Package: RMediation (via r-universe)

November 10, 2026

Type Package

Title Mediation Analysis Confidence Intervals

Version 1.11.1

Date 2026-11-09

Description Computes confidence intervals for nonlinear functions of model parameters (e.g., product of k coefficients) in single-level and multilevel structural equation models. Methods include the distribution of the product, Monte Carlo simulation, and bootstrap methods. It also performs the Model-Based Constrained Optimization (MBCO) procedure for hypothesis testing of indirect effects. References: Tofighi, D., and MacKinnon, D. P. (2011). RMediation: An R package for mediation analysis confidence intervals. Behavior Research Methods, 43, 692-700. <doi:10.3758/s13428-011-0076-x>; Tofighi, D., and Kelley, K. (2020). Improved inference in mediation analysis: Introducing the model-based constrained optimization procedure. Psychological Methods, 25(4), 496-515. <doi:10.1037/met0000259>; Tofighi, D. (2020). Bootstrap Model-Based Constrained Optimization Tests of Indirect Effects. Frontiers in Psychology, 10, 2989. <doi:10.3389/fpsyg.2019.02989>.

License GPL (>= 3) | file LICENSE

URL https://data-wise.github.io/rmediation/, https://github.com/data-wise/rmediation

BugReports https://github.com/data-wise/rmediation/issues

Depends R (>= 4.1.0)

Imports checkmate (>= 2.1.0), cubature (>= 2.1.0), graphics (>= 4.1.0), grDevices (>= 4.1.0), lavaan (>= 0.5-20), MASS (>= 7.3), methods, S7

Suggests knitr, medfit (>= 0.2.0), mvtnorm, OpenMx (>= 2.13), rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE, roclets = c(``rd", ``collate", ``namespace"))

Config/testthat/edition 3

Config/Needs/website pkgdown, quarto

Config/roxygen2/version 8.1.0

Config/pak/sysreqs make

Repository https://data-wise.r-universe.dev

Date/Publication 2026-11-10 02:14:49 UTC

RemoteUrl https://github.com/Data-Wise/rmediation

RemoteRef main

RemoteSha 2b3907a16cb329cca32dc4d1b5919d8b8af62e1c

Contents

cdf . . . 3
ci . . . 3
ci_mediation_data . . . 4
dist_quantile . . . 6
dprodnormal . . . 6
dprodnormal3 . . . 7
mbco . . . 9
mbco_product_null . . . 13
MBCOResult . . . 14
medci . . . 15
memory_exp . . . 18
ml_omitted_confounder . . . 19
pMC . . . 21
pprodnormal . . . 22
pprodnormal3 . . . 23
ProductNormal . . . 26
ProductNormal2 . . . 26
ProductNormal3 . . . 27
qMC . . . 27
qprodnormal . . . 28
qprodnormal3 . . . 30
tidy . . . 31
tidy.logLik . . . 32

Index **33**

cdf *Cumulative Distribution Function*

Description

Generic function for computing cumulative distribution function.

Usage

```
cdf(object, ...)
```

Arguments

object	A distribution object.
...	Additional arguments passed to methods.

Value

A numeric vector of cumulative probabilities for the supplied quantiles, as returned by the dispatched method.

Examples

```
pn <- ProductNormal(mu = c(0.5, 0.3), Sigma = diag(2) * 0.1)
cdf(pn, q = 0)
```

ci *Confidence Interval*

Description

Generic function for computing confidence intervals.

Usage

```
ci(object, ...)
```

Arguments

object	A distribution object or numeric vector of means.
...	Additional arguments passed to methods.

Details

For numeric estimates, pass them as the first argument by position, followed by their covariance matrix: `ci(c(a = .5, b = .4), Sigma, quant = ~ a * b)`. The estimates cannot be named `mu`, because the generic dispatches on its first argument, `object`. The other arguments are `Sigma` (a covariance matrix, or its lower triangle by column), `quant` (a one-sided formula in the names of the estimates), `alpha` (default 0.05), `type` (`"MC"`, `"asymp"` or `"all"`), `n.mc`, `plot`, `plotCI`, `H0`, `mu0` and `Sigma0`.

For numeric vectors or lavaan fits with a formula quantity, the Monte Carlo component reports MCError as the sampling standard error of the simulated mean (SE / sqrt(n.mc)); this is not the error of the interval endpoints. With H0 = TRUE, the Monte Carlo result contains the alternative and null results. The type = "all" route preserves H0, mu0 and Sigma0 for its MC component; the asymptotic component remains the interval at the supplied estimates.

Value

A confidence interval (numeric vector of lower/upper bounds, or a list) as returned by the dispatched method.

Examples

```
pn <- ProductNormal(mu = c(0.5, 0.3), Sigma = diag(2) * 0.1)
ci(pn)
```

`ci_mediation_data` *Confidence Interval for MediationData Objects*

Description

Computes confidence intervals for the indirect effect from a medfit MediationData object using RMediation's methods (DOP, Monte Carlo, etc.).

Usage

```
ci_mediation_data(object, level = 0.95, type = "dop", n.mc = 1e+05, ...)

ci_serial_mediation_data(object, level = 0.95, type = "MC", n.mc = 1e+05, ...)
```

Arguments

`object`	A MediationData object from the medfit package
`level`	Confidence level (default 0.95 for 95% CI)
`type`	Type of CI method: "dop" (Distribution of Product), "MC" (Monte Carlo), or "asymp" (asymptotic normal)
`n.mc`	Number of Monte Carlo samples (for type = "MC")
`...`	Additional arguments passed to underlying methods

Details

This method extracts the a and b path coefficients from the MediationData object, along with their standard errors and covariance, and computes confidence intervals using RMediation's methods.

Method Options:

- **"dop"**: Distribution of Product method. Uses the exact or approximate distribution of the product of two normal random variables. Recommended for most applications.
- **"MC"**: Monte Carlo simulation. Samples from the joint distribution of a and b to estimate the CI. Use `n.mc` to control precision.
- **"asymp"**: Asymptotic normal approximation using the delta method. Fast but may be inaccurate for small samples or non-normal distributions.

Value

A list with components:

`CI`	The confidence interval (lower, upper)
`Estimate`	Point estimate of indirect effect (a*b)
`SE`	Standard error of indirect effect
`type`	Method used for CI computation

See Also

`ci` for the generic function, `MediationData` for the input class, `ProductNormal` for the underlying distribution class

Examples

```
## Not run:
library(medfit)
library(RMediation)

# Fit mediation models
fit_m <- lm(M ~ X + C, data = mydata)
fit_y <- lm(Y ~ X + M + C, data = mydata)

# Extract mediation structure
med_data <- extract_mediation(fit_m,
  model_y = fit_y,
  treatment = "X", mediator = "M"
)

# Compute CI using Distribution of Product
ci(med_data, type = "dop")

# Compute CI using Monte Carlo
ci(med_data, type = "MC", n.mc = 10000)

## End(Not run)
```

dist_quantile *Distribution Quantile Function*

Description

Compute quantiles for distribution objects. This function computes quantiles for product normal distributions, not data quantiles (use `stats::quantile` for data).

