the first-order GUF variance. For the GUM example H.2 with $n = 5$ ($\nu = 4$) and cov = "diag",

$$u_{\mathrm{MC}}^2 \approx \sum_i c_i^2 \cdot 2\, u_i^2 = 2\, u_1^2, \qquad u_{\mathrm{MC}} \approx \sqrt{2}\, u_1.$$

This is not a discrepancy: the GUM uncertainty framework keeps u_1 and accounts for the few indications by the coverage factor $k = t_{\nu_{\mathrm{eff}}}$, the Monte Carlo method by the wider distribution of the inputs. Compare the coverage intervals of both methods, not the standard uncertainties. For $\nu_i \le 2$ the variance does not exist and u.MC is not meaningful.

Worked examples and their comparison with published values: see doGauge.

Author(s)

Andrej-Nikolai Spiess

References

Uncertainty propagation (in general):
An Introduction to error analysis.
Taylor JR.
University Science Books (1996), New York.

Effective degrees of freedom for correlated inputs:
A generalization of the Welch-Satterthwaite formula for use with correlated uncertainty components.
Willink R.
Metrologia (2007), **44**: 340-349.
doi:10.1088/00261394/44/5/010

GUM, GUM Supplements and conformity assessment:
Evaluation of measurement data - Guide to the expression of uncertainty in measurement.
JCGM 100:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1002008E

Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

Evaluation of measurement data - Supplement 2 to the "Guide to the expression of uncertainty in measurement" - Extension to any number of output quantities.
JCGM 102:2011. Joint Committee for Guides in Metrology (2011).
doi:10.59161/JCGM1022011

Evaluation of measurement data - The role of measurement uncertainty in conformity assessment.
JCGM 106:2012. Joint Committee for Guides in Metrology (2012).
doi:10.59161/JCGM1062012

Collections of worked examples:
Compendium of examples: Good practice in evaluating measurement uncertainty (EMUE), version 2.
van der Veen AMH, Cox MG (Eds.).
Zenodo (2021).
doi:10.5281/zenodo.5142180

Quantifying Uncertainty in Analytical Measurement (QUAM:2012.P1), 3rd edition.
Ellison SLR, Williams A (Eds.).
EURACHEM/CITAC Guide CG 4 (2012). ISBN 978-0-948926-30-3.
https://www.eurachem.org/index.php/publications/guides/quam

Simple Guide for Evaluating and Expressing the Uncertainty of NIST Measurement Results.
Possolo A.
NIST Technical Note 1900, National Institute of Standards and Technology (2015).
doi:10.6028/NIST.TN.1900

Higher-order Taylor expansion:
Tolerance analysis by computer.
Cox ND.
Journal of Quality Technology (1979), **11**: 80-87.

On higher-order corrections for propagating uncertainties.
Wang CM & Iyer HK.
Metrologia (2005), **42**: 406-410.

Propagation of uncertainty: Expressions of second and third order uncertainty with third and fourth moments.
Mekid S & Vaja D.
Measurement (2008), **41**: 600-609.

Matrix algebra for uncertainty propagation:
An Introduction to Error Propagation: Derivation, Meaning and Examples of Equation Cy = Fx-CxFx^t.
doi:10.3929/ethza010113668.

Second order nonlinear uncertainty modeling in strapdown integration using MEMS IMUs.
Zhang M, Hol JD, Slot L, Luinge H.
2011 Proceedings of the 14th International Conference on Information Fusion (FUSION) (2011).

Uncertainty propagation in non-linear measurement equations.
Mana G & Pennecchi F.
Metrologia (2007), **44**: 246-251.

A compact tensor algebra expression of the law of propagation of uncertainty.
Bouchot C, Quilantan JLC, Ochoa JCS.
Metrologia (2011), **48**: L22-L28.

Nonlinear error propagation law.
Kubacek L.
Appl Math (1996), **41**: 329-345.

Monte Carlo simulation (normal- and t-distribution):
MUSE: computational aspects of a GUM supplement 1 implementation.
Mueller M, Wolf M, Roesslein M.
Metrologia (2008), **45**: 586-594.

Copulas for uncertainty analysis.
Possolo A.
Metrologia (2010), **47**: 262-271.
doi:10.1088/00261394/47/3/017

Random variables, joint distribution functions, and copulas.
Sklar A.
Kybernetika (1973), **9**: 449-460.

Examples

```
## ATTENTION: all examples with nsim = 10000
## and check = FALSE for reduced CRAN check time.
## It is advised to use nsim = 1000000 (default)
## and check = TRUE in real-life setup.

## Example without given degrees of freedom.
EXPR1 <- expression(x/y)
x <- c(5, 0.01)
y <- c(1, 0.01)
DF1 <- cbind(x, y)
```

```
propagate(expr = EXPR1, data = DF1, type = "stat",
          nsim = 10000, check = FALSE)

## Same example with given degrees of freedom
## => third row in input 'data'.
EXPR2 <- expression(x/y)
x <- c(5, 0.01, 12)
y <- c(1, 0.01, 5)
DF2 <- cbind(x, y)
propagate(expr = EXPR2, data = DF2, type = "stat",
          nsim = 10000, check = FALSE)

## A single Type A input from n = 5 indications (nu = 4), Y = X:
## the MC coverage interval equals the GUM interval xbar +/- t_4(0.975) u,
## Var(X) = nu/(nu-2) u^2 = 2 u^2, hence u.MC = sqrt(2) * u.1
## (JCGM 101:2008, 6.4.9).
set.seed(1)
X <- c(10.1, 9.8, 10.3, 10.0, 9.9)
propagate(expression(X), cbind(X), type = "raw",
          nsim = 1E5, check = FALSE)

## JCGM 100:2008, H.1, end-gauge calibration (l - 50 mm in nm):
## u.1 = 32 nm; with the second-order terms u.2 = 34 nm (both as in H.1).
## As the Hessian has no diagonal terms (linear in each input), u.2m = u.2 here.
## The model is linear in each input separately, so the third-derivative
## terms of the Note to 5.1.2 vanish. dt has nu = 2 (message: u.MC not meaningful).
data(H.1)
propagate(expression(Ls + d - Ls * (da * theta + as * dt) - 50000000), H.1,
          type = "stat", alpha = 0.01, nsim = 1E4, check = FALSE)
```

`propagateMV` *Uncertainty propagation for several output quantities (JCGM 102:2011)*

Description

Propagation of uncertainty to several output quantities that depend on common input quantities, by the GUM uncertainty framework (law of propagation of uncertainty for the covariance matrix of the outputs) and by the Monte Carlo method on a common input sample, as described in GUM Supplement 2 (JCGM 102:2011). The function uses `propagate` internally, so that all its input types and Monte Carlo input models are available.

Usage

```
propagateMV(expr, data, type = c("stat", "raw", "sim"), cov = c("diag", "full"),
            nsim = 1000000, alpha = 0.05, mc.margins = c("S1", "mvt"), ...)

## S3 method for class 'propagateMV'
```

```
print(x, digits = 7, ...)
## S3 method for class 'propagateMV'
summary(object, ...)
## S3 method for class 'propagateMV'
plot(x, pairs = NULL, npoints = 3000, ...)
```

Arguments

expr	an expression with one element per output quantity, preferably named, *e.g.* `expression(R = (V/I) * cos(phi), X = (V/I) * sin(phi))`, or a list of expressions. Unnamed outputs are called `Y1`, `Y2`, ...
data	the input data as for `propagate`, containing all variables of all output models.
type	`"stat"`, `"raw"` or `"sim"`, as in `propagate`.
cov	`"diag"`, `"full"` or an external covariance matrix of the inputs, as in `propagate`.
nsim	the number of Monte Carlo trials.
alpha	1 - coverage probability of the coverage regions and intervals.
mc.margins	the Monte Carlo input model for Type A inputs, as in `propagate`.
x, object	an object returned by `propagateMV`.
digits	the number of significant digits printed.
pairs	a list of pairs of output indices to plot; default all pairs.
npoints	the number of Monte Carlo points plotted per panel.
...	further arguments to `propagate` (*e.g.* `df.tot`, `ws.group`) or to `plot`.

Details

Joint Monte Carlo sample. `propagate` is called once with a *carrier* model: the first output model plus $0 \cdot x$ for every input x that it does not contain. This generates the joint sample of all input quantities with the chosen input model (Gaussian copula with individual degrees of freedom, multivariate t-distribution, user samples, ...). All output models are evaluated on this same sample, so that the Monte Carlo results describe the joint distribution of the outputs (JCGM 102:2011, 7): estimates $\tilde{\mathbf{y}}$ (means), covariance matrix $\mathbf{U}_{\tilde{y}}$, correlation matrix and, for each output, the probabilistically symmetric coverage interval (JCGM 101:2008, 7.7.1). Trials with non-finite outputs are removed with a message.

GUM uncertainty framework. `propagate` is called once per output to obtain its sensitivity coefficients at the estimates (symbolic derivatives, or numerical derivatives if necessary). With the sensitivity matrix $\mathbf{C}_x$ (outputs $\times$ inputs) and the covariance matrix $\mathbf{U}_x$ of the inputs,

$$\mathbf{U}_y = \mathbf{C}_x \mathbf{U}_x \mathbf{C}_x^\top$$

(JCGM 102:2011, 6.2.1.3, Eq. 3).

Coverage regions for coverage probability $p = 1 - \alpha$ (JCGM 102:2011, 6.5 and 7.7):

- hyper-ellipsoidal: $(\boldsymbol{\eta} - \mathbf{y})^\top \mathbf{U}_y^{-1} (\boldsymbol{\eta} - \mathbf{y}) = k_p^2$. GUF: $k_p = \sqrt{\chi_p^2(m)}$ for m outputs (Gaussian); Monte Carlo: k_p^2 is the p-quantile of the squared Mahalanobis distances of the trials from the Monte Carlo mean.

- hyper-rectangular: $\mathbf{y} \pm k_q \mathbf{u}(\mathbf{y})$. GUF: $k_q = \Phi^{-1}((1+p^{1/m})/2)$ (independent Gaussian components, *e.g.* 2.24 for $m = 2$, $p = 0.95$, JCGM 102:2011, Table 2); Monte Carlo: k_q is the p-quantile of $\max_j |y_j - \tilde{y}_j|/u(\tilde{y}_j)$.

k_p is not available if the covariance matrix of the outputs is singular (linearly dependent outputs).

Plot. For each pair of outputs, the Monte Carlo sample with the bivariate $100(1-\alpha)\%$ ellipses of the GUF (dashed) and the Monte Carlo method (solid), each with $k = \sqrt{\chi^2_{1-\alpha}(2)}$ from the respective estimates and covariance matrix of the pair. The Monte Carlo ellipse therefore uses the Gaussian factor, not the Monte Carlo k_p, and is not the Monte Carlo coverage region.

