Package 'covsim'

October 29, 2026

Type Package

Title VITA, IG, PLSIM and VM Simulation for Given Covariance and Marginals

Version 1.2.0

Description Random sampling from distributions with user-specified population covariance matrix. Marginal information may be fully specified, for which the package implements the VITA (VIne-To-Anything) algorithm Grønneberg and Foldnes (2017) <doi:10.1007/s11336-017-9569-6>. See also Grønneberg, Foldnes and Marcoulides (2022) <doi:10.18637/jss.v102.i03>. Alternatively, marginal skewness and kurtosis may be specified, for which the package implements the IG (independent generator), PLSIM (piecewise linear) and VM (Vale-Maurelli) algorithms, see Foldnes and Olsson (2016) <doi:10.1080/00273171.2015.1133274>, Foldnes and Grønneberg (2021) <doi:10.1080/10705511.2021.1949323> and Vale and Maurelli (1983) <doi:10.1007/BF02293687>, respectively.

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports rvinecopulib (>= 0.5.1.1.0), lavaan (>= 0.6-5), nleqslv, PearsonDS, MASS, stats, parallel, Matrix

Encoding UTF-8

Suggests rmarkdown, knitr, R.rsp, GGally, psych, testthat (>= 3.0.0), magmaan (>= 0.2.0)

Additional_repositories https://jonasmoss.r-universe.dev

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Njaal Foldnes [aut, cre], Steffen Grønneberg [aut]

Maintainer Njaal Foldnes <njal.foldnes@gmail.com>

Repository CRAN

Date/Publication 2026-10-29 11:30:28 UTC

Contents

calibrate_ig 2
calibrate_plsim 3
calibrate_vita 4
calibrate_vm 6
rIG 7
rPLSIM 8
rVM 9
simulate.covsim_calib_ig 10
simulate.covsim_calib_plsim 11
simulate.covsim_calib_vita 11
simulate.covsim_calib_vm 12
simulate_estimate 13
vita 15

Index **18**

`calibrate_ig` *Calibrate the IG (Independent Generator) model*

Description

Performs the calibration step of the IG algorithm: computes the transformation matrix `A` and fits a Pearson distribution to each independent generator such that samples drawn from the calibrated model match the target covariance, skewness, and excess kurtosis. Calibration is decoupled from simulation so the same calibration object can be reused to draw many samples without recomputation.

Usage

```
calibrate_ig(sigma.target, skewness, excesskurtosis, typeA = "triang")
```

Arguments

`sigma.target`	Target population covariance matrix.
`skewness`	Target skewness vector.
`excesskurtosis`	Target excess kurtosis vector.
`typeA`	Either `"triang"` (default, triangular Cholesky factor) or `"symm"` (symmetric square root).

Value

An object of class `c("covsim_calib_ig", "covsim_calib")` that can be passed to `simulate` to draw samples.

Author(s)

Njål Foldnes (<njal.foldnes@gmail.com>)

References

Foldnes, N. and Olson, U. H. (2016). A simple simulation technique for nonnormal data with prespecified skewness, kurtosis, and covariance matrix. Multivariate behavioral research, 51(2-3), 207-219

See Also

rIG, simulate

Examples

```
sigma.target <- diag(3)
cal <- calibrate_ig(sigma.target, skewness = rep(1, 3), excesskurtosis = rep(3, 3))
samples <- stats::simulate(cal, nsim = 5, N = 1000)
```

---

calibrate_plsim *Calibrate the PLSIM (Piecewise Linear) model*

Description

Performs the calibration step of the PLSIM algorithm: fits piecewise linear marginal transforms with the target skewness and kurtosis and searches for intermediate normal correlations that yield the target covariance after transformation. The (expensive) pairwise calibration is decoupled from simulation so the resulting calibration object can be reused to draw many samples without recomputation.

Usage

```
calibrate_plsim(
  sigma.target,
  skewness,
  excesskurtosis,
  numsegments = 4,
  gammalist = NULL,
  monot = FALSE,
  verbose = TRUE
)
```

Arguments

sigma.target	Target population covariance matrix.
skewness	Target skewness vector.
excesskurtosis	Target excess kurtosis vector.
numsegments	The number of line segments in each marginal.
gammalist	A list of breakpoints in each margin.
monot	If TRUE, piecewise linear functions are forced to be monotonous. The implied copula is then normal.
verbose	If TRUE, prints calibration progress.

Value

An object of class `c("covsim_calib_plsim", "covsim_calib")` that can be passed to `simulate` to draw samples.

Author(s)

Njål Foldnes (<njal.foldnes@gmail.com>)

References

Foldnes, N. and Grønneberg S. (2021). Non-normal data simulation using piecewise linear transforms.