Usage

```
dist_quantile(object, ...)
```

Arguments

`object`	A distribution object (e.g., ProductNormal).
`...`	Additional arguments passed to methods.

Value

A numeric vector of quantiles for the supplied probabilities, as returned by the dispatched method.

Examples

```
pn <- ProductNormal(mu = c(0.5, 0.3), Sigma = diag(2) * 0.1)
dist_quantile(pn, p = 0.025)
```

dprodnormal *Density of the Product of Two Normal Random Variables*

Description

Evaluates the exact probability density function of XY, where (X, Y) is bivariate normal. It is the density companion to `pprodnormal` and `qprodnormal` and uses no Monte Carlo simulation.

Usage

```
dprodnormal(x, mu.x, mu.y, se.x = 1, se.y = 1, rho = 0, log = FALSE)
```

Arguments

`x`	numeric vector of values at which to evaluate the density (the value of the product).
`mu.x`, `mu.y`	means of X and Y.
`se.x`, `se.y`	standard deviations of X and Y; must be positive.
`rho`	correlation between X and Y; must satisfy $-1 < \rho < 1$. The default is 0.
`log`	logical; if `TRUE`, the logarithm of the density is returned.

Details

For general means and correlation the density is the one-dimensional integral $f(z) = \int f_{X,Y}(x, z/x)\,|x|^{-1}\,dx$ (Rohatgi; Glen et al., 2004), evaluated by adaptive quadrature over the range that carries the mass of the conditioning variable and split at the sign change at zero. When both means are zero the closed form

$$f(z) = \frac{\exp\{\rho z/(1-\rho^2)\}\,K_0\{|z|/(1-\rho^2)\}}{\pi\sqrt{1-\rho^2}}$$

(unit variances; rescaled here) is used instead, with K_0 the modified Bessel function of the second kind (Nadarajah and Pogany, 2016; Gaunt, 2019). The two agree to near machine precision.

The density is infinite at exactly `x = 0` for every parameter setting (a logarithmic singularity: XY is near zero whenever either factor is), so `dprodnormal(0, ...)` returns `Inf`. Plot it on a grid that avoids 0 and clip the vertical axis.

Value

A numeric vector of densities (or log densities), one per element of `x`.

References

Craig, C. C. (1936). On the frequency function of xy. *Annals of Mathematical Statistics*, **7**, 1–15.

Glen, A. G., Leemis, L. M. and Drew, J. H. (2004). Computing the distribution of the product of two continuous random variables. *Computational Statistics & Data Analysis*, **44**, 451–464.

Nadarajah, S. and Pogany, T. K. (2016). On the distribution of the product of correlated normal random variables. *Comptes Rendus Mathematique*, **354**, 201–204.

See Also

`pprodnormal`, `qprodnormal`

Examples

```
dprodnormal(c(-0.1, 0.05, 0.3), mu.x = 0.5, mu.y = 0.3, se.x = 0.2, se.y = 0.4, rho = 0.3)

# the density integrates to the CDF
integrate(dprodnormal, 0.001, 0.3, mu.x = 0.5, mu.y = 0.3, se.x = 0.2, se.y = 0.4)$value
pprodnormal(0.3, 0.5, 0.3, 0.2, 0.4) - pprodnormal(0.001, 0.5, 0.3, 0.2, 0.4)
```

`dprodnormal3` *Density of the Product of Three Normal Variables*

Description

Exact density of $X_1X_2X_3$ when (X_1, X_2, X_3) is trivariate normal with mean `mean` and covariance `cov`, the density counterpart of `pprodnormal3()`. No closed form is known for the general case (Gaunt, 2022), so the density is a two-dimensional integral: conditioning on $X_1 = x$ makes (X_2, X_3) bivariate normal, whose product has the density of `dprodnormal()`, and

$$f(v) = \int \phi(x; \mu_1, \sigma_1^2)\, f_{X_2X_3|x}(v/x)\, /|x|\, dx.$$

The inner density is the Rohatgi/Glen integral used by `dprodnormal()` (conditional means and the partial correlation are computed per x).

Usage

```
dprodnormal3(x, mean, cov, log = FALSE)
```

Arguments

`x`	Numeric vector of quantiles.
`mean`	Numeric vector of means of length 3.
`cov`	3x3 covariance matrix; must be positive definite.
`log`	Logical; return the log density.

Details

The density has a logarithmic singularity at exactly `0` (returned as `Inf`), as for `dprodnormal()`. Unlike `pprodnormal3()`, zero-variance components are not supported: a point mass has no density.

Value

Numeric vector of densities, the length of `x`.

References

Gaunt, R. E. (2022). On the product of three normal random variables. See also the references of `dprodnormal()`.

See Also

`pprodnormal3()` for the CDF, `dprodnormal()` for two factors.

Examples

```
Sigma <- matrix(c(1, .3, .2, .3, 1, .1, .2, .1, 1), 3)
dprodnormal3(c(0.1, 0.5, 1), mean = c(0.5, 0.4, 0.3), cov = Sigma)
```

mbco *Model-based Constrained Optimization (MBCO) Chi-squared Test*

Description

This function computes asymptotic MBCO chi-squared test for a smooth function of model parameters including a function of indirect effects.

Usage

```
mbco(h0, h1, ...)
```

Arguments

h0	A fitted restricted OpenMx model or medfit SEMFit under the null hypothesis.
h1	A fitted full model of the same class as h0, under the alternative hypothesis.
...	Additional arguments.

Details

Two medfit SEMFits (medfit >= 0.8.0). Supply the already-fitted restricted model as h0 and the full model as h1. Only type = "asymp" is supported; alpha is validated as for OpenMx. The adapter compares the fits through stats::anova(h1, h0) and uses the degrees-of-freedom difference. It never refits. Bootstrap types, mixed model classes, OpenMx-only fitting controls (R, checkHess, checkSE, optim, precision) and unused dots are rejected. This route does not require OpenMx.

medfit checks convergence, identical data and observed variables, unweighted fits, and parameter-table nesting. The analyst must establish substantive nesting and a valid chi-square reference distribution; arbitrary nonlinear restrictions do not necessarily have one. An active tested bound or inequality at the null causes an error here because the ordinary chi-square p-value is invalid. Other medfit warnings propagate.

For a product-zero null, medfit::fit_sem() with a*b == 0 fits one branch per factor from clean starts and selects the smallest discrepancy. Pass that fitted null and the full fit to mbco(). medfit is optional: this route needs 0.8.0 or later (GitHub/r-universe); existing ci() integration needs only 0.2.0. No change to the CRAN dependency minimum is implied.

OpenMx comparisons require a finite nonnegative observed statistic and a positive degrees-of-freedom difference: supply the restricted model as h0 and the full model as h1. Equal-df and reversed pairs error before bootstrap sampling. These checks do not establish substantive model nesting.