The function reproduces the examples of JCGM 102:2011, clause 9; see `demo("JCGM102", package = "propagate")`.

Value

An object of class `propagateMV`, a list with

`guf`	results of the GUM uncertainty framework: estimates `y`, covariance matrix `U`, standard uncertainties `u`, correlation matrix `r`, coverage factors `kp` and `kq`, sensitivity matrix `Cx`.
`mc`	Monte Carlo results: `y`, `U`, `u`, `r`, `kp`, `kq` and the coverage intervals `ci` of the single outputs.
`resSIM`	the matrix of Monte Carlo output values (trials $\times$ outputs).
`datSIM`	the joint Monte Carlo input sample.
`covMat`	the covariance matrix $\mathbf{U}_x$ of the inputs.
`expr`, `names`, `alpha`, `nsim`, `type`, `mc.margins`, `df.MC`	the model, output names and settings.
`runs`	the `propagate` objects of the single outputs (univariate GUF results, sensitivities, expanded uncertainties).

Note

Worked examples and their comparison with published values: see `doGauge`.

Author(s)

Andrej-Nikolai Spiess

References

Evaluation of measurement data - Supplement 2 to the "Guide to the expression of uncertainty in measurement" - Extension to any number of output quantities.
JCGM 102:2011. Joint Committee for Guides in Metrology (2011).
doi:10.59161/JCGM1022011

Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

See Also

propagate, makeDat

Examples

```
## JCGM 102:2011, 9.4: resistance, reactance and impedance from six sets of
## simultaneous observations (GUM H.2 plus one set), multivariate t-distribution
## of the inputs with nu = n - N = 3 degrees of freedom.
data(S2.9.4)
EXPR <- expression(R = (V/I) * cos(phi), X = (V/I) * sin(phi), Z = V/I)
RES <- propagateMV(EXPR, S2.9.4, type = "raw", cov = "full", mc.margins = "mvt", nsim = 1E5)
RES
## JCGM 102:2011, Table 11: GUF u = 0.058, 0.241, 0.193 Ohm;
## MCM u = 0.130, 0.536, 0.429 Ohm, r(R, X) = -0.587 (M = 1e6)
plot(RES)

## JCGM 102:2011, 9.2, case 3: common rectangular effect X3,
## input samples from an uncertainty budget
BUD <- data.frame(name = c("X1", "X2", "X3"), value = 0, u = c(1, 1, NA),
                  a = c(NA, NA, 3 * sqrt(3)), dist = c("normal", "normal", "rect"))
EX2 <- expression(Y1 = X1 + X3, Y2 = X2 + X3)
DAT <- makeDat(EX2, data = BUD, output = "sim", nsim = 1E5)
RES2 <- propagateMV(EX2, DAT, type = "sim")
c(RES2$guf$kp, RES2$guf$kq)   ## GUF: 2.45, 2.24
c(RES2$mc$kp, RES2$mc$kq)     ## MCM: 2.28, 1.87 (JCGM 102:2011, Table 5)
```

---

rDistr *Creating random samples from a variety of useful distributions*

Description

These are random sample generators for 34 continuous distributions, collated here with a consistent parameterisation for Monte Carlo-based uncertainty propagation. Most of them are not part of R's Distributions; some are implemented in other specialized packages (*e.g.* rsn in package 'sn' or rtrapezoid in package 'trapezoid'), and some are simple transformations of base R generators (*e.g.* rchi, rlnorm3, rgamma3, runorm). The functions are internal (not exported) and are therefore called with propagate:::, *e.g.* propagate:::rtrap(1000).

Details

Random samples can be drawn from the following 34 distributions (with n = number of samples):
1) Skewed-normal distribution: propagate:::rsn(n, location = 0, scale = 1, shape = 0)
2) Generalized normal distribution: propagate:::rgnorm(n, alpha = 1, xi = 1, kappa = -0.1)
3) Scaled and shifted t-distribution: propagate:::rst(n, mean = 0, sd = 1, df = 10); sd is the scale u, not the standard deviation, which is $u\sqrt{\nu/(\nu-2)}$ with ν = df
4) Gumbel distribution: propagate:::rgumbel(n, location = 0, scale = 1)
5) Johnson SU distribution: propagate:::rJSU(n, xi = 0, lambda = 1, gamma = 1, delta = 1)
6) Johnson SB distribution: propagate:::rJSB(n, xi = 0, lambda = 1, gamma = 1, delta = 1)

7) 3P Weibull distribution: `propagate:::rweibull2(n, location = 0, shape = 1, scale = 1)`
8) 4P Beta distribution: `propagate:::rbeta2(n, alpha1 = 1, alpha2 = 1, a = 0, b = 1)`
9) Triangular distribution with mode b: `propagate:::rtriang(n, a = 0, b = 0.5, c = 1)`
10) Trapezoidal distribution: `propagate:::rtrap(n, a = 0, b = 1, c = 2, d = 3)`
11) Laplacian distribution: `propagate:::rlaplace(n, mean = 0, sigma = 1)`
12) Arcsine distribution: `propagate:::rarcsin(n, a = 2, b = 1)`
13) von Mises distribution: `propagate:::rmises(n, mu = 1, kappa = 3)`
14) Curvilinear Trapezoidal distribution: `propagate:::rctrap(n, a = 0, b = 1, d = 0.1)`
15) Generalized trapezoidal distribution:
`propagate:::rgtrap(n, min = 0, mode1 = 1/3, mode2 = 2/3, max = 1, n1 = 2, n3 = 2, alpha = 1)`
16) Inverse Gaussian distribution: `propagate:::rinvgauss(n, mean = 1, dispersion = 1)`
17) Generalized Extreme Value distribution: `propagate:::rgevd(n, loc = 0, scale = 1, shape = 0)`
18) Inverse Gamma distribution: `propagate:::rinvgamma(n, shape = 1, scale = 5)`
19) Rayleigh distribution: `propagate:::rrayleigh(n, mu = 1, sigma = 1)`
20) Burr distribution: `propagate:::rburr(n, k = 1)`
21) Chi distribution: `propagate:::rchi(n, nu = 5)`
22) Inverse Chi-Square distribution: `propagate:::rinvchisq(n, nu = 5)`
23) Cosine distribution: `propagate:::rcosine(n, mu = 5, sigma = 1)`
24) Pareto distribution: `propagate:::rpareto(n, scale = 1, shape = 1)`
25) Levy distribution: `propagate:::rlevy(n, loc = 0, scale = 1)`
26) Gompertz distribution: `propagate:::rgompertz(n, shape = 1, rate = 1)`
27) 3P log-normal distribution (Johnson SL): `propagate:::rlnorm3(n, xi = 0, meanlog = 0, sdlog = 1)`
28) 3P gamma distribution (Pearson type III): `propagate:::rgamma3(n, xi = 0, shape = 1, rate = 1)`
29) Rectangular-normal convolution: `propagate:::runorm(n, a = -1, b = 1, sigma = 0.1)`
30) Exponential power distribution: `propagate:::rexpow(n, mu = 0, alpha = 1, beta = 2)`
31) Skew-t distribution: `propagate:::rskewt(n, xi = 0, omega = 1, alpha = 0, nu = 10)`
32) Rice distribution: `propagate:::rrice(n, nu = 1, sigma = 1)`
33) Folded normal distribution: `propagate:::rfnorm(n, mu = 0, sigma = 1)`
34) Two-component normal mixture: `propagate:::rnmix(n, p = 0.5, mu1 = -1, sd1 = 1, mu2 = 1, sd2 = 1)`

Generation methods.
Inverse transform method ($U \sim \mathcal{U}(0,1)$, $Y = F^{-1}(U, \beta)$): 2), 4) - 7), 9) - 12), 15), 17), 19), 20), 24), 26). The generalized trapezoidal distribution 15) is inverted in closed form segment by segment (van Dorp & Kotz 2003).
Transformation of normal, gamma or chi-square variables: 1) (skew-normal representation $\delta|U_0| + \sqrt{1-\delta^2}U_1$), 3) ($x + u \cdot t_\nu$, scale u as in JCGM 101:2008, 6.4.9), 8) (scaled beta variable $a + (b-a)B$), 16) (binomial selection from a χ^2-distribution), 18), 21), 22), 25) ($\mu + c/Z^2$), 27) ($\xi + e^{\mu+\sigma Z}$), 28) ($\xi + G$ with G a gamma variable), 30) ($(|X-\mu|/\alpha)^\beta \sim \Gamma(1/\beta, 1)$ with random sign), 31) (skew-normal divided by $\sqrt{\chi^2_\nu/\nu}$), 32) (magnitude of a complex Gaussian), 33) (absolute value of a normal variable).
Convolution and mixture by construction: 29) (sum of a rectangular and a normal variable), 34) (normal variable from the first component with probability p).
Exact direct method: 14) as in JCGM 101:2008, 6.4.3.4.
Exact rejection sampling: 13) by the algorithm of Best & Fisher (1979) with a wrapped Cauchy envelope; 23) with a uniform envelope and the exact maximum of the density (acceptance rate 1/2).

The code for the random generators is in file "distr-samplers.R".

Value

A vector with n samples from the corresponding distribution.

Author(s)

Andrej-Nikolai Spiess

References

Efficient simulation of the von Mises distribution.
Best DJ, Fisher NI.
Applied Statistics (1979), **28**: 152-157.

Generalized trapezoidal distributions.
van Dorp JR, Kotz S.
Metrika (2003), **58**: 85-97.

The skew-normal distribution and related multivariate families.
Azzalini A.
Scandinavian Journal of Statistics (2005), **32**: 159-188.

Rejection Sampling in R:
Rejection Sampling.
https://www.r-bloggers.com/2011/06/rejection-sampling/.

Rejection Sampling in general:
Non-uniform random variate generation.
Devroye L.
Springer-Verlag, New York (1986).

Distributions:
Continuous univariate distributions, Volume 1.
Johnson NL, Kotz S and Balakrishnan N.
Wiley Series in Probability and Statistics, 2.ed (2004).

See Also

propagate and makeDat, in which the JCGM 101:2008 examples use these distributions; fitDistr.

Examples

```
## The generators are internal: call them with 'propagate:::'.
## Trapezoidal distribution.
set.seed(123)
X1 <- propagate:::rtrap(10000, a = 0, b = 1, c = 2, d = 3)
hist(X1, breaks = 50)

## Random samples from the von Mises distribution.
set.seed(123)
X <- propagate:::rmises(2000, mu = 1, kappa = 2)
```

```
## Then we fit a selection of distributions
## (distsel = 20:25) with 'fitDistr'.
fitDistr(X, distsel = 20:25)
## => von Mises wins! (lowest BIC)
```

`report` *PDF report of one or several uncertainty evaluations*

Description

Writes a PDF report of one object returned by `propagate`, or of a list of them (*e.g.* all examples of a demo), with about two pages per evaluation: measurement model, input data with distributions and Type A/B classification, result statement of the GUM uncertainty framework (GUF) and the Monte Carlo method (MCM), results of all methods, validation of the GUF by the MCM, numerical accuracy of the Monte Carlo results, nonlinearity of the model, uncertainty budget with first-order and Sobol' indices, histogram, Q-Q plot and convergence trace of the Monte Carlo output, an optional conformity assessment, derivatives, covariance check and a reproducibility record. If gauge (reference) values are supplied, the report adds a comparison with them.