See Also

`rPLSIM`, `simulate`

Examples

```
set.seed(1)
sigma.target <- cov(MASS::mvrnorm(5, rep(0, 3), diag(3)))
cal <- calibrate_plsim(sigma.target, skewness = rep(1, 3), excesskurtosis = rep(4, 3))
samples <- stats::simulate(cal, nsim = 2, N = 1000)
```

---

calibrate_vita *Calibrate a regular vine (VITA)*

Description

Wraps `vita` and returns a calibration object that can be passed to `simulate`. The returned object embeds the calibrated `vine_dist` so it can also be used directly with the **rvinecopulib** simulator if desired.

Usage

```
calibrate_vita(
  margins,
  sigma.target,
  vc = NULL,
  family_set = c("clayton", "gauss", "joe", "gumbel", "frank"),
  Nmax = 10^6,
  numrootpoints = 10,
  conflevel = 0.995,
  numpoints = 4,
  verbose = TRUE,
  cores = parallel::detectCores()
)
```

Arguments

margins	A list where each element corresponds to a margin. Each margin element is a list containing the distribution family ("distr") and additional parameters. Must be a distribution available in the stats package.
sigma.target	The target covariance matrix that is to be matched. The diagonal elements must contain the variances of marginal distributions.
vc	A vine dist object as specified by the rvinecopulib package. This object specifies the vine that is to be calibrated. If not provided, a D-vine is assumed.
family_set	A vector of one-parameter pair-copula families that is to be calibrated at each node in the vine. Possible entries are "gauss", "clayton", "joe", "gumbel" and "frank". Calibration of pair-copula families is attempted in the order provided.
Nmax	The sample size used for calibration. Reduce for faster calibration, at the cost of precision.
numrootpoints	The number of estimated roots at the initial calibration stage, which determines a search interval where Nmax samples are drawn
conflevel	Confidence level for determining search interval
numpoints	The number of samples drawn with size Nmax, to determine the root within search interval To increase precision increase this number. To calibrate faster (but less precisely), may be reduced to a number no lower than 2
verbose	If TRUE, outputs details of calibration of each bicopula
cores	Number of cores to use. If larger than 1, computations are done in parallel. May be determined with parallel:detectCores()

Value

If a feasible solution was found, an object of class `c("covsim_calib_vita", "covsim_calib")` that can be passed to `simulate`; otherwise `NULL`.

See Also

`vita`, `simulate`

Examples

```
set.seed(1)
sigma.target <- cov(MASS::mvrnorm(10, mu = rep(0, 3), Sigma = diag(1, 3)))
marginsnorm <- lapply(X = sqrt(diag(sigma.target)),
                      function(X) list(distr = "norm", sd = X))
cal <- calibrate_vita(marginsnorm, sigma.target = sigma.target,
                      Nmax = 10^5, cores = 1)
samples <- stats::simulate(cal, nsim = 1, N = 1000)
```

`calibrate_vm` *Calibrate the VM (Vale-Maurelli) model*

Description

Performs the calibration step of the Vale-Maurelli algorithm: for each variable the Fleishman polynomial $a + bZ + cZ^2 + dZ^3$ (with $a = -c$) matching the target skewness and excess kurtosis is found, and for each pair of variables the intermediate normal correlation that yields the target correlation after transformation is solved from the Vale-Maurelli cubic. Calibration is decoupled from simulation so the same calibration object can be reused to draw many samples without recomputation.

Usage

```
calibrate_vm(sigma.target, skewness, excesskurtosis)
```

Arguments

`sigma.target`	Target population covariance matrix.
`skewness`	Target skewness vector.
`excesskurtosis`	Target excess kurtosis vector.

Details

When the Fleishman equations have several solutions, a monotone increasing polynomial is preferred. For many common skewness/kurtosis combinations (e.g. skewness 1 and excess kurtosis 1.5) only polynomials that turn down in the far tails exist; these are used, and flagged in `monotone`.

Value

An object of class `c("covsim_calib_vm", "covsim_calib")` that can be passed to `simulate` to draw samples. The element `coef` holds the Fleishman coefficients (one row per variable), `monotone` whether each polynomial is monotone increasing, and `z.corr` the intermediate correlation matrix.

References

Fleishman, A. I. (1978). A method for simulating non-normal distributions. Psychometrika, 43(4), 521-532.

Vale, C. D. and Maurelli, V. A. (1983). Simulating multivariate nonnormal distributions. Psychometrika, 48(3), 465-471.