OpenMx bootstrap methods restore global options on success and error. Set a seed before calling for reproducibility; bootstrap sampling consumes the current RNG stream. The semiparametric method requires positive-definite sample and null covariances. The parametric method simulates each replicate from the fitted h0 with OpenMx::mxGenerateData() (the mvtnorm package is required) and fits h0 and h1 to it in a loop the package runs itself, under the same rules as the semiparametric method; it needs a model with a single raw data set. For both bootstrap types a

replicate whose fits did not converge is left out of the null distribution, and the dropped slot of the result counts them. If more than 5% of the R replicates are dropped, the call stops with an error of class RMediation_bootstrap_too_many_dropped (with fewer than 20 replicates no replicate may be dropped). Nothing else is dropped: a replicate with an unrepairable constraint violation (class RMediation_bootstrap_infeasible), a non-finite statistic, or an h1 fit worse than its h0 fit stops the call. The p-value is the proportion of the kept replicates, R - dropped of them, whose statistic is strictly above the observed one, mean(null > observed) over the kept replicates. There is no (1 + k)/(R + 1) correction, so it can be exactly 0 when no kept replicate exceeds the observed statistic.

Testing an OpenMx product null ($ab = 0$). The null $ab = 0$ holds when $a = 0$ or $b = 0$. A single nonlinear constraint such as ind == 0 can converge to either branch, and the optimizer may return the branch with the larger statistic, which gives a wrong test. mbco_product_null() avoids this: give it the fitted full model and the path labels. By hand, fit one linear null per path (a == 0 and b == 0), run mbco() on each, and report the smaller statistic with 1 degree of freedom (see the example). The smaller statistic is the exact constrained likelihood-ratio statistic for $ab = 0$. Its chi-square(1) p-value is conservative when both paths are 0 (an intersection-union test; standard theory, not verified here).

Build each h0 from the unfitted model. A constraint refit from the fitted model's estimates can stop at an infeasible point and return a statistic of 0.

Constraint violations and the automatic refit (OpenMx). OpenMx accepts a constrained fit whose constraints are violated by up to its "Feasibility tolerance" (default 0.05), which can understate the statistic (for example 0 where the exact value is 0.4745). Before comparing, mbco() evaluates every equality and inequality constraint of h0 and h1 at the fitted solution. A constraint counts as violated when its scaled violation, |lhs - rhs| / max(1, |lhs|, |rhs|) for equalities and the wrong-side gap for inequalities, exceeds 1e-6. Constraints with other operators are skipped; a constraint that cannot be evaluated is an error naming it.

When h0 is violated, all three types refit it once from its own estimates with "Feasibility tolerance" set to 1e-8, then use the refit model for the observed statistic and the bootstrap, and warn with class RMediation_h0_refit (statistic before and after). The refit slot of the result records tolerance, statistic_before and statistic_after, and is NULL when no refit was needed. Your own model object is not modified. This is the one case where the returned statistic differs from OpenMx::mxCompare() on your fits. If the refit errors, does not converge or leaves the constraint violated, mbco() stops with an error of class RMediation_h0_refit_failed that names the constraint and the reason, before any bootstrap work and for every type: no statistic is returned from a fit known to be invalid. Refit yourself with mxOption(model, "Feasibility tolerance", 1e-8) or use mbco_product_null() for a product null. h1 is checked and warned about (RMediation_constraint_violation) but never refit. Both bootstrap types ("semi" and "parametric") apply the same checks to every replicate: a replicate whose h0 or h1 fit does not converge stops the call at the first one (6 of 100 runs in a simulation with R = 200); a replicate null fit that violates its constraint is refit as above, and one that cannot be repaired stops the call with an error of class RMediation_bootstrap_infeasible instead of entering the null distribution. (Before 1.8.0 type = "parametric" delegated its bootstrap to OpenMx::mxCompare(boot = TRUE), which does neither.)

When h0 came back identical to h1. A refit whose statistic_before is about 0 (|statistic| < 1e-6, h0 and h1 have the same -2LL although a constraint of h0 was violated) also raises a warning of class RMediation_constraint_not_enforced, right after RMediation_h0_refit. It means the constraint may not have been enforced. Two causes give this symptom. OpenMx (2.19.1 and later) silently drops an equality constraint whose Jacobian row is zero at the starting values, so b1 * b2

== 0 started at b1 = b2 = 0 returns the unconstrained fit (OpenMx/OpenMx#433); a violation within the feasibility tolerance looks the same. A fitted h0 keeps no starting values, so mbco() cannot tell the causes apart and the warning does not name either one as the cause. The reported statistic is the refit one. For a product null, skip the problem: mbco_product_null() fits exact single-path nulls and never relies on a nonlinear constraint. The refit and this warning apply to the observed statistic for every type. Both bootstrap types ("semi" and "parametric") repeat the refit on every bootstrap replicate, and a replicate null fit that cannot be repaired stops the call (see above).

Wrong-branch check (OpenMx). A feasible, converged fit of a product null such as a*b == 0 can still sit on the larger-statistic branch (for example a = 0 when b = 0 fits far better), which neither the violation guard nor the refit can see. After the refit, mbco() therefore checks every equality constraint of the h0 it will use. It finds the free parameters the constraint depends on (by perturbing each at the full model's estimates, without parsing the formula), drops the constraint, fixes each of them at 0 in turn, and refits. If a converged fit that still satisfies the constraint has a -2LL lower than h0's by more than 1e-4, mbco() warns with class RMediation_constraint_branch, naming the constraint, the parameter and the gap. It runs once per call for every type, costs one extra fit per involved parameter, and skips (with a message of class RMediation_constraint_branch_skipped) a constraint that depends on more than 5 parameters. The check is advisory: it never changes the result and a failure inside it never fails the call. It only searches fits that fix one involved parameter at 0, so it can show that h0 is not the constrained optimum but not that it is. Use mbco_product_null() to avoid the problem.

Value

An object of class MBCOResult containing

statistic	asymptotic chi-squared test statistic value
df	chi-squared df
p_value	chi-squared p-value computed based on the method specified by the argument type
type	The type of test performed
provenance	what produced the result: the RMediation, OpenMx (if loaded), medfit (if loaded) and R versions, type, the bootstrap size R, RNGkind() and the run time. The random seed is not recorded, because it cannot be recovered after the fact; call set.seed() yourself and report it

Author(s)