Usage

```
report(object, file = tempfile(fileext = ".pdf"), title = "Uncertainty evaluation",
       measurand = NULL, unit = "", compare = TRUE, nsim.compare = 1E5,
       fitDistr = FALSE, fit.nmax = 1E4, gauge.mean = NULL, gauge.u = NULL,
       gauge.method = c("GUF", "MC"), gauge.dec = NULL, gauge.M = 1E6,
       interval = c("symmetric", "shortest"), ndig = 2, sobol = TRUE,
       conformity = NULL, seed = NULL, fontsize = 1)
```

Arguments

`object`	an object returned by `propagate`, or a list of such objects, one per evaluation. The names of the list are used as page titles if `title` is not given.
`file`	the path and name of the PDF file, *e.g.* 'c:/temp/test.pdf.
`title`	the title in the page header.
`measurand, unit`	symbol and unit of the output quantity (default symbol: the name of the expression, else `"y"`); an underscore gives a subscript, units such as `"Ohm"` or `"degC"` are typeset with their symbols.
`compare, nsim.compare`	if `TRUE` and the object has Type A inputs (`type = "stat"` or `"raw"`), the Monte Carlo simulation is repeated with the other input model (`"S1"` or `"mvt"`) with `nsim.compare` trials, if applicable.
`fitDistr, fit.nmax`	if `fitDistr = TRUE`, all distributions of `fitDistr` are fitted to the Monte Carlo output (at most `fit.nmax` values) and the best one by BIC is drawn on the histogram.

`gauge.mean`, `gauge.u`
: gauge (reference) values of the estimate and the standard uncertainty, *e.g.* published or certified values. If given, the report compares them with the result of `gauge.method` and adds the section 'Comparison with the gauge values'; the histogram shows the gauge value $\pm ku$.

`gauge.method`
: `"GUF"`: comparison with the first-order GUM uncertainty framework; `"MC"`: with the Monte Carlo method.

`gauge.dec`
: the numbers of decimals of `gauge.mean` and `gauge.u`, which define the rounding tolerance; by default taken from the values (set it when trailing zeros matter, *e.g.* `64.0`).

`gauge.M`
: the number of Monte Carlo trials of the gauge values (`gauge.method = "MC"`).

`interval`
: the Monte Carlo coverage interval shown and used for the validation and the numerical accuracy: probabilistically symmetric or shortest.

`ndig`
: the number of significant digits of the standard uncertainty $u(y)$ that defines the numerical tolerance δ of the validation, as in `summary.propagate`.

`sobol`
: logical. If `TRUE` (default) and the input quantities are independent, the first-order Sobol' indices of the Monte Carlo output are computed with `sobol` and shown in the uncertainty budget.

`conformity`
: `NULL` (default) or a list with the elements `lower`, `upper` (tolerance limits) and optionally `rule` (decision rules, default `c("stringent", "simple")`, `"all"` for all six), `method` (`"MC"`, default, or `"GUF"`) and `p` (coverage probability of the expanded uncertainty, default `1 - alpha`), passed to `conformity`. If given, page 2 shows the conformity assessment. For a list of objects, a list of such lists, one per evaluation (`NA` or `NULL`: none).

`seed`
: the seed of the random number generator used before `propagate` (*e.g.* `set.seed(123)`), for the reproducibility record. The `propagate` object does not contain the seed, so it must be given here; `NULL`: 'not recorded'.

`fontsize`
: a multiplier for all font sizes.

Details

Contents. Each evaluation starts on a new page; the sections follow in this order and move to the next page as a whole if they do not fit, so that an evaluation usually takes two pages (models with many input quantities or many decision rules may need a third page).

- *Measurement model* and *input data*: raw observations for `type = "raw"` with at most 15 rows; for each input quantity the estimate, standard uncertainty, degrees of freedom, the distribution and the type of evaluation (see below); the correlation matrix of correlated inputs.
- *Result statement* of the GUF and the MCM, rounded to two significant digits of the standard uncertainty (JCGM 100:2008, 7.2.6), and the *results of all methods* (GUF first and second order, the latter also with the input moments, `u.2m` (moments extension, see `propagate`), Monte Carlo with the input models `"S1"` and `"mvt"`; GUF first order and `"S1"` in dark red).
- *Validation of the GUF* and *numerical accuracy of the Monte Carlo results* (see below).
- *Nonlinearity of the model and shape of the output* (see below), and the *comparison with gauge values*, if given.

- *Uncertainty budget*: estimates, standard uncertainties, degrees of freedom, sensitivity coefficients c_i, contributions $|c_i|u(x_i)$, the first-order index $c_i^2 u^2(x_i)/u^2(y)$ (blue bar), the share of the covariance terms (negative: dark red) and the first-order Sobol' index S_i of the Monte Carlo output (violet bar), with $1-\sum S_i$ as 'interactions'. Bars are scaled to the largest absolute share if one exceeds 100 %.
- *Histogram* of the Monte Carlo output with the Monte Carlo and GUF coverage intervals (and the fitted distribution and the gauge band, if requested); a *Q-Q plot* of the quantiles of the Monte Carlo output against those of the GUF distribution (scaled and shifted t with ν_{eff} degrees of freedom, or Gaussian), with the endpoints of the coverage intervals marked: points off the diagonal show where the GUF distribution, and with it the coverage factor k, does not describe the output; and a *convergence trace* of the running mean and standard deviation over the trials, relative to the final values in units of u.
- *Conformity assessment* (optional, argument `conformity`): tolerance interval, conformance probability p_c (JCGM 106:2012, 7) and, for each decision rule, a bar with the acceptance (green), uncertainty (grey) and rejection (red) zones, the estimate and the decision (ISO 14253-1, see `conformity`).
- *Sensitivity coefficients and Hessian matrix*: $c_i = \partial f/\partial x_i$, symbolic and evaluated, with the evaluated Hessian matrix as additional columns (for at most six input quantities), and the symbolic Hessian (lower triangle) for at most three input quantities (otherwise see `summary.propagate`); long expressions are shortened with '...'.
- *Covariance check* of the Monte Carlo input samples, as in `summary.propagate`, and, for Type A inputs, a footnote on the Monte Carlo input models (`"S1"`: *t*-margins, correlated inputs joined by a Gaussian copula, Possolo 2010; `"mvt"`: multivariate *t*-distribution).
- *Reproducibility record* (see below).

Distribution and type of evaluation. If the input data carry an uncertainty budget (from `makeDat` with `output = "sim"`, or `corDist`), the distributions are taken from it; an input with a *t*-distribution and finite degrees of freedom is Type A, all others Type B. Otherwise, for `type = "raw"` all inputs are Type A (*t* or `mvt`); for `type = "stat"`, inputs with finite degrees of freedom (< 1000) are Type A, the others Type B (Gaussian); for `type = "sim"` without a budget, the distribution is 'sample'. The budget is found in `object$input$budget` or as the attribute `"budget"` of `object$input$data`.

Validation of the GUF by the MCM (JCGM 101:2008, 8). As in `summary.propagate`: the numerical tolerance $\delta = \frac{1}{2}10^l$ for $u(y) = c \times 10^l$ with `ndig` significant digits in c, the differences d_{low} and d_{high} between the endpoints of the GUF and the Monte Carlo coverage intervals, and twice the numerical standard uncertainty s of the Monte Carlo endpoints (`sim.se` of the object). The GUF is validated if $d_{\text{low}}, d_{\text{high}} \leq \delta$; the comparison is inconclusive if $2s > \delta$. In addition, the report gives the *Monte Carlo coverage of the GUF interval*, the fraction of the Monte Carlo output within the GUF coverage interval, next to the nominal coverage probability: for a nonlinear model it can be much smaller, *e.g.* about 33 % instead of 95 % for $Y = X_1^2 + X_2^2$ with $x_1 = 0.001$, $x_2 = 0$, $u(x_1) = u(x_2) = 0.005$ (JCGM 101:2008, 9.4).

Numerical accuracy of the Monte Carlo results (JCGM 101:2008, 7.9). For the estimate, the standard uncertainty and the endpoints of the coverage interval, with the numerical standard uncertainties s from 10 blocks of the Monte Carlo trials (`sim.se`): the number of decimals d for which $2s \leq \frac{1}{2}10^{-d}$, the value rounded to d decimals ('stable value') and its number of stable significant digits. Digits beyond these are Monte Carlo noise; increase `nsim` or use `adaptive = TRUE` in `propagate` for more digits.

Nonlinearity of the model and shape of the output. Five measures, each zero for a linear model with Gaussian inputs: the share of the second-order terms in the second-order variance with the input moments, $(u_{2m}^2 - u_1^2)/u_{2m}^2$ (moments extension; for Gaussian inputs $\frac{1}{2}\mathrm{tr}(\mathbf{H\Sigma H\Sigma})/u_2^2$, JCGM 100:2008, 5.1.2, note); the shift of the estimate $(\bar{y}_2 - \bar{y}_1)/u_1$; the quantile skewness $(q_{0.975} + q_{0.025} - 2q_{0.5})/(q_{0.975} - q_{0.025})$, which exists also for heavy-tailed output; and the skewness and excess kurtosis (moments) of the Monte Carlo output. They are rated 'negligible', 'moderate' or 'large' (limits 1 % and 10 %, 0.05 and 0.2, 0.05 and 0.2, 0.1 and 0.5, 0.2 and 1); skewness and kurtosis are not rated for inputs with $\nu \leq 4$, whose moments may not exist. Large values indicate that the first-order GUF may be inadequate and the Monte Carlo results should be preferred (JCGM 101:2008, 5.8). Note that the excess kurtosis also reflects *t*-distributed inputs (*e.g.* $6/(\nu - 4)$ for a linear model with one *t*-input), not only nonlinearity.

Sobol' indices. The first-order index S_i is the share of the variance of the Monte Carlo output due to x_i alone, including all nonlinear effects; unlike the first-order index of the GUF it does not vanish for $c_i = 0$. It is defined only for independent input quantities and is therefore computed with `sobol` from the Monte Carlo input samples only for them (not for correlated inputs or `mc.margins = "mvt"`).