See Also

rVM, simulate

Examples

```
sigma.target <- matrix(c(1, 0.5, 0.5, 1), 2)
cal <- calibrate_vm(sigma.target, skewness = c(1, 2), excesskurtosis = c(2, 7))
samples <- stats::simulate(cal, nsim = 5, N = 1000)
```

---

rIG *Simulation of non-normal data*

Description

Backward-compatible wrapper that combines calibrate_ig and simulate in one call. New code should prefer the calibrate/simulate split so calibration can be reused.

Usage

```
rIG(N, sigma.target, skewness, excesskurtosis, reps = 1, typeA = "triang")
```

Arguments

N	Number of observations to simulate.
sigma.target	Target population covariance matrix.
skewness	Target skewness vector.
excesskurtosis	Target excess kurtosis vector.
reps	Number of simulated samples.
typeA	Either "triang" (default, triangular Cholesky factor) or "symm" (symmetric square root).

Value

A list of simulated samples.

Author(s)

Njål Foldnes (<njal.foldnes@gmail.com>)

References

Foldnes, N. and Olson, U. H. (2016). A simple simulation technique for nonnormal data with prespecified skewness, kurtosis, and covariance matrix. Multivariate behavioral research, 51(2-3), 207-219

Examples

```
set.seed(1234)
model <- '
 # measurement model
   ind60 =~ x1 + x2 + x3
   dem60 =~ y1 + y2 + y3 + y4
   dem65 =~ y5 + y6 + y7 + y8
 # regressions
   dem60 ~ ind60
   dem65 ~ ind60 + dem60
 # residual correlations
   y1 ~~ y5
   y2 ~~ y4 + y6
   y3 ~~ y7
   y4 ~~ y8
   y6 ~~ y8'
fit  <- lavaan::sem(model, data=lavaan::PoliticalDemocracy)
population.sigma <- lavaan::lavInspect(fit, "sigma.hat")
population.skew  <- c(0, 0, 0, 0, 1, 1, 1, 1, 2,2,2 )
population.excesskurt <- c( 1 , 1, 1, 1, 3, 3, 3, 3, 15, 15, 15)
my.samples <- rIG(N=10^3, sigma=population.sigma,
        skewness=population.skew,
        excesskurt=population.excesskurt,
        reps=5)
```

---

rPLSIM *Simulation of non-normal data*

Description

Backward-compatible wrapper that combines `calibrate_plsim` and `simulate` in one call. New code should prefer the calibrate/simulate split so the (expensive) pairwise calibration can be reused.

Usage

```
rPLSIM(
  N,
  sigma.target,
  skewness,
  excesskurtosis,
  reps = 1,
  numsegments = 4,
  gammalist = NULL,
  monot = FALSE,
  verbose = TRUE
)
```

Arguments

N	Number of observations to simulate.
sigma.target	Target population covariance matrix.
skewness	Target skewness vector.
excesskurtosis	Target excess kurtosis vector.
reps	Number of simulated samples.
numsegments	The number of line segments in each marginal.
gammalist	A list of breakpoints in each margin.
monot	If TRUE, piecewise linear functions are forced to be monotonous. The implied copula is then normal.
verbose	If TRUE, prints calibration progress.

Value

A list with two elements. First element: the list of simulated samples. Second element: the fitted piecewise linear functions and the intermediate correlations matrix.

Author(s)

Njål Foldnes (<njal.foldnes@gmail.com>)

References

Foldnes, N. and Grønneberg S. (2021). Non-normal data simulation using piecewise linear transforms.Under review.

Examples

```
set.seed(1)
sigma.target  <- cov(MASS::mvrnorm(5, rep(0,3), diag(3)))
res  <- covsim::rPLSIM(10^5, sigma.target, skewness=rep(1,3), excesskurtosis=rep(4,3))
my.sample  <- res[[1]][[1]]
```

---

rVM *Simulation of non-normal data with the Vale-Maurelli method*

Description

Convenience wrapper that combines calibrate_vm and simulate in one call. Prefer the calibrate/simulate split when drawing many samples from the same target.

Usage

```
rVM(N, sigma.target, skewness, excesskurtosis, reps = 1)
```

Arguments

N	Number of observations to simulate.
sigma.target	Target population covariance matrix.
skewness	Target skewness vector.
excesskurtosis	Target excess kurtosis vector.
reps	Number of simulated samples.

Value

A list of simulated samples.

References

Vale, C. D. and Maurelli, V. A. (1983). Simulating multivariate nonnormal distributions. Psychometrika, 48(3), 465-471.

Examples

```
sigma.target <- matrix(c(1, 0.5, 0.5, 1), 2)
my.samples <- rVM(N = 1000, sigma.target, skewness = c(1, 2),
                  excesskurtosis = c(2, 7), reps = 5)
```

---

simulate.covsim_calib_ig

Simulate from a calibrated IG model

Description

S3 method for simulate that draws samples from a covsim_calib_ig object produced by calibrate_ig.