Davood Tofighi <dtofighi@gmail.com>

Examples

```
if (requireNamespace("medfit", quietly = TRUE) &&
    utils::packageVersion("medfit") >= "0.9.0") {
  d <- as.data.frame(memory_exp)
  d$x <- as.numeric(d$x) - 1
  syntax <- paste("repetition ~ a1*x", "imagery ~ a2*x",
                  "recall ~ cp*x + b1*repetition + b2*imagery", sep = "\n")
  full <- medfit::fit_sem(syntax, d, fixed_x = FALSE, meanstructure = NULL)
  null <- medfit::fit_sem(paste(syntax, "a1*b1 == 0", sep = "\n"), d,
```

```
                          fixed_x = FALSE, meanstructure = NULL)
  mbco(null, full) # statistic 0.0831, df 1
}
## Not run:
data(memory_exp)
memory_exp <- as.data.frame(memory_exp) # mxData() does not accept tibbles
memory_exp$x <- as.numeric(memory_exp$x) - 1 # manually creating dummy codes
endVar <- c("repetition", "imagery", "recall") # endogenous variables (not "x")
manifests <- c("x", "repetition", "imagery", "recall")
full_model <- OpenMx::mxModel(
  "memory_example",
  type = "RAM",
  manifestVars = manifests,
  OpenMx::mxPath(
    from = "x",
    to = endVar,
    arrows = 1,
    free = TRUE,
    values = .2,
    labels = c("a1", "a2", "cp")
  ),
  OpenMx::mxPath(
    from = "repetition",
    to = "recall",
    arrows = 1,
    free = TRUE,
    values = .2,
    labels = "b1"
  ),
  OpenMx::mxPath(
    from = "imagery",
    to = "recall",
    arrows = 1,
    free = TRUE,
    values = .2,
    labels = "b2"
  ),
  OpenMx::mxPath(
    from = manifests,
    arrows = 2,
    free = TRUE,
    values = .8
  ),
  OpenMx::mxPath(
    from = "one",
    to = manifests,
    arrows = 1,
    free = TRUE,
    values = .1
  ),
  OpenMx::mxAlgebra(a1 * b1, name = "ind1"),
  OpenMx::mxAlgebra(a2 * b2, name = "ind2"),
  OpenMx::mxCI("ind1", type = "both"),
```

```
  OpenMx::mxCI("ind2", type = "both"),
  OpenMx::mxData(observed = memory_exp, type = "raw")
)
## H0: a1*b1 = 0 holds if a1 = 0 or b1 = 0. Fit one linear null per path and
## report the smaller statistic (what medfit::mbco_sem() automates). A single
## nonlinear null ind1 == 0 can converge to the wrong branch (rmediation#35).
## Build each h0 from the unfitted model: a constraint refit from the fitted
## model's estimates can stop at an infeasible point.
null_a1 <- OpenMx::mxModel(
  model = full_model,
  name = "Null a1 = 0",
  OpenMx::mxConstraint(a1 == 0, name = "a1_eq0_constr")
)
null_b1 <- OpenMx::mxModel(
  model = full_model,
  name = "Null b1 = 0",
  OpenMx::mxConstraint(b1 == 0, name = "b1_eq0_constr")
)
full_model <- OpenMx::mxTryHard(full_model, checkHess = FALSE, silent = TRUE)
null_a1 <- OpenMx::mxTryHard(null_a1, checkHess = FALSE, silent = TRUE)
null_b1 <- OpenMx::mxTryHard(null_b1, checkHess = FALSE, silent = TRUE)
res_a1 <- mbco(null_a1, full_model)
res_b1 <- mbco(null_b1, full_model)
res_a1
res_b1
## Test of H0: a1*b1 = 0 (df = 1): the smaller of the two statistics
min(res_a1@statistic, res_b1@statistic)
## The same test in one call: fixes each path at 0 in a copy of the fitted
## full model and returns the binding branch (here b1, statistic 0.0831)
mbco_product_null(full_model, c("a1", "b1"))

## End(Not run)
```

`mbco_product_null` *MBCO test of a product null (a zero-product indirect effect)*

Description

Tests the null that a product of paths is zero (`a * b = 0`, or a serial `a1 * a2 * b = 0`) with OpenMx. The null holds when any single path is 0, so it is a union of single-path nulls. `mbco_product_null()` fits one null per path by fixing that path at 0 in a copy of the fitted full model, runs `mbco()` on each, and returns the result of the branch with the largest p-value, because a test of a union rejects only if every single-path null is rejected (an intersection-union test). For `type = "asymp"` this is the smaller statistic with 1 degree of freedom.

Usage

```
mbco_product_null(h1, paths, ...)
```

Arguments

h1	A fitted full OpenMx model.
paths	Character vector of two or more labels of free parameters of h1, the paths whose product is tested.
...	Passed to mbco() (type, R, alpha, optim, ...).

Details

A single nonlinear constraint such as ind == 0 can converge to the branch with the larger statistic and give a wrong test; see mbco(), which warns when it detects this. Fixing a parameter is exact, so this function avoids that and the feasibility tolerance problem. The p-value is conservative when more than one path is 0 (standard intersection-union theory, not verified here). Each branch fit must converge (OpenMx status code 0 or 1); otherwise the function stops with an error naming the branch. h1 must have converged too (class RMediation_h1_nonconverged otherwise): the null models start from its estimates and every statistic is measured against its fit, so a p-value from an unconverged h1 would not be reliable.

Value

An MBCOResult for the binding branch, with the label of its single-path null in the branch slot.

MBCOResult *MBCO Result Class*

Description

A class representing the results of a Model-Based Constrained Optimization (MBCO) test.

Usage

```
MBCOResult(
  statistic = integer(0),
  df = integer(0),
  p_value = integer(0),
  type = character(0),
  refit = NULL,
  branch = NULL,
  dropped = NULL,
  provenance = NULL
)
```

Arguments

statistic	Numeric test statistic value.
df	Numeric degrees of freedom.
p_value	Numeric p-value.

type	Character string indicating the type of test.
refit	NULL (default) when no refit happened, or a list with tolerance, statistic_before and statistic_after (all single finite numbers) when mbco() refit a violated h0 at a tighter feasibility tolerance. print() and summary() show a note when it is set.
branch	NULL (default), or the label of the single-path null that gave the reported result when the result comes from mbco_product_null().
dropped	NULL (default) for a result without a bootstrap, or a single non-negative whole number: the count of bootstrap replicates whose fits did not converge and were left out of the null distribution (0 when none were). The p-value is taken over the remaining R - dropped replicates.
provenance	NULL (default), or a list recording what produced the result: rmediation, openmx and medfit (package versions; NULL for a package not loaded), r (the R version), type, R (bootstrap replicates, NULL without a bootstrap), rng_kind (from RNGkind()) and time. mbco() fills it. The seed is not recorded: it cannot be recovered after the fact, so call set.seed() yourself and report it.

Examples

```
res <- MBCOResult(statistic = 4.2, df = 1, p_value = 0.04, type = "asymp")
```

---

medci *Confidence Interval for the Mediated Effect*

Description

Produces confidence intervals for the mediated effect and the product of two normal random variables

Usage

```
medci(
  mu.x,
  mu.y,
  se.x,
  se.y,
  rho = 0,
  alpha = 0.05,
  type = "dop",
  plot = FALSE,
  plotCI = FALSE,
  n.mc = 1e+05,
  ...
)
```

Arguments

mu.x	mean of x
mu.y	mean of y
se.x	standard error (deviation) of x
se.y	standard error (deviation) of y
rho	correlation between x and y, where -1 <rho < 1. The default value is 0.
alpha	significance level for the confidence interval. The default value is .05.
type	method used to compute confidence interval. It takes on the values "dop" (default), "MC", "asymp" or "all"
plot	when TRUE, plots the distribution of n.mc data points from the distribution of product of two normal random variables using the density estimates provided by the function density. The default value is FALSE. Drawing the sample consumes random numbers, so a later Monte Carlo call in the same session differs from the same call made without plot = TRUE.
plotCI	when TRUE, overlays a confidence interval with error bars on the plot for the mediated effect. Note that to obtain the CI plot, one must also specify plot="TRUE". The default value is FALSE.
n.mc	Monte Carlo sample size when type="MC" (default: 1e5). Larger values provide more precision but take longer to compute. For most applications, 1e5 provides adequate precision (MC error < 0.05% of SE).
...	additional arguments to be passed on to the function.