Reproducibility record (ISO/IEC 17025:2017, 7.5, 7.8). The call of `propagate` (if stored in the object as `call`, otherwise reconstructed from the object and marked as such), the number of Monte Carlo trials, `alpha`, the input model, `cov` and the coverage interval, the seed (argument `seed`), the kind of random number generator of the session, the versions of `propagate` and R, the platform and the date.

Several evaluations. For a list of objects, `title`, `measurand`, `unit`, `gauge.mean`, `gauge.u`, `gauge.dec`, `seed` and `conformity` may be vectors or lists with one element per evaluation (recycled); `NA` means 'not given' for that evaluation, *e.g.* `gauge.mean = c(-0.1494, NA)`. For `gauge.dec`, which has two values per evaluation, use a list, *e.g.* `list(c(4, 4), NA)`; for `conformity`, a list of lists, *e.g.* `list(list(lower = 994, upper = 1006), NA)`. All other arguments apply to all evaluations.

Comparison with gauge values. The tolerance is half a unit of the last given decimal of the gauge value; for `gauge.method = "MC"`, four Monte Carlo standard errors of both calculations are added (from `sim.se` of the object and `gauge.M`). A check passes if $|value - gauge| \leq$ tolerance.

Value

Invisibly, for one object a list with the elements `file`, `results` (the table of results), `validation` (validation of the GUF, or `NULL`), `accuracy` (numerical accuracy of the Monte Carlo results, or `NULL`), `budget` (the uncertainty budget), `sobol` (the result of `sobol`, or `NULL`), `nonlinearity` (the five measures of nonlinearity with their ratings), `derivatives` (symbolic and evaluated sensitivity coefficients), `shape` (skewness and excess kurtosis), `checks` (against the gauge values, or `NULL`), `conformity` (the result of `conformity`, or `NULL`), `fit` (the distribution fit, or `NULL`) and `reproducibility` (the lines of the reproducibility record). For a list of objects, a list with `file` and `analyses`, one such list (without `file`) per evaluation.

Author(s)

Andrej-Nikolai Spiess

References

Evaluation of measurement data - Guide to the expression of uncertainty in measurement.
JCGM 100:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1002008E

Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

Evaluation of measurement data - Supplement 2 to the "Guide to the expression of uncertainty in measurement" - Extension to any number of output quantities.
JCGM 102:2011. Joint Committee for Guides in Metrology (2011).
doi:10.59161/JCGM1022011

Evaluation of measurement data - The role of measurement uncertainty in conformity assessment.
JCGM 106:2012. Joint Committee for Guides in Metrology (2012).
doi:10.59161/JCGM1062012

Geometrical product specifications (GPS) - Inspection by measurement of workpieces and measuring equipment - Part 1: Decision rules for verifying conformity or nonconformity with specifications.
ISO 14253-1:2017. International Organization for Standardization (2017).

General requirements for the competence of testing and calibration laboratories.
ISO/IEC 17025:2017. International Organization for Standardization (2017).

Copulas for uncertainty analysis.
Possolo A.
Metrologia (2010), **47**: 262-271.
doi:10.1088/00261394/47/3/017

See Also

propagate, summary.propagate, sobol, conformity, corDist, makeDat, fitDistr

Examples

```
## GUM H.3: calibration of a thermometer, compared with the published values.
## The PDF is written to 'file', e.g. file = file.path(tempdir(), "report.pdf").
data(H.3); fit <- lm(bk ~ I(tk - 20), data = H.3)
V12 <- vcov(fit); dimnames(V12) <- list(c("y1", "y2"), c("y1", "y2"))
set.seed(123)
H3 <- propagate(expression(y1 + y2 * (30 - 20)),
                cbind(y1 = c(coef(fit)[1], sqrt(V12[1, 1]), 9),
                y2 = c(coef(fit)[2], sqrt(V12[2, 2]), 9)),
                type = "stat", cov = V12, ws.group = c(1, 1),
                mc.margins = "mvt", nsim = 1E5)
f <- tempfile(fileext = ".pdf")
RES <- report(H3, file = f, title = "GUM H.3: calibration of a thermometer",
              measurand = "b", unit = "degC", seed = 123,
              gauge.mean = -0.1494, gauge.u = 0.0041)
RES$validation    ## INCONCLUSIVE at nsim = 1E5 (2s > delta): increase nsim
RES$accuracy
```

`sobol` *Sobol' sensitivity indices from a propagate object or sample matrices*

Description

Computes first-order and total-order Sobol' sensitivity indices from an object returned by `propagate`, using its Monte Carlo simulated samples, or from two external sample matrices and an expression.

Usage

```
sobol(prop = NULL, A = NULL, B = NULL, expr = NULL,
      method = c("jansen", "saltelli", "sobol", "homma"))
```

Arguments

`prop`	a `propagate` object.
`A`	an $M \times N$ matrix of Monte Carlo samples with column names matching the variables in `expr`. Only used together with `B` and `expr`.
`B`	a second $M \times N$ matrix, sampled independently of `A` from the same distributions, with identical dimensions and column names.
`expr`	an `expression` to be evaluated on the rows of `A`, `B` and the hybrid matrices.
`method`	the estimator(s) for the sensitivity indices. See 'Details'.

Details

Sampling design: $\mathbf{A}$ and $\mathbf{B}$ are two independent $M \times N$ sample matrices of the inputs. For each input X_i, the hybrid matrix $\mathbf{A}_B^{(i)}$ equals $\mathbf{A}$ except for column i, which is taken from $\mathbf{B}$. The model is evaluated on $\mathbf{A}$, $\mathbf{B}$ and all N hybrid matrices, i.e. $M(N+2)$ evaluations in total (Saltelli 2002; Saltelli *et al.* 2010).

If `prop` is supplied, $\mathbf{A}$ and $\mathbf{B}$ are the two independent copula draws `prop$datSIM` and `prop$datSIM2`. For `type = "sim"` objects, which contain only the user-supplied samples, the rows are split into two halves that serve as $\mathbf{A}$ and $\mathbf{B}$. This requires the rows to be independent draws.

Mean, variance and centring: All model outputs are first centred on their pooled mean. This leaves every estimator unchanged in expectation, but it is essential for the product-type estimators (`"sobol"`, `"saltelli"` first-order, `"homma"`): without centring, their Monte Carlo variance scales with f_0^2 and they become unusable when $|f_0| \gg \sigma_Y$, the typical situation in metrology (see 'Examples'). The pooled mean and variance of the centred outputs are

$$f_0 = \frac{1}{2M}\sum_{v=1}^{M}\left[f(\mathbf{A})_v + f(\mathbf{B})_v\right], \quad V = \frac{1}{2M-1}\sum_{v=1}^{M}\left[\left(f(\mathbf{A})_v - f_0\right)^2 + \left(f(\mathbf{B})_v - f_0\right)^2\right].$$

Rows for which any of the $N+2$ evaluations is not finite are removed from all of them.

Estimators: All sums run over $v = 1, \ldots, M$. The first-order index S_i is the fraction of the output variance explained by X_i alone; the total-order index S_{T_i} additionally includes all interactions of X_i with the other inputs.

`method = "jansen"` (default), Jansen (1999) for both indices:

$$S_i = \frac{V - \frac{1}{2M}\sum_v \left[f(\mathbf{B})_v - f(\mathbf{A}_B^{(i)})_v\right]^2}{V}, \quad S_{T_i} = \frac{\frac{1}{2M}\sum_v \left[f(\mathbf{A})_v - f(\mathbf{A}_B^{(i)})_v\right]^2}{V}$$

Both estimators are built from squared differences only and are therefore insensitive to the output mean even without centring.

`method = "saltelli"`, Saltelli *et al.* (2010) for S_i and Jansen (1999) for S_{T_i}, the combination recommended by Saltelli *et al.* (2010):

$$S_i = \frac{\frac{1}{M}\sum_v f(\mathbf{B})_v \left[f(\mathbf{A}_B^{(i)})_v - f(\mathbf{A})_v\right]}{V}, \quad S_{T_i} \text{ as for Jansen (1999).}$$

`method = "sobol"`, Sobol' (1993) for S_i and Sobol' (2001) for S_{T_i}:

$$S_i = \frac{\frac{1}{M}\sum_v f(\mathbf{B})_v\, f(\mathbf{A}_B^{(i)})_v - f_0^2}{V}, \quad S_{T_i} = \frac{\frac{1}{M}\sum_v f(\mathbf{A})_v \left[f(\mathbf{A})_v - f(\mathbf{A}_B^{(i)})_v\right]}{V}$$

Sobol' (1993) is usually written with the pair $f(\mathbf{A})_v f(\mathbf{B}_A^{(i)})_v$. Here the mirror pair $f(\mathbf{B})_v f(\mathbf{A}_B^{(i)})_v$ is used, which also shares X_i and differs in all other inputs, so that no second set of hybrid matrices is needed.

`method = "homma"`, Homma & Saltelli (1996), total-order index only (`S` is `NA`):

$$S_{T_i} = \frac{V - \frac{1}{M}\sum_v f(\mathbf{A})_v\, f(\mathbf{A}_B^{(i)})_v + f_0^2}{V}$$

The estimators and their notation follow Tables 1 and 2 of Puy *et al.* (2022). Jansen (1999) is recommended as the default because it is robust and needs no f_0 correction. The estimators differ in their Monte Carlo variance, not in their expectation, so all methods converge to the same indices with increasing M.

Dependent inputs: Sobol' indices are based on the functional ANOVA decomposition, which assumes independent inputs. With correlated inputs (a correlated Gaussian copula, `cov = "full"` or an external covariance matrix in `propagate`), the hybrid matrices $\mathbf{A}_B^{(i)}$ no longer follow the joint input distribution, and the indices lose their interpretation as shares of the output variance. `sobol` warns if any empirical input correlation exceeds $4/\sqrt{M}$ in absolute value (about four standard errors under independence). Inputs from `mc.margins = "mvt"` always trigger a warning, because the multivariate t-distribution couples all inputs through a common chi-square variable even when they are uncorrelated.

Value

A list with the following components:

`S` vector of first-order indices S_i (`NA` for `method = "homma"`).

`ST`	vector of total-order indices S_{T_i}.
`method`	the method used.
`N`	the number of rows M used, after removal of non-finite evaluations.
`V`	the pooled output variance V.
`f0`	the pooled output mean before centring.

Note

The indices are Monte Carlo estimates and carry a numerical uncertainty that decreases with $1/\sqrt{M}$. Small negative values, or first-order indices slightly above the corresponding total-order indices, are sampling noise. For `propagate` objects, M equals `nsim`; for `type = "sim"`, it is half the number of simulated rows.

Author(s)

Andrej-Nikolai Spiess

References

Sensitivity estimates for nonlinear mathematical models.
Sobol IM.
Math Model Comput Exp (1993), **1**: 407-414.

Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates.
Sobol IM.
Math Comp Sim (2001), **55**: 271-280.