Usage

```
## S3 method for class 'covsim_calib_ig'
simulate(object, nsim = 1, seed = NULL, N, ...)
```

Arguments

object	A covsim_calib_ig object.
nsim	Number of independent samples (datasets) to return.
seed	Optional seed passed to set.seed.
N	Number of observations per simulated dataset.
...	Unused.

Value

A list of nsim matrices, each with N rows.

simulate.covsim_calib_plsim

Simulate from a calibrated PLSIM model

Description

S3 method for `simulate` that draws samples from a `covsim_calib_plsim` object produced by `calibrate_plsim`.

Usage

```
## S3 method for class 'covsim_calib_plsim'
simulate(object, nsim = 1, seed = NULL, N, ...)
```

Arguments

`object`	A `covsim_calib_plsim` object.
`nsim`	Number of independent samples (datasets) to return.
`seed`	Optional seed passed to `set.seed`.
`N`	Number of observations per simulated dataset.
`...`	Unused.

Value

A list of `nsim` matrices, each with `N` rows and the column names of `sigma.target`.

simulate.covsim_calib_vita

Simulate from a calibrated VITA vine

Description

S3 method for `simulate` that draws samples from a `covsim_calib_vita` object produced by `calibrate_vita`.

Usage

```
## S3 method for class 'covsim_calib_vita'
simulate(object, nsim = 1, seed = NULL, N, cores = 1, ...)
```

Arguments

object	A covsim_calib_vita object.
nsim	Number of independent samples (datasets) to return.
seed	Optional seed passed to set.seed.
N	Number of observations per simulated dataset.
cores	Number of cores passed to rvine.
...	Unused.

Value

A list of nsim data frames (or matrices), each with N rows.

---

simulate.covsim_calib_vm

Simulate from a calibrated VM model

Description

S3 method for simulate that draws samples from a covsim_calib_vm object produced by calibrate_vm.

Usage

```
## S3 method for class 'covsim_calib_vm'
simulate(object, nsim = 1, seed = NULL, N, ...)
```

Arguments

object	A covsim_calib_vm object.
nsim	Number of independent samples (datasets) to return.
seed	Optional seed passed to set.seed.
N	Number of observations per simulated dataset.
...	Unused.

Value

A list of nsim matrices, each with N rows.

`simulate_estimate` *Simulate data and estimate a model repeatedly*

Description

Runs a Monte Carlo study: for every calibrated distribution in `dist` and every sample size in `n`, draws `reps` samples with `simulate`, fits `model` to each sample, and collects what `extractor` returns into one data frame.

Usage

```
simulate_estimate(
  dist,
  model,
  n,
  reps,
  extractor,
  ...,
  fit = "lavaan",
  cores = 1,
  seed = NULL
)
```

Arguments

`dist`	A calibration object (from `calibrate_ig`, `calibrate_plsim`, `calibrate_vm` or `calibrate_vita`), or a named list of such objects to compare several distributions.
`model`	Model syntax (lavaan syntax for `fit = "lavaan"` and `fit = "magmaan"`), or whatever a user-supplied `fit` function expects. Variable names must match the column names of `sigma.target` used in calibration.
`n`	Vector of sample sizes.
`reps`	Number of replications per distribution and sample size.
`extractor`	A function of one fitted object that returns what is to be kept from that replication, as a named vector or list of scalars. Each element becomes a column. The researcher decides what to extract; see the examples for parameter estimates, standard errors, p-values, fit measures and test statistics with lavaan and magmaan. Use `function(fit) list(fit = fit)` to keep the fitted objects themselves (as a list column).
`...`	Further arguments passed to `lavaan::sem()` or `magmaan::magmaan()`, for example `estimator = "MLR"`, `se = "robust.sem"`, `start = "simple"` or `control = list(iter.max = 500)`. Not used with a user-supplied `fit` function, which can set such options itself.
`fit`	`"lavaan"`, `"magmaan"`, or a function `function(data, model)` returning a fitted object.

cores	Number of cores. Values above 1 use mclapply (not available on Windows, where the replications run sequentially).
seed	Optional integer seed. If NULL, a seed is drawn from the current random number generator.

Details

Estimation is done with lavaan (fit = "lavaan", the default), with magmaan (fit = "magmaan", if installed), or with any function function(data, model). With magmaan the model is constructed once per distribution with magmaan::magmaan_model() and reused for every sample, which avoids repeated parsing and setup.

Each replication draws its data from its own L'Ecuyer-CMRG random number stream, determined by seed and the position of the replication in the design. Results are therefore identical for any value of cores. Errors and warnings from fitting or extraction are caught and recorded per