Details

This function returns a confidence interval at significance level α for the mediated effect (product of two normal random variables). To obtain a confidence interval using a specific method, the argument type should be specified. The default is type="dop", which uses the code we wrote in R to implement the distribution of product of the coefficients method described by Meeker and Escobar (1994) to evaluate the CDF of the distribution of product. type="MC" uses the Monte Carlo approach to compute the confidence interval (Tofighi & MacKinnon, 2011). type="asymp" produces the asymptotic normal confidence interval. Note that except for the Monte Carlo method, the standard error for the indirect effect is based on the analytical results by Craig (1936):

$$SE = \sqrt{\sigma_y^2\mu_x^2 + \sigma_x^2\mu_y^2 + 2\mu_x\mu_y\rho\sigma_x\sigma_y + \sigma_x^2\sigma_y^2 + \sigma_x^2\sigma_y^2\rho^2}$$

where σ_x and σ_y are the standard errors of x and y, respectively. In addition, the estimate of the indirect effect is $\mu_x\mu_y + \sigma_{xy}$, where σ_{xy} is the covariance between x and y. The argument type="all" prints confidence intervals using all available methods.

Value

A vector of lower confidence limit and upper confidence limit. When type is "prodclin" (default), "DOP", "MC" or "asymp", medci returns a list that contains:

CI	a vector of lower and upper confidence limits (at significance level α),

`Estimate`	a point estimate of the quantity of interest,
`SE`	standard error of the quantity of interest,
`MC Error`	When `type="MC"`, standard error of the simulated mean, `SE / sqrt(n.mc)`.

Note that when `type="all"`, `medci` returns a list of *four* objects, each of which a list that contains the results produced by each method as described above.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

References

Craig, C. C. (1936). On the frequency function of xy. *The Annals of Mathematical Statistics*, **7**, 1–15.

MacKinnon, D. P., Fritz, M. S., Williams, J., and Lockwood, C. M. (2007). Distribution of the product confidence limits for the indirect effect: Program PRODCLIN. *Behavior Research Methods*, **39**, 384–389.

Meeker, W. and Escobar, L. (1994). An algorithm to compute the CDF of the product of two normal random variables. *Communications in Statistics: Simulation and Computation*, **23**, 271–280.

Tofighi, D. and MacKinnon, D. P. (2011). RMediation: An R package for mediation analysis confidence intervals. *Behavior Research Methods*, **43**, 692–700. doi:10.3758/s134280110076x

See Also

`qprodnormal` `pprodnormal` `ci` `RMediation-package`

Examples

```
## Example 1
res <- medci(
  mu.x = .2, mu.y = .4, se.x = 1, se.y = 1, rho = 0, alpha = .05,
  type = "dop", plot = TRUE, plotCI = TRUE
)
## Example 2
res <- medci(
  mu.x = .2, mu.y = .4, se.x = 1, se.y = 1, rho = 0,
  alpha = .05, type = "all", plot = TRUE, plotCI = TRUE
)
```

`memory_exp` *Memory Experiment Data Description from MacKinnon et al., 2018*

Description

Data were obtained from eight replicated experiments. The data were collected on the first day of class as part of the first Dr. MacKinnon's (2018) classroom teaching. The pedagogical value of the experiment was that students would have first-hand knowledge of the experiment thereby increasing their understanding of course concepts. Permission to use the data was obtained from the university Institutional Review Board.

Usage

```
data(memory_exp)
```

Format

A data frame with 369 rows and 5 variables:

study Replication ID, ranges from 1 to 8

repetition Use of repetition rehearsal technique, ranges from 1 to 12

recall Total words recalled out of 20 words, ranges from 3 to 20

imagery Use of imagery rehearsal technique on a 1 to 9 scale

x A factor with two levels: "repetition" (primary rehearsal) or "imagery" (secondary rehearsal)

Note

If you use the data set, please cite the original article by MacKinnon et al. (2018) cited below.

Source

doi:10.1037/met0000174.supp

References

MacKinnon, D. P., Valente, M. J., & Wurpts, I. C. (2018). Benchmark validation of statistical models: Application to mediation analysis of imagery and memory. Psychological Methods, 23, 654–671. doi:10.1037/met0000174

`ml_omitted_confounder` *Sensitivity of a Multilevel Indirect Effect to an Omitted Confounder*

Description

Sensitivity analysis for an omitted confounder `Z` of the mediator-outcome relation in a two-level 1-1-1 mediation model (treatment `A`, mediator `M` and outcome `Y`, all measured at level 1), after Tofighi and Kelley (2016). For each level the unmeasured, standardized confounder is described by its correlations with the treatment, mediator and outcome (`r_za`, `r_zm`, `r_zy`), the sensitivity parameters. The adjusted paths are `a = a* - d_MZ * d_ZA` and `b = b* - d_YZ * s_ZM`, where `a*` and `b*` are the supplied (unadjusted) estimates, `d_ZA` is the slope of `Z` on `A`, `d_MZ` is the coefficient of `Z` in the regression of `M` on `A` and `Z`, `s_ZM` is the coefficient of `M` in the regression of `Z` on `A` and `M`, and `d_YZ` is the coefficient of `Z` in the regression of `Y` on `A`, `M` and `Z`. These coefficients are obtained from the observed covariance matrix of `A`, `M` and `Y` together with the sensitivity correlations. The adjusted indirect effect is `a * b`.

Usage

```
ml_omitted_confounder(
  between = NULL,
  within = NULL,
  r_za = c(0, 0.1, 0.3),
  r_zm = seq(-0.5, 0.5, 0.01),
  r_zy = seq(-0.5, 0.5, 0.01),
  screen = c("pd", "none"),
  var_order = c("A", "M", "Y")
)
```

Arguments

`between`, `within`	Lists describing the between-cluster and within-cluster levels (see Details). At least one is required.
`r_za`	Correlation(s) of the confounder with the treatment: a numeric vector, or a list with elements `between` and/or `within`.
`r_zm`, `r_zy`	Numeric vectors of correlations of the confounder with the mediator and the outcome.
`screen`	`"pd"` (default) keeps only admissible points; `"none"` keeps every grid point.
`var_order`	Character vector, a permutation of `c("A", "M", "Y")`, giving the row and column order of the supplied matrices.

Details

Inputs. Each level is a list with the unadjusted estimates `a` (the `A` to `M` path) and `b` (the `M` to `Y` path, adjusting for `A`) and the observed covariance matrix of `A`, `M` and `Y` at that level, given either as `cov` or as `cor` together with `sd`. Rows and columns follow `var_order`. If the matrix has dimnames they must equal `var_order`; otherwise they are absent. The supplied `a` and `b` need not equal the

values implied by the covariance matrix (the published example does not). The function subtracts a covariance-derived bias term from the supplied estimates, so the result is the supplied estimate adjusted by that term, not a refit.

Assumption. As in the published illustration, the covariance between the level-2 random slopes of the a and b paths is taken to be zero; it is not estimated.