Importance measures in global sensitivity analysis of nonlinear models.
Homma T, Saltelli A.
Rel Eng Sys Safety (1996), **52**: 1-17.

Analysis of variance designs for model output.
Jansen MJW.
Comp Phys Comm (1999), **117**: 35-43.

Making best use of model evaluations to compute sensitivity indices.
Saltelli A.
Comp Phys Comm (2002), **145**: 280-297.

Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index.
Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S.
Comp Phys Comm (2010), **181**: 259-270.

sensobol: an R package to compute variance-based sensitivity indices.
Puy A, Lo Piano S, Saltelli A, Levin SA.
J Stat Softw (2022), **102**(5): 1-37.

An importance quantification technique in uncertainty analysis for computer models.
Ishigami T, Homma T.
Proceedings of the First International Symposium on Uncertainty Modeling and Analysis (1990): 398-403.

See Also

propagate, stochContr.

Examples

```
## Example from 'propagate', independent inputs.
EXPR1 <- expression(x/y)
x <- c(5, 0.01, 12)
y <- c(1, 0.01, 5)
DF1 <- cbind(x, y)
RES1 <- propagate(expr = EXPR1, data = DF1, type = "stat", nsim = 10000, check = FALSE)
sobol(RES1)

## Classical Ishigami function (a = 7, b = 0.1) for gauging.
## Analytical values:
##   S  = 0.3139, 0.4424, 0
##   ST = 0.5576, 0.4424, 0.2437
## Homma & Saltelli (1996), Table 4, total-order: 0.557, 0.444, 0.241
set.seed(123)
N <- 1e5
A <- cbind(x1 = runif(N, -pi, pi), x2 = runif(N, -pi, pi), x3 = runif(N, -pi, pi))
B <- cbind(x1 = runif(N, -pi, pi), x2 = runif(N, -pi, pi), x3 = runif(N, -pi, pi))
EXPR2 <- expression(sin(x1) + 7 * (sin(x2))^2 + 0.1 * x3^4 * sin(x1))
sobol(A = A, B = B, expr = EXPR2, method = "jansen")
sobol(A = A, B = B, expr = EXPR2, method = "saltelli")
sobol(A = A, B = B, expr = EXPR2, method = "sobol")
sobol(A = A, B = B, expr = EXPR2, method = "homma")

## Same function with a large offset, as for a measurand whose mean is
## much larger than its uncertainty. Thanks to the internal centring,
## all methods return the same indices as above.
EXPR3 <- expression(1E4 + sin(x1) + 7 * (sin(x2))^2 + 0.1 * x3^4 * sin(x1))
sobol(A = A, B = B, expr = EXPR3, method = "saltelli")

## Simulated inputs (type = "sim"): GUM Supplement 1, 9.3, mass calibration.
## The simulated rows are split into A and B. The constants Pa0 and Mnom get
## first-order indices within Monte Carlo noise of 0 (here about 0.006) and
## total indices exactly 0. Pa and Pw have S about 0 but ST about 0.5:
## they act only through their interaction (Pa - Pa0) * (1/Pw - 1/Pr).
EXPR4 <- expression((Mrc + dMrc) * (1 + (Pa - Pa0) * ((1/Pw) - (1/Pr))) - Mnom)
Mrc <- rnorm(1E5, 100000, 0.050)
dMrc <- rnorm(1E5, 1.234, 0.020)
Pa <- runif(1E5, 1.10, 1.30)
Pw <- runif(1E5, 7000, 9000)
Pr <- runif(1E5, 7950, 8050)
Pa0 <- 1.2
Mnom <- 100000
DF4 <- cbind(Mrc, dMrc, Pa, Pw, Pr, Pa0, Mnom)
RES4 <- propagate(expr = EXPR4, data = DF4, type = "sim", check = FALSE)
sobol(RES4)
```

`statVec` *Transform an input vector into one with defined mean and standard deviation*

Description

Transforms an input vector into one with defined μ and σ by using a scaled-and-shifted Z-transformation.

Usage

```
statVec(x, mean, sd)
```

Arguments

`x`	the input vector to be transformed.
`mean`	the desired mean of the created vector.
`sd`	the desired standard deviation of the created vector.

Details

Calculates the vector V by a Z-transformation of the input vector X and subsequent scaling by `sd` and shifting by `mean`:

$$V = \frac{X - \bar{X}}{s_X} \cdot \mathrm{sd} + \mathrm{mean}$$

Only location and scale are changed: the shape of the distribution of `x` (skewness, kurtosis) is preserved. The mean and standard deviation of `x` are computed from its finite values, so that missing and infinite values remain as they are; `x` needs at least two finite values that are not all equal. This can be used, *e.g.*, to create Monte Carlo samples of an input quantity with exactly the stated estimate and standard uncertainty for `propagate` with `type = "sim"`.

Value

A vector of the same length as `x`, with mean equal to `mean` and standard deviation equal to `sd` (of its non-missing values).

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Create a vector of 10 values with mean = 5 and s.d. = 2.
x <- rnorm(10, 5, 2)
mean(x) ## => mean is not 5!
sd(x)   ## => s.d. is not 2!

z <- statVec(x, 5, 2)
mean(z) ## => mean is 5!
sd(z)   ## => s.d. is 2!
```

`stochContr` *Stochastic contribution analysis of the Monte Carlo output*

Description

Conducts a "stochastic contribution analysis" by calculating the change in the variance of the Monte Carlo output when each of the simulated input variables is kept constant at its mean, *i.e.* its uncertainty is removed.

Usage

```
stochContr(prop, plot = TRUE)
```

Arguments

`prop`	a `propagate` object.
`plot`	logical. If `TRUE`, boxplots of the original output distribution and of those with each input fixed at its mean are drawn.

Details

This function takes the Monte Carlo simulated input data from a `propagate` object (`...$datSIM`), sequentially substitutes each input X_i by its mean $\bar{x}_i$, and re-evaluates the model. Optional boxplots compare the original output distribution with those obtained by removing the uncertainty of each X_i. The contribution of X_i is the resulting decrease in output *variance*,

$$D_i = \mathrm{V}(Y) - \mathrm{V}(Y \mid X_i = \bar{x}_i),$$

and the relative contributions are $C_i = D_i / \sum_j D_j$, so that $\sum_i C_i = 1$.

Interpretation. For a linear model with independent inputs, $D_i = c_i^2 \mathrm{Var}(X_i)$ up to Monte Carlo noise, so that C_i equals the relative variance contribution of the law of propagation of uncertainty if all inputs have Gaussian margins or the same ν. In general, the expectation of $\mathrm{V}(Y \mid X_i)$ over X_i is $\mathrm{V}(Y) - V_i$, with V_i the first-order Sobol' variance, so that $D_i / \mathrm{V}(Y)$ is a one-point estimate of the first-order index S_i, evaluated at $X_i = \bar{x}_i$ instead of averaged over X_i. For nonlinear models, D_i can therefore be negative, which is reported in a message. Use `sobol` for variance-based sensitivity indices.

Dependent inputs. Fixing one input at its mean while keeping the simulated values of the others ignores their conditional distribution. For correlated inputs ($|r_{ij}| > 4/\sqrt{M}$ for any pair of the M simulated input vectors) and for `mc.margins = "mvt"`, a warning is issued, and the contributions are not interpretable.

Value

The relative contributions C_i for all input variables, with the variance differences D_i as attribute `"varDiff"`.

Author(s)

Andrej-Nikolai Spiess

See Also

sobol, summary.propagate (relative contributions from the law of propagation of uncertainty).

Examples

```
a <- c(15, 1)
b <- c(100, 5)
phi <- c(0.5, 0.02)
DAT <- cbind(a, b, phi)
EXPR <- expression(a * b^sin(phi))
RES <- propagate(EXPR, DAT, nsim = 10000)
SC <- stochContr(RES)
SC
attr(SC, "varDiff")
```

---

summary.propagate *Summary function for 'propagate' objects*

Description

Provides a printed summary of the results obtained by propagate, with statistics relevant to JCGM 100:2008 (GUM), JCGM 101:2008 (GUM Supplement 1) and JCGM 102:2011 (GUM Supplement 2), as outlined in 'Details'. This includes a validation of the GUM uncertainty framework by the Monte Carlo method (JCGM 101:2008, 8) and checks of the Monte Carlo input against the assigned input distribution.

Usage

```
## S3 method for class 'propagate'
summary(object, normality = FALSE, validate = TRUE, ndig = 2,
        interval = c("symmetric", "shortest"), sort = FALSE, width = 25, ...)
```

Arguments

object	an object returned from propagate.
normality	logical. If TRUE, the three repeated subsampling normality tests are performed. See 'Details'.
validate	logical. If TRUE, the GUM uncertainty framework is validated by the Monte Carlo method according to JCGM 101:2008, 8. See 'Details'.
interval	the Monte Carlo coverage interval against which the GUF interval is validated: the probabilistically symmetric interval (default) or the shortest interval, as in JCGM 101:2008, 9.3. For the stability check, the blocks use the same type of interval.

ndig	the number of significant decimal digits of the standard uncertainty $u(y)$ that defines the numerical tolerance δ for validate.
sort	logical. If TRUE, the rows of the first- and second-order uncertainty budgets are sorted by decreasing index.
width	the maximal width (characters) of the bars of the uncertainty budgets; in the second-order budget, the bars are scaled to the largest absolute share if one exceeds 100 %.
...	other parameters for future methods.

Details

summary.propagate prints the following items, in this order. Items that are not available for a given object (*e.g.* the Hessian if no symbolic or numerical derivatives could be obtained, or the correlation checks for a single input) are omitted.

Monte Carlo input model:

1. The input distribution assigned for the Monte Carlo simulation, mc.margins ("S1" or "mvt"), with the corresponding JCGM clause, or "user-supplied samples" for type = "sim". See 'Details' in propagate.
2. The degrees of freedom df.MC actually used for sampling. These are the per-input degrees of freedom ν_i, or the common $\nu = n - N$ for "mvt" (n observations, N input quantities).

Results (JCGM 100:2008 and JCGM 101:2008):

1. Results of the first- (mean.1, u.1) and second-order (mean.2, u.2; u.2m with the input moments, see 'Details' of propagate) uncertainty propagation, with the coverage interval $y \pm U$ of the GUM uncertainty framework (GUF).
2. Results of the Monte Carlo simulation (mean.MC, u.MC, median.MC, mad.MC) with the probabilistically symmetric coverage interval, formed from the order statistics of the M = nsim model evaluations as in JCGM 101:2008, 7.7.1.
3. The shortest coverage interval from the same order statistics (JCGM 101:2008, 7.7.2), preferable for asymmetric output distributions.
4. **Moments extension:** the standard uncertainties of the Taylor expansion next to the Monte Carlo value, u.1 (first order), u.2 (second order, Gaussian inputs), u.2m (second order with the skewness and excess kurtosis of each input) and u.MC, with their differences to u.MC in percent. For quadratic models with independent inputs, u.2m equals u.MC within the Monte Carlo noise (*e.g.* 1.622, 3.768, 2.694 and 2.696 W for the power of a heater with a rectangular phase angle, see the vignette); the remaining difference u.2m - u.MC shows higher-order terms of the model (and, for correlated inputs, their joint moments). Printed if u.2m is available.
5. The effective degrees of freedom ν_{eff} (Welch-Satterthwaite formula or its generalization to correlated components), as obtained from WelchSatter, or df.tot if supplied.
6. The coverage factor $k = t_{\max(1,\lfloor \nu_{\text{eff}} \rfloor)}(1 - \alpha/2)$, or the supplied k.
7. The expanded uncertainty $U = k \cdot u$, where u is either the first-order u.1 or the second-order u.2, see exp.uc in propagate.

Nonlinearity of the model:
Two measures of the nonlinearity of f at the input estimates, from the second-order Taylor expansion (see 'Details' of propagate):

1. The share of the second-order terms in the second-order variance with the input moments (moments extension),
$$\frac{u_{2m}^2(y) - u_1^2(y)}{u_{2m}^2(y)},$$
i.e. the part of the variance that the first-order law of propagation of uncertainty misses because of the curvature of f. For Gaussian inputs $u_{2m} = u_2$, and the share is $\frac{1}{2}\mathrm{tr}(\mathbf{H\Sigma H\Sigma})/u_2^2(y)$, which is never negative because $\mathrm{tr}(\mathbf{H\Sigma H\Sigma}) = \mathrm{tr}((\mathbf{\Sigma}^{1/2}\mathbf{H\Sigma}^{1/2})^2) \geq 0$; with skewed inputs, the term $c_i h_{ii} \gamma_{1,i} u_i^3$ can make it negative. It is zero for models that are linear at the estimates. Without `u.2m` (objects of earlier versions), `u.2` is used.
2. The shift of the estimate by the second-order term, $(\mathrm{mean.2} - \mathrm{mean.1})/u_1(y) = \frac{1}{2}\mathrm{tr}(\mathbf{H\Sigma})/u_1(y)$, in units of the first-order standard uncertainty.

Both measures are exact for a quadratic model with independent inputs (the shift for any input distribution, the share with the input moments) and approximations otherwise; they do not include the third-derivative terms of the Note to JCGM 100:2008, 5.1.2, nor nonlinearity beyond second order (*e.g.* kinks of `abs` or `min`). Large values indicate that the first-order GUF may be inadequate; the validation by the Monte Carlo method (below) is the decisive criterion.

Uncertainty budget (first order):
For each input quantity the estimate x_i, standard uncertainty $u(x_i)$, the distribution used for the Monte Carlo simulation (`"t"`, `"normal"` for degrees of freedom ≥ 1000, `"mvt"`, `"sample"` for `type = "sim"`), the degrees of freedom, the sensitivity coefficient c_i, the contribution $u_i(y) = |c_i|\, u(x_i)$ (JCGM 100:2008, 5.1.3) and the index
$$I_i = \frac{c_i^2\, u^2(x_i)}{u_1^2(y)}, \qquad u_1^2(y) = \sum_i \sum_j c_i c_j \Sigma_{ij},$$
the share of input i in the first-order variance, with a bar of proportional length (block characters in UTF-8 locales, `"#"` otherwise). For correlated inputs, the covariance terms $2\sum_{i<j} c_i c_j \Sigma_{ij}/u_1^2(y)$ are given in an additional row `correlations`, so that all rows sum to 100 %; negative covariance terms (cancellation) are shown with a bar of `"-"` and can make single indices exceed 100 %. The indices are shares of $u_1^2(y)$ itself, as in the budget tables of EA-4/02; the relative contributions `rel.contr` of the `propagate` object, $\mathrm{RC}_{ij} = |c_i c_j \Sigma_{ij}| / \sum_{k,\ell} |c_k c_\ell \Sigma_{k\ell}|$, are shares of the sum of absolute terms instead. If all first-order terms are zero ($u_1(y) = 0$), the budget is not defined, and a note is printed instead. Inputs with $c_i = 0$ whose second derivatives do not vanish (*e.g.* $x_2 = 0$ in $Y = X_1^2 + X_2^2$) contribute only through the second-order terms; a note points this out, as their index is 0. Variance-based shares from the Monte Carlo simulation are given by `sobol`.

Uncertainty budget (second order):
Printed for models with a non-zero Hessian matrix. It decomposes the second-order variance with the input moments (moments extension, `u.2m` of `propagate`),
$$u_{2m}^2(y) = \mathbf{c}^T \mathbf{\Sigma} \mathbf{c} + \frac{1}{2}\mathrm{tr}(\mathbf{H\Sigma H\Sigma}) + \sum_i \left(\frac{1}{4} h_{ii}^2 \gamma_{2,i} u^4(x_i) + c_i h_{ii} \gamma_{1,i} u^3(x_i) \right),$$
into the terms of the input quantities and of their interactions, with the skewness $\gamma_{1,i}$ and excess kurtosis $\gamma_{2,i}$ of each input (component `moments` of the `propagate` object; both 0 for Gaussian inputs, where $u_{2m} = u_2$). For independent inputs,
$$u_{2m}^2(y) = \sum_i \left[c_i^2 u^2(x_i) + \frac{1}{4} h_{ii}^2 (\gamma_{2,i} + 2) u^4(x_i) + c_i h_{ii} \gamma_{1,i} u^3(x_i) \right] + \sum_{i<j} h_{ij}^2 u^2(x_i) u^2(x_j),$$

which are the first-order (main effect) and interaction terms of the variance-based (Sobol') decomposition of the second-order Taylor model (JCGM 100:2008, 5.1.2, note). The table gives for each input the first-order term $c_i^2 u^2(x_i)$, the second-order term ($\frac{1}{2} h_{ii}^2 u^4(x_i)$ for Gaussian inputs), their sum and its share `Index` of $u_{2m}^2(y)$ with a bar, next to the first-order index (`Index 1st`) for comparison; each non-zero interaction $h_{ij}^2 u^2(x_i) u^2(x_j)$ is given in a row `x_i x x_j`, and interactions with shares below 0.05 % are pooled in the row `other interactions`. For correlated inputs, all covariance terms of first and second order are given in the row `correlations`, as they cannot be uniquely allocated to the inputs. The table thus shows inputs that the first-order budget misses: in $Y = X_1^2 + X_2^2$ with $x_1 = 0.001$, $x_2 = 0$, $u(x_1) = u(x_2) = 0.005$ and independent inputs, X_2 has a first-order index of 0 % but a second-order index of 48 %. If the second-order terms are less than 0.1 % of $u_{2m}^2(y)$, only a note is printed. With the input moments, the shares agree with the Sobol' indices of `sobol` for quadratic models with independent inputs (heater example of the vignette: 64 % for the rectangular phase angle; with Gaussian moments, as before version 1.3-0, it would be 81 %). Correlated pairs keep Gaussian joint moments, and the third-derivative term of JCGM 100:2008, 5.1.2, is not included; for strongly nonlinear models, the Monte Carlo results are decisive.

Validation of the GUM uncertainty framework by the Monte Carlo method (`validate = TRUE`, JCGM 101:2008, 8):
The endpoints of the GUF coverage interval $y \pm U_p$ are compared with those of the Monte Carlo coverage interval $[y_{\text{low}}, y_{\text{high}}]$ selected by `interval` (probabilistically symmetric by default):

$$d_{\text{low}} = |y - U_p - y_{\text{low}}|, \quad d_{\text{high}} = |y + U_p - y_{\text{high}}|$$

The numerical tolerance δ is derived from the GUF standard uncertainty $u(y) = U_p/k$: expressing $u(y) = c \times 10^l$, with c an integer of `ndig` digits, gives $\delta = \frac{1}{2} \times 10^l$ (JCGM 101:2008, 7.9.2). The GUF is validated if both d_{low} and d_{high} are not larger than δ; otherwise, the Monte Carlo results should be used. The comparison is only conclusive if the Monte Carlo coverage interval endpoints are themselves numerically stable to within δ (JCGM 101:2008, 7.9). Therefore, the standard deviation of the full-sample endpoints, $s_{\text{MC}} = \text{sd}(\text{block endpoints})/\sqrt{h}$, is taken from the h blocks of `propagate` (`sim.se`: $h = 10$ equal parts of the results, or the blocks of the adaptive procedure; the larger of the two values, for the lower and the upper endpoint, is reported as `s.MC`). If `sim.se` is not available, the stability cannot be assessed, and a note is printed. If $2 s_{\text{MC}} > \delta$, the result is reported as "INCONCLUSIVE", and `nsim` should be increased. If $u(y) = 0$, which happens when all sensitivity coefficients vanish at the input means, the GUF is reported as not validated. If inputs are sampled with $\nu \leq 4$ degrees of freedom, the output has no finite fourth moment: `u.MC` and the Monte Carlo correlations then converge slowly, and a note recommends the coverage interval endpoints as the reliable Monte Carlo results.

Derivatives and contributions:

1. The symbolic gradient ∇, also often termed $\mathbf{G}$, $\mathbf{J}$ or $\mathbf{C_x}$ (JCGM 102:2011).
2. The evaluated gradient, also known as "sensitivity coefficients", $c_i = \partial f / \partial x_i$, evaluated at the input means.
3. The symbolic Hessian matrix $\mathbf{H}$, also often termed ∇^2.
4. The evaluated Hessian matrix $h_{ij} = \partial^2 f / \partial x_i \partial x_j$, evaluated at the input means.

Checks of the Monte Carlo input (extending JCGM 101:2008):

1. The input covariance matrix $\mathbf{\Sigma}$ of the GUF, as provided from (i) the uncertainties in the 2nd row of the input `data`, (ii) estimation from the raw data, or (iii) an externally supplied one. Also often termed $\mathbf{V}$ or $\mathbf{U_x}$ (JCGM 102:2011).

2. The target covariance matrix $\boldsymbol{\Sigma}^{\text{target}}$ that the Monte Carlo input should have under the chosen `mc.margins` model (`covMat.target`, see `propagate`), printed only if it differs from $\boldsymbol{\Sigma}$. For "S1", its variances are larger than those of Σ by the factors $\nu_i/(\nu_i - 2)$, for "mvt" by $(n - 1)/(n - N - 2)$.
3. The covariance matrix estimated from the Monte Carlo input, $\hat{\boldsymbol{\Sigma}}^{\text{MC}} = \text{cov}(\mathbf{X}^{\text{MC}})$.
4. The relative Frobenius norm $\varepsilon_F = \|\hat{\boldsymbol{\Sigma}}^{\text{MC}} - \boldsymbol{\Sigma}^{\text{target}}\|_F / \|\boldsymbol{\Sigma}^{\text{target}}\|_F$, together with its ratio to the Monte Carlo noise level (see below). `"not applicable"` is printed if the target covariance does not exist ($\nu \leq 2$).
5. For correlated inputs only: the correlation matrix $\mathbf{R}$ derived from $\boldsymbol{\Sigma}$ (by `cov2cor`), used as the parameter of the Gaussian copula for `"S1"`.
6. For correlated inputs only: the relative Frobenius norm $\varepsilon_F = \|\hat{\mathbf{R}}^{\text{MC}} - \mathbf{R}\|_F / \|\mathbf{R}\|_F$ of the correlation matrix $\hat{\mathbf{R}}^{\text{MC}} = \text{cor}(\mathbf{X}^{\text{MC}})$ of the Monte Carlo input, together with its ratio to the Monte Carlo noise level. Being scale-free, it detects a wrong dependence structure also between inputs with small variances, which hardly affect the covariance norm.
7. A comparison of the input parameters (means, standard uncertainties u_i, `df.MC`) with those of scaled and shifted *t*-distributions fitted to each margin of the Monte Carlo input (only if `check = TRUE` in `propagate`). The fitted scale is reported on the scale of u_i.

Monte Carlo noise level: ε_F is never exactly zero, because $\hat{\boldsymbol{\Sigma}}^{\text{MC}}$ and $\hat{\mathbf{R}}^{\text{MC}}$ are estimates from `nsim` draws. The expected size of this sampling noise depends on the number of inputs, on `nsim` and on the tails of the margins, and is estimated empirically: the Monte Carlo input is split into two halves, the matrix is estimated from each half ($\hat{\boldsymbol{\Sigma}}^{(1)}$, $\hat{\boldsymbol{\Sigma}}^{(2)}$), and

$$\varepsilon_{\text{noise}} = \frac{\|\hat{\boldsymbol{\Sigma}}^{(1)} - \hat{\boldsymbol{\Sigma}}^{(2)}\|_F}{2\,\|\boldsymbol{\Sigma}^{\text{target}}\|_F}$$

approximates the sampling noise of the full-sample estimate, since each half-sample estimate has twice the sampling variance of the full-sample one. The ratio $\varepsilon_F/\varepsilon_{\text{noise}}$ is printed, with `"OK"` if it is at most 3 and `"CHECK"` otherwise; the same is done for the correlation matrices. A ratio above 3 indicates a systematic deviation of the Monte Carlo input from the target. Being based on a single split, $\varepsilon_{\text{noise}}$ is itself a rough estimate, and occasional ratios slightly above 3 can occur by chance. For correlated inputs joined by the Gaussian copula (`"S1"`) with small ν_i, a small systematic deviation is expected, because the Pearson correlation of *t*-margins is slightly smaller in absolute value than the copula parameter; it becomes visible as `"CHECK"` for large `nsim`. If any $\nu_i \leq 4$, the *t*-margins have infinite kurtosis, covariance estimates converge slowly and erratically, and a note is printed that the covariance check is unreliable.

Shape of the Monte Carlo output distribution:

1. Skewness and excess kurtosis of the model evaluations $Y = f(\mathbf{X}^{\text{MC}})$, see `skewness`.
2. If `normality = TRUE`: Shapiro-Wilk test (2000 repeats, 5000 samples).
3. If `normality = TRUE`: Anderson-Darling test (1000 repeats, 10000 samples).
4. If `normality = TRUE`: Lilliefors test (1000 repeats, 10000 samples).

For each test, the median *p*-value over the repeats is reported (see Thode 2002 for the tests, and Royston 1992 for the Shapiro-Wilk test of large samples). Non-finite model evaluations are removed before subsampling. With large subsamples, these tests detect even small, practically irrelevant deviations from normality; the validation of the GUF (see above) is the more relevant criterion for deciding whether the GUF coverage interval can be used.

Value

A printed output with the items listed under 'Details'. The `propagate` object is returned invisibly.

Author(s)

Andrej-Nikolai Spiess

References

Evaluation of measurement data - Guide to the expression of uncertainty in measurement.
JCGM 100:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1002008E

Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method.
JCGM 101:2008. Joint Committee for Guides in Metrology (2008).
doi:10.59161/JCGM1012008

Evaluation of measurement data - Supplement 2 to the "Guide to the expression of uncertainty in measurement" - Extension to any number of output quantities.
JCGM 102:2011. Joint Committee for Guides in Metrology (2011).
doi:10.59161/JCGM1022011

Testing for Normality.
Thode Jr. HC.
Marcel Dekker (2002), New York.

Approximating the Shapiro-Wilk W-test for non-normality.
Royston P.
Stat Comp (1992), **2**: 117-119.

For further references, see `propagate`.

See Also

`propagate`, `plot.propagate`, `WelchSatter`, `report`, `sobol`.

Examples