Screen. A grid point is admissible when the augmented 4 by 4 correlation matrix of A, M, Y and Z is positive definite, equivalently `r' R^-1 r < 1` with `r = (r_za, r_zm, r_zy)` and R the observed correlation matrix; this quantity is returned as r2 in the grid. With `screen = "pd"` (the default) points that are not admissible get NA for the adjusted paths. **The default does not reproduce the published tables:** the endpoints printed by Tofighi and Kelley (2016) are computed over the whole grid, which corresponds to `screen = "none"`. With the published example, the default shrinks the between-level ranges.

Sensitivity grids. r_za may be a single numeric vector (used for every supplied level) or a list with elements between and/or within. A level named in the list must have data, and a level with data must be named. r_zm and r_zy are the grid of confounder correlations with the mediator and outcome. All correlations must be finite with absolute value below 1. Points where a regression block is numerically singular return NA.

Value

An MLOmittedConfounder object with properties grid (one row per level and grid point: level, r_za, r_zm, r_zy, a_adj, b_adj, ab_adj, r2, admissible), ranges (minimum and maximum adjusted indirect effect and where they occur, per level and r_za, with the number of admitted points), unadjusted, screen, inputs and call. It has print, summary, plot and as.data.frame methods.

References

Tofighi, D., & Kelley, K. (2016). Assessing omitted confounder bias in multilevel mediation models. *Multivariate Behavioral Research, 51*(1), 86-105. doi:10.1080/00273171.2015.1105736

Examples

```
# Synthetic between-cluster level: correlation matrix of A, M, Y and SDs
R <- matrix(c(1, .3, .2, .3, 1, .4, .2, .4, 1), 3)
res <- ml_omitted_confounder(
  between = list(a = 0.3, b = 0.4, cor = R, sd = c(1, 1, 1)),
  r_za = c(0, 0.1)
)
res
summary(res)
head(as.data.frame(res))
plot(res, r_za = 0.1)
```

pMC *Probability (percentile) for the Monte Carlo Sampling Distribution of a nonlinear function of coefficients estimates*

Description

This function returns a probability corresponding to the quantile q.

Usage

```
pMC(q, mu, Sigma, quant, lower.tail = TRUE, n.mc = 1e+05, ...)
```

Arguments

q	quantile
mu	a vector of means (e.g., coefficient estimates) for the normal random variables. A user can assign a name to each mean value, e.g., mu=c(b1=.1,b2=3); otherwise, the coefficient names are assigned automatically as follows: b1,b2,....
Sigma	either a covariance matrix or a vector that stacks all the columns of the lower triangle variance–covariance matrix one underneath the other.
quant	quantity of interest, which is a nonlinear/linear function of the model parameters. Argument quant is a formula that **must** start with the symbol "tilde" (~): e.g., ~b1*b2*b3*b4. The names of coefficients must conform to the names provided in the argument mu or to the default names, i.e., b1,b2,....
lower.tail	logical; if TRUE (default), the probability is $P[quant < q]$; otherwise, $P[quant > q]$
n.mc	Monte Carlo sample size (default: 1e5). Larger values provide more precision but take longer to compute.
...	additional arguments.

Value

scalar probability value.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

References

Tofighi, D. and MacKinnon, D. P. (2011). RMediation: An R package for mediation analysis confidence intervals. *Behavior Research Methods*, **43**, 692–700. doi:10.3758/s13428-011-0076-x

Examples

```
pMC(.2,
  mu = c(b1 = 1, b2 = .7, b3 = .6, b4 = .45), Sigma = c(.05, 0, 0, 0, .05, 0, 0, .03, 0, .03),
  quant = ~ b1 * b2 * b3 * b4
)
```

`pprodnormal` *Percentile for the Distribution of Product of Two Normal Variables*

Description

Generates percentiles (100 based quantiles) for the distribution of product of two normal random variables and the mediated effect

Usage

```
pprodnormal(
  q,
  mu.x,
  mu.y,
  se.x = 1,
  se.y = 1,
  rho = 0,
  lower.tail = TRUE,
  type = "dop",
  n.mc = 1e+05
)
```

Arguments

`q`	quantile or value of the product; a numeric vector returns one probability per element (for `type="MC"` all elements share one simulated sample)
`mu.x`	mean of x
`mu.y`	mean of y
`se.x`	standard error (deviation) of x
`se.y`	standard error (deviation) of y
`rho`	correlation between x and y, where -1 <`rho` < 1. The default value is 0.
`lower.tail`	logical; if `TRUE` (default), the probability is $P(XY \leq q)$; otherwise, $P(XY > q)$
`type`	method used to compute confidence interval. It takes on the values `"dop"` (default), `"MC"`, or `"all"`
`n.mc`	Monte Carlo sample size when `type="MC"` (default: 1e5). Larger values provide more precision but take longer to compute.

Details

This function returns the percentile (probability) and the associated error for the distribution of product of mediated effect (two normal random variables). To obtain a percentile using a specific method, the argument `type` should be specified. The default method is `type="dop"`, which is based on the method described by Meeker and Escobar (1994) to evaluate the CDF of the distribution of product of two normal random variables. `type="MC"` uses the Monte Carlo approach (Tofighi & MacKinnon, 2011). `type="all"` prints percentiles using all three options. For the method `type="dop"`, the error is the modulus of absolute error for the numerical integration (for more information see Meeker and Escobar, 1994). For `type="MC"`, the error refers to the Monte Carlo error.

Value

A numeric vector of probabilities, one per element of `q`, for `type="dop"` or `"MC"`; for `type="all"` a list of two such vectors (`"Distribution of Product"` and `"Monte Carlo"`).

Author(s)

Davood Tofighi <dtofighi@gmail.com>

References

Tofighi, D. and MacKinnon, D. P. (2011). RMediation: An R package for mediation analysis confidence intervals. *Behavior Research Methods*, **43**, 692–700. doi:10.3758/s13428-011-0076-x

See Also

`medci` `RMediation-package`

Examples

```
pprodnormal(q = 0, mu.x = .5, mu.y = .3, se.x = 1, se.y = 1, rho = 0, type = "all")
```

`pprodnormal3` *Cumulative Distribution Function for the Product of Three Normal Variables*

Description

Computes `P(X1 * X2 * X3 <= q)` where `(X1, X2, X3)` follows a trivariate normal distribution with mean vector `mean` and covariance matrix `cov`. `Z` is marginalized analytically through the conditional Gaussian structure `Z | (X, Y)`, and the remaining two-dimensional integral is evaluated with adaptive cubature (dimension reduction; see Tofighi, 2026).

Usage

```
pprodnormal3(
  q,
  mean,
  cov,
  method = c("gauss", "hcubature"),
  tol = 1e-06,
  nodes = NULL,
  diagnostics = FALSE,
  lower.tail = TRUE
)

p_prod3(...)
```

Arguments

q	Numeric vector of quantiles (finite, no missing values). A single value takes the scalar path unchanged.
mean	Numeric vector of means of length 3.
cov	3x3 covariance matrix. Must be positive-definite, except that a single zero-variance component is permitted (the problem then reduces to the two-variable product-normal case). Two zero-variance components give the exact scaled-normal CDF, and three give a point mass at `prod(mean)` (CDF 1 at and above it). Perfect correlation between two components (`|rho| = 1`) or an indefinite matrix is rejected.
method	Integration method. `"gauss"` (default) is tensor-product Gauss-Legendre quadrature on a domain partitioned at the coordinate axes, with the node count escalated until successive rules agree to `tol`. `"hcubature"` is the adaptive method used before version 1.7.0; it is retained for cross-checking and backward comparison, but is **not recommended** when the `(X, Y)` block is ill-conditioned (see Note).
tol	Numeric tolerance passed to the integration routine; must be strictly positive.
nodes	Optional integer: force a fixed number of Gauss-Legendre nodes per dimension instead of escalating adaptively. Ignored when `method = "hcubature"`. Mainly useful for reproducing a specific rule or for cross-checking a result at two node counts.
diagnostics	Logical. If `TRUE`, the returned value carries `"error"` and `"nodes"` attributes. Defaults to `FALSE` so the return value stays a bare numeric for existing callers. What `"error"` means depends on `method`, and the two are **not comparable**: • `method = "gauss"`: the gap between the last two quadrature rules, a genuine convergence estimate for discretization error. It does not see truncation error, which is instead bounded by construction. It is `NA` when `nodes` is supplied, because a single fixed rule produces no successive-rule gap to measure; to check convergence at a fixed rule, evaluate at `nodes` and `2 * nodes` and compare.

- `method = "hcubature"`: that integrator's *own* reported error estimate, always present and never `NA` (`nodes` is ignored on this path). Treat it with suspicion. It is satisfied at exactly the configurations where the result is badly wrong (see Note), which is why it cannot be used to detect the failure this method was replaced over.

`lower.tail` Logical; if `TRUE` (default) probabilities are `P(X1 * X2 * X3 <= q)`, otherwise `P(X1 * X2 * X3 > q)`. The upper tail is integrated directly, not computed as `1 - p`, so it stays accurate far out where `1 - p` would be lost to rounding. The accuracy is still the absolute tolerance `tol`, so for a tail probability of size `p` use `tol` well below `p` (for example `tol = 1e-14` for a probability near 1e-10).

`...` Additional arguments (unused; present for the `p_prod3` deprecated alias).

Value

Probability `P(X1 * X2 * X3 <= q)` in `[0, 1]`: a numeric scalar for a single `q`, and a numeric vector of the same length (and names) as `q` otherwise, in the order given. With `diagnostics = TRUE` and more than one `q`, the `"error"` and `"nodes"` attributes are vectors with one entry per `q` (`NA` where there is nothing to report, for example a point mass).

Note

`p_prod3()` is a superseded alias for `pprodnormal3()`, kept for backward compatibility; use `pprodnormal3()` in new code.

Note

A point mass (zero variances) has the right-continuous CDF: for a product identically equal to `c`, the result is `1` for `q >= c` and `0` below. Before 1.8.0 an atom at exactly `0` returned `0.5` at `q == 0`.

Before version 1.7.0 the default integrator was `"hcubature"`, which returns materially wrong values on ill-conditioned covariance matrices – at `rho = 0.999` the error reaches 24% at `q = 0.5` and 92% in relative terms in the lower tail, silently and with no diagnostic. The default is now `"gauss"`, which agrees with `"hcubature"` to roughly `1e-6` on well-conditioned input and remains accurate where `"hcubature"` fails. Results from the two methods therefore differ slightly on benign input and substantially on ill-conditioned input; the latter is the correction.

See Also

`pprodnormal()` and `qprodnormal()` for the two-variable product- normal CDF/quantile; ProductNormal3 for the corresponding S7 class.

Examples

```
Sigma <- diag(3)
pprodnormal3(q = 0, mean = c(0, 0, 0), cov = Sigma)
# the upper tail, integrated directly (far out, 1 - p would be lost to rounding)
pprodnormal3(30, mean = c(1, 1, 1), cov = diag(3), tol = 1e-10, lower.tail = FALSE)
# several quantiles at once
pprodnormal3(q = c(-1, 0, 1), mean = c(0, 0, 0), cov = Sigma)
```

ProductNormal *ProductNormal Class*

Description

Represents the distribution of the product of normal random variables.

Usage

```
ProductNormal(mu = integer(0), Sigma = integer(0))
```

Arguments

mu	Numeric vector of means.
Sigma	Covariance matrix.

Examples

```
pn <- ProductNormal(mu = c(0.5, 0.3), Sigma = diag(2) * 0.1)
```

ProductNormal2 *Enhanced ProductNormal constructor with better validation*

Description

Enhanced ProductNormal constructor with better validation

Usage

```
ProductNormal2(mu, Sigma, validate = TRUE)
```

Arguments

mu	Numeric vector of means
Sigma	Covariance matrix
validate	Whether to run additional validation (default: TRUE)

Examples

```
pn <- ProductNormal2(mu = c(0.5, 0.3), Sigma = diag(2) * 0.1)
```

ProductNormal3 *ProductNormal3 Class*

Description

Represents the distribution of the product of three normal random variables. Intended for sequential indirect effects of the form `a1 * a2 * b`, where `(a1, a2, b)` is asymptotically trivariate normal.

Usage

```
ProductNormal3(mu = integer(0), Sigma = integer(0), method = "gauss")
```

Arguments

`mu`	Numeric vector of means of length 3: `c(a1_hat, a2_hat, b_hat)`.
`Sigma`	3x3 asymptotic covariance matrix of `(a1, a2, b)`.
`method`	Integration method passed to `pprodnormal3()`: `"gauss"` (default) or `"hcubature"`. See the Note in `pprodnormal3()` for why `"gauss"` became the default in version 1.7.0.

Examples

```
obj <- ProductNormal3(
  mu = c(0.5, 0.3, 0.2),
  Sigma = diag(3)
)
obj
cdf(obj, q = 1)
## Not run:
confint(obj, level = 0.95)

## End(Not run)
```

qMC *Quantile for the Monte Carlo Sampling Distribution of a nonlinear function of coefficients estimates*

Description

This function returns a quantile corresponding to the probability p.

Usage

```
qMC(p, mu, Sigma, quant, n.mc = 1e+05, ...)
```

Arguments

p	probability.
mu	a vector of means (e.g., coefficient estimates) for the normal random variables. A user can assign a name to each mean value, e.g., `mu=c(b1=.1,b2=3)`; otherwise, the coefficient names are assigned automatically as follows: `b1,b2,....`
Sigma	either a covariance matrix or a vector that stacks all the columns of the lower triangle variance–covariance matrix one underneath the other.
quant	quantity of interest, which is a nonlinear/linear function of the model parameters. Argument `quant` is a formula that **must** start with the symbol "tilde" (`~`): e.g., `~b1*b2*b3*b4`. The names of coefficients must conform to the names provided in the argument `mu` or to the default names, i.e., `b1,b2,....`
n.mc	Monte Carlo sample size (default: 1e5). Larger values provide more precision but take longer to compute.
...	additional arguments.