```
## GUM Supplement 1, 9.4, comparison loss with x1 = 0.010:
## asymmetric output distribution. The GUF is not validated
## (JCGM 101:2008, 8), and the shortest coverage interval
## differs from the probabilistically symmetric one.
## Use nsim = 1E6 in practice
EXPR1 <- expression(X1^2 + X2^2)
X1 <- c(0.010, 0.005)
X2 <- c(0, 0.005)
DF1 <- cbind(X1, X2)
RES1 <- propagate(expr = EXPR1, data = DF1, type = "stat",
                  nsim = 1E5, check = FALSE)
summary(RES1)
```

`WelchSatter` *Effective degrees of freedom: Welch-Satterthwaite formula and its generalization to correlated components*

Description

Calculates the 'effective degrees of freedom' of a combined standard uncertainty, the corresponding coverage factor and the expanded uncertainty. For independent uncertainty components, this is the Welch-Satterthwaite formula of the GUM (Welch 1947, Satterthwaite 1946). For correlated components with finite degrees of freedom, where the Welch-Satterthwaite formula is not applicable, the generalization of Willink (2007) is used, including the combination of jointly observed input quantities into a single component with $n - 1$ degrees of freedom.

Usage

```
WelchSatter(ui, ci = NULL, df = NULL, df.tot = NULL, uc = NULL, alpha = 0.05, k = NULL,
            cov = NULL, group = NULL, method = c("auto", "gen", "sim", "WS"))
```

Arguments

`ui`	the standard uncertainties u_i of the input quantities x_i.
`ci`	the sensitivity coefficients $c_i = \partial y / \partial x_i$. If `NULL`, all $c_i = 1$.
`df`	the degrees of freedom ν_i of the u_i. `Inf` is allowed for inputs with exactly known uncertainty (JCGM 100:2008, G.4.3).
`df.tot`	an optional known total degrees of freedom ν_{tot}. Overrides the calculation of ν_{eff} and is used as given.
`uc`	the combined standard uncertainty $u(y)$. If `NULL`, $u(y)^2 = \sum_i \sum_j K_{ij}$ with the contribution matrix K defined in 'Details'.
`alpha`	the significance level: the coverage probability is $1 - \alpha$.
`k`	an external coverage factor, say 2, that overrides the one obtained from `alpha`.
`cov`	an optional covariance matrix $\mathbf{\Sigma}$ of the input quantities, with $\Sigma_{ii} = u_i^2$ and $\Sigma_{ij} = r_{ij} u_i u_j$. If `NULL`, the inputs are taken as uncorrelated.
`group`	an optional vector of length m that labels input quantities estimated jointly from the same n sets of observations (equal labels = same group, `NA` = component of its own). All members of a group must have the same `df`, $n - 1$. See 'Details', Result 1.
`method`	the formula for correlated components: `"auto"` (default), `"gen"`, `"sim"`, or `"WS"` for the classical formula. See 'Details'.

Details

Contribution matrix. With sensitivity coefficients c_i, standard uncertainties u_i and correlation coefficients r_{ij}, the contribution matrix and the combined variance are (JCGM 100:2008, 5.2.2, Eq.

(13))

$$K_{ij} = c_i c_j r_{ij} u_i u_j = c_i c_j \Sigma_{ij}, \qquad u(y)^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} K_{ij}.$$

Welch-Satterthwaite formula (`method = "WS"`). For independent components (JCGM 100:2008, G.4.1, Eq. (G.2b)):

$$\nu_{\text{eff}} = \frac{u(y)^4}{\sum_{i=1}^{m} \frac{(c_i u_i)^4}{\nu_i}}.$$

It requires $r_{ij} = 0$ for every pair of components that both have finite degrees of freedom (JCGM 100:2008, G.4.2), and its use with correlated components is outside the GUM.

Generalization to correlated components (Willink 2007). Willink distinguishes two situations: jointly observed input quantities (Result 1) and correlated components with independently estimated uncertainties (Results 2 and 3).

Result 1 (jointly observed input quantities, argument `group`). If a group of input quantities is estimated from the same n sets of simultaneous observations, as in JCGM 100:2008, H.2, the standard uncertainties of these inputs are themselves correlated, and the group is combined into a single component with $n - 1$ degrees of freedom. This result is exact for sampling from a multivariate normal distribution, for any correlation and sensitivity coefficients: the variance of a fixed linear combination of the sample means is then estimated with exactly $n - 1$ degrees of freedom (Willink 2007, section 4.1). Computationally, the contribution matrix is aggregated over the groups, $K'_{GH} = \sum_{i\in G}\sum_{j\in H} K_{ij}$, and the formulas below are applied to the components of K' with $\nu_G = n_G - 1$, where n_G is the number of sets of observations of group G (n in the argument `group`). If all inputs form one group, $\nu_{\text{eff}} = n - 1$.

Result 2 (`method = "gen"`, Willink 2007, Eqs. (26)-(27)). For correlated components whose standard uncertainties are estimated independently (*e.g.* from previous evaluations or Type B):

$$\nu_{\text{eff}} = \max\left\{1,\ \frac{u(y)^4}{\sum_i \left(\sum_j K_{ij}\right)^2/\nu_i}\right\}.$$

Result 3 (`method = "sim"`, Willink 2007, Eq. (29)). The Welch-Satterthwaite value calculated as if the components were independent:

$$\nu_{\text{eff}} = \frac{\left(\sum_i K_{ii}\right)^2}{\sum_i K_{ii}^2/\nu_i}.$$

Choice (`method = "auto"`). In Willink's simulations, `"sim"` performed slightly better than `"gen"` in most situations, but considerably worse when the components cancel because of negative correlations or opposite signs of the sensitivity coefficients. `"auto"` therefore uses `"sim"` if $u(y)^2 \geq \sum_i K_{ii}$ (no net cancellation) and `"gen"` otherwise. This rule is the practical suggestion of Willink (2007, section 4.2.2), who notes that the choice of formula after seeing the data is not fully resolved; it should be regarded as a heuristic.

Without correlations, `"gen"`, `"sim"` and `"auto"` all reduce to the Welch-Satterthwaite formula, so that results for independent inputs are unchanged.

Coverage factor and expanded uncertainty.

$$k = t_{\max(1,\lfloor\nu_{\text{eff}}\rfloor)}(1-\alpha/2), \qquad U = k\,u(y).$$

For the coverage factor, ν_{eff} is truncated to the next lower integer (JCGM 100:2008, G.4.1, Note 1), but to at least 1; the returned `ws.df` is not truncated. For $\nu_{\text{eff}} = \infty$, k is the Gaussian quantile.

For independent components, $\nu_{\text{eff}} \geq \min_i \nu_i$, so that values below 3 only occur if at least one input has $\nu_i < 3$. The value is then used as it is, with a message: the output distribution is heavy-tailed, the coverage factor is large (*e.g.* $k = 12.71$ for $\nu_{\text{eff}} = 1$ and $\alpha = 0.05$), and the Monte Carlo coverage interval of `propagate` is preferable. Versions up to 1.2-0 replaced $\nu_{\text{eff}} < 3$ by 3, which gave too small coverage factors.

If $u(y) = 0$, *i.e.* all contributions vanish, ν_{eff} is undefined $(0/0)$: `ws.df` and `k` are `NA` and `u.exp` is 0.

Use in `propagate`. `propagate` passes the input covariance matrix as `cov`. For `type = "raw"` with `cov = "full"` and complete joint observations, all inputs are one group (Result 1), so that the GUM coverage interval uses $\nu_{\text{eff}} = n - 1$; up to version 1.2-0, the classical formula was applied to these correlated inputs, which is outside the GUM and does not give $n - 1$ (in the GUM H.2 example below, $\nu_{\text{eff}} = 0.13$ with the correlated $u(y)$ in the numerator and 7.1 with the inputs taken as independent, instead of 4). For summary data, jointly observed subsets can be declared with the argument `ws.group` of `propagate`.

Scope and validation. All formulas are based on the first-order Taylor expansion of the measurement model. Willink (2007) evaluated Results 2 and 3 by simulation of the coverage probability for linear models with multivariate Gaussian errors and independent chi-square distributed variance estimates, with $m = 2$ and $m = 4$ components; `"gen"` achieved coverage probabilities similar to those of the Welch-Satterthwaite formula for independent components (about 93% or more for a nominal 95%) when $\min_i \nu_i \geq 2$, and `"sim"` did so except when the components cancel (down to about 85%), which motivates the rule of `"auto"`. Their performance for nonlinear models and non-Gaussian inputs has not been established. For the reliability of the Welch-Satterthwaite formula itself, see Hall and Willink (2001). If `uc` is the second-order uncertainty from `propagate`, ν_{eff} is a further approximation.

Value

A list with the following items:

`ws.df`	the effective degrees of freedom (not truncated), or `df.tot` if supplied.
`k`	the coverage factor for calculating the expanded uncertainty.
`u.exp`	the expanded uncertainty U.
`method`	the formula used: `"WS"`, `"gen"` or `"sim"`; for `method = "auto"`, the selected one, which is `"sim"` for uncorrelated inputs (where it equals the Welch-Satterthwaite formula); `"df.tot"` if `df.tot` is supplied.

Author(s)

Andrej-Nikolai Spiess

References

Evaluation of measurement data - Guide to the expression of uncertainty in measurement. JCGM 100:2008. Joint Committee for Guides in Metrology (2008). doi:10.59161/JCGM1002008E

An Approximate Distribution of Estimates of Variance Components.
Satterthwaite FE.
Biometrics Bulletin (1946), **2**: 110-114.

The generalization of "Student's" problem when several different population variances are involved.
Welch BL.
Biometrika (1947), **34**: 28-35.

A generalization of the Welch-Satterthwaite formula for use with correlated uncertainty components.
Willink R.
Metrologia (2007), **44**: 340-349.

Does "Welch-Satterthwaite" make a good uncertainty estimate?
Hall BD, Willink R.
Metrologia (2001), **38**: 9-15.

See Also

propagate

Examples

```
## GUM H.1.6: nu_eff = 16.8 (truncated to 16), k99 = 2.92, U99 = 93 nm.
WelchSatter(ui = c(25, 9.7, 2.9, 16.6), df = c(18, 25.6, 50, 2), uc = 32, alpha = 0.01)

## A dominating input with one degree of freedom:
## nu_eff < 3 is kept, k = t_1(0.975) = 12.71.
WelchSatter(ui = c(1, 0.1), df = c(1, 50))

## Two correlated components with independently estimated
## uncertainties and strong cancellation (Willink 2007, Table 1):
## "auto" selects "gen" (nu_eff = 1.44, larger k), which keeps the
## coverage probability; "sim" (nu_eff = 2.16) gives too narrow
## intervals in this situation (coverage about 91 % instead of 95 %).
COV <- matrix(c(1, -0.8 * 0.2, -0.8 * 0.2, 0.04), 2)
WelchSatter(ui = c(1, 0.2), df = c(2, 2), cov = COV, method = "gen")
WelchSatter(ui = c(1, 0.2), df = c(2, 2), cov = COV, method = "sim")
WelchSatter(ui = c(1, 0.2), df = c(2, 2), cov = COV)

## GUM H.2: resistance R = V/I * cos(phi) from n = 5 joint
## observations of V, I and phi (Willink 2007, section 4.1.1).
## As one group, nu_eff = n - 1 = 4 exactly.
data(H.2)
X <- as.matrix(H.2)[, c("V", "I", "phi")]
m <- colMeans(X)
V <- m[["V"]]; I <- m[["I"]]; phi <- m[["phi"]]
CI <- c(cos(phi)/I, -V * cos(phi)/I^2, -V * sin(phi)/I)
SIG <- cov(X)/5
WelchSatter(ui = sqrt(diag(SIG)), ci = CI, df = rep(4, 3), cov = SIG, group = c(1, 1, 1))
## Classical formula with the correlated u(y) in the numerator
## (outside the GUM): nu_eff = 0.13, far too small ...
WelchSatter(ui = sqrt(diag(SIG)), ci = CI, df = rep(4, 3), cov = SIG, method = "WS")
## ... and with the inputs taken as independent (no cov): nu_eff = 7.1.
```

```
WelchSatter(ui = sqrt(diag(SIG)), ci = CI, df = rep(4, 3), method = "WS")
```

Index

∗ **IO**
makeDat, 26
∗ **algebra**
cor2cov, 8
mixCov, 32
propagate, 41
∗ **array**
bigcor, 2
cor2cov, 8
makeDerivs, 29
matrixStats, 31
mixCov, 32
numDerivs, 35
propagate, 41
∗ **datagen**
statVec, 69
∗ **datasets**
datasets, 12
∗ **distribution**
fitDistr, 18
rDistr, 57
∗ **hplot**
plot.propagate, 36
∗ **manip**
makeDat, 26
∗ **math**
makeDerivs, 29
numDerivs, 35
∗ **misc**
conformity, 4
doGauge, 16
interval, 23
report, 60
∗ **models**
propagateMV, 54
summary.propagate, 71
∗ **multivariate**
bigcor, 2
cor2cov, 8
corDist, 9
makeDerivs, 29
mixCov, 32
numDerivs, 35
predictNLS, 37
propagate, 41
propagateMV, 54
sobol, 65
stochContr, 70
WelchSatter, 77
∗ **nonlinear**
predictNLS, 37
summary.propagate, 71
∗ **regression**
predictNLS, 37
∗ **univar**
fitDistr, 18
matrixStats, 31
moments, 34
rDistr, 57
statVec, 69
WelchSatter, 77

apply, *46*

BIC, *18*, *21*
bigcor, 2

colVarsC *(*matrixStats*)*, 31
conformity, 4, *12*, *18*, *61–64*
cor, *2, 3*
cor2cov, 8, *28*
corDist, 9, *16*, *18*, *46*, *62*, *64*
cov, *2, 3*
cov2cor, *8*, *75*

datasets, 12
Distributions, *57*
distributions, *10*, *42*
doGauge, *14*, 16, *42*, *47*, *51*, *56*

environment, *35*
eval, *46*
evalDerivs *(*makeDerivs*)*, 29

fitDistr, 18, *59, 60*, *64*

H.1 *(*datasets*)*, 12
H.2 *(*datasets*)*, 12
H.3 *(*datasets*)*, 12
H.4 *(*datasets*)*, 12
H.5 *(*datasets*)*, 12
H.6 *(*datasets*)*, 12
hist, *37*

interval, 23

kurtosis *(*moments*)*, 34

make.unique, *32*
makeDat, *6, 7*, *12, 13*, *15*, *25*, 26, *46, 47*, *51*, *57*, *59*, *62*, *64*
makeDerivs, 29
makeGrad, *46*
makeGrad *(*makeDerivs*)*, 29
makeHess, *46*
makeHess *(*makeDerivs*)*, 29
matrix, *3*
matrixStats, 31
mixCov, 32, *39*
moments, 34

nls, *38*
numDerivs, 35
numGrad, *46*
numGrad *(*numDerivs*)*, 35
numHess, *46*
numHess *(*numDerivs*)*, 35

optim, *22*, *24*

plot.conformity *(*conformity*)*, 4
plot.propagate, 36, *76*
plot.propagateMV *(*propagateMV*)*, 54
predict.lm, *39*
predict.nls, *38*
predictNLS, *13*, 37, *43*, *48*
print.conformity *(*conformity*)*, 4
print.interval *(*interval*)*, 23
print.propagateMV *(*propagateMV*)*, 54

propagate, *4, 5*, *7*, *10–12*, *15*, *18*, *22*, *25–29*, *32*, *36–39*, 41, *54–57*, *59–65*, *68–72*, *74–76*, *79, 80*
propagateMV, *12*, *15*, *18*, 54

QUAM.A4 *(*datasets*)*, 12
QUAM.A5 *(*datasets*)*, 12

rDistr, *10–12*, *42*, 57
read.table, *27*
report, *51*, 60, *76*
rowVarsC *(*matrixStats*)*, 31

S1.9.3 *(*datasets*)*, 12
S1.9.5 *(*datasets*)*, 12
S2.9.4 *(*datasets*)*, 12
S2.9.5 *(*datasets*)*, 12
skewness, *75*
skewness *(*moments*)*, 34
sobol, *50*, *61*, *63, 64*, 65, *70, 71*, *73, 74*, *76*
statVec, 69
stochContr, *68*, 70
summary.propagate, *43*, *46, 47*, *49, 50*, *61, 62*, *64*, *71*, 71
summary.propagateMV *(*propagateMV*)*, 54

TN.E11 *(*datasets*)*, 12
TN.E2 *(*datasets*)*, 12

var, *31*

WelchSatter, *38*, *42, 43*, *50, 51*, *72*, *76*, 77