Value

scalar quantile value.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

References

Tofighi, D. and MacKinnon, D. P. (2011). RMediation: An R package for mediation analysis confidence intervals. *Behavior Research Methods*, **43**, 692–700. doi:10.3758/s13428-011-0076-x

Examples

```
qMC(.05,
  mu = c(b1 = 1, b2 = .7, b3 = .6, b4 = .45), Sigma = c(.05, 0, 0, 0, .05, 0, 0, .03, 0, .03),
   quant = ~ b1 * b2 * b3 * b4
)
```

---

qprodnormal *Quantile for the Distribution of Product of Two Normal Variables*

Description

Generates quantiles for the distribution of product of two normal random variables

Usage

```
qprodnormal(
  p,
  mu.x,
  mu.y,
  se.x,
  se.y,
  rho = 0,
  lower.tail = TRUE,
  type = "dop",
  n.mc = 1e+05
)
```

Arguments

p	probability; a numeric vector returns one quantile per element (for type="MC" all elements share one simulated sample)
mu.x	mean of x
mu.y	mean of y
se.x	standard error (deviation) of x
se.y	standard error (deviation) of y
rho	correlation between x and y, where -1 <rho < 1. The default value is 0.
lower.tail	logical; if TRUE (default), the probability is $P(XY \leq q)$; otherwise, $P(XY > q)$
type	method used to compute confidence interval. It takes on the values "dop" (default), "MC", or "all"
n.mc	Monte Carlo sample size when type="MC" (default: 1e5). Larger values provide more precision but take longer to compute.

Details

This function returns a quantile and the associated error (accuracy) corresponding the requested percentile (probability) p of the distribution of product of mediated effect (product of two normal random variables). To obtain a quantile using a specific method, the argument type should be specified. The default method is type="dop", which uses the method described by Meeker and Escobar (1994) to evaluate the CDF of the distribution of product of two normal variables. type="MC" uses the Monte Carlo approach (Tofighi & MacKinnon, 2011). type="all" prints quantiles using all three options. For the method type="dop", the error is the modulus of absolute error for the numerical integration (for more information see Meeker and Escobar, 1994). For type="MC", the error refers to the Monte Carlo error.

Value

A numeric vector of quantiles, one per element of p, for type="dop" or "MC"; for type="all" a list of two such vectors ("Distribution of Product" and "Monte Carlo").

Author(s)

Davood Tofighi <dtofighi@gmail.com>

References

Tofighi, D. and MacKinnon, D. P. (2011). RMediation: An R package for mediation analysis confidence intervals. *Behavior Research Methods*, **43**, 692–700. doi:10.3758/s13428-011-0076-x

See Also

medci RMediation-package

Examples

```
## lower tail
qprodnormal(
  p = .1, mu.x = .5, mu.y = .3, se.x = 1, se.y = 1, rho = 0,
  lower.tail = TRUE, type = "all"
)
## upper tail
qprodnormal(
  p = .1, mu.x = .5, mu.y = .3, se.x = 1, se.y = 1, rho = 0,
  lower.tail = FALSE, type = "all"
)
```

---

qprodnormal3 *Quantile Function for the Product of Three Normal Variables*

Description

Computes the quantile q with P(X1 * X2 * X3 <= q) = p (or > q, for lower.tail = FALSE) where (X1, X2, X3) follows a trivariate normal distribution with mean vector mean and covariance matrix cov. It inverts pprodnormal3() by root-finding, bracketed by the delta-method interval, and is the same routine behind dist_quantile() for a ProductNormal3 object.

Usage

```
qprodnormal3(p, mean, cov, lower.tail = TRUE, tol = 1e-06)
```

Arguments

p	Numeric vector of probabilities in [0, 1], no missing values. 0 and 1 return -Inf and Inf (reversed for lower.tail = FALSE).
mean	Numeric vector of means of length 3.
cov	3x3 covariance matrix, as for pprodnormal3().
lower.tail	Logical; if TRUE (default) p is P(X1 * X2 * X3 <= q), otherwise P(X1 * X2 * X3 > q). The upper tail is inverted through the directly integrated upper tail of pprodnormal3(), so a small upper-tail p is resolved.

`tol`	Positive tolerance used for both the CDF evaluation and the root-finder (default `1e-6`). It is absolute, so for a probability `p` use a `tol` well below `p`; `pprodnormal3()` explains why.

Value

A numeric vector the same length and names as `p`. Three zero-variance components give a point mass, whose quantile is `prod(mean)` for any `p` strictly between 0 and 1.

See Also

`pprodnormal3()` for the CDF, `qprodnormal()` for two factors, `dist_quantile()` for the object interface.

Examples

```
qprodnormal3(c(0.025, 0.5, 0.975), mean = c(1, 1, 1), cov = diag(3))
# an upper-tail probability of 1e-8, resolved with a matching tolerance
qprodnormal3(1e-8, mean = c(1, 1, 1), cov = diag(3), lower.tail = FALSE, tol = 1e-12)
```

`tidy` *Tidy generic function*

Description

Tidy generic function

Usage

```
tidy(x, ...)
```

Arguments

`x`	An object to tidy
`...`	Additional arguments passed to methods

Value

A tibble or data.frame with tidy output

Examples

```
## Not run:
tidy(mbco_result)

## End(Not run)
```

`tidy.logLik` *Creates a data.frame for a log-likelihood object*

Description

Creates a data.frame for a log-likelihood object

Usage

```
## S3 method for class 'logLik'
tidy(x, ...)
```

Arguments

`x`	x A log-likelihood object, typically returned by logLik.
`...`	Additional arguments (not used)

Value

A data.frame with columns:

term The term name

estimate The log-likelihood value

df The degrees of freedom

Author(s)

Davood Tofighi <dtofighi@gmail.com>

See Also

`logLik`

Examples

```
fit <- lm(mpg ~ wt, data = mtcars)
logLik_fit <- logLik(fit)
tidy(logLik_fit)
```

Index

∗ **data**
    memory_exp, 18
∗ **distribution**
    pMC, 21
    qMC, 27
∗ **mediation**
    medci, 15
∗ **regression**
    pMC, 21
    qMC, 27
∗ **sets**
    memory_exp, 18

cdf, 3
ci, 3, *5*, *17*
ci_mediation_data, 4
ci_serial_mediation_data
        *(*ci_mediation_data*)*, 4

data.frame, *32*
density, *16*
dist_quantile, 6
dist_quantile(), *30, 31*
dprodnormal, 6
dprodnormal(), *8*
dprodnormal3, 7

factor, *18*
formula, *21*, *28*

list, *16, 17*
logLik, *32*

mbco, 9
mbco(), *13–15*
mbco_product_null, 13
mbco_product_null(), *10, 11*, *15*
MBCOResult, 14, *14*
medci, 15, *23*, *30*
MediationData, *5*
memory_exp, 18
ml_omitted_confounder, 19

OpenMx::mxGenerateData(), *9*

p_prod3 *(*pprodnormal3*)*, 23
pMC, 21
pprodnormal, *6, 7*, *17*, 22
pprodnormal(), *25*
pprodnormal3, 23
pprodnormal3(), *8*, *27*, *30, 31*
ProductNormal, *5*, 26
ProductNormal2, 26
ProductNormal3, *25*, 27, *30*

qMC, 27
qprodnormal, *6, 7*, *17*, 28
qprodnormal(), *25*, *31*
qprodnormal3, 30

RNGkind(), *15*

set.seed(), *15*

tidy, 31
tidy.logLik, 32

vector, *21*, *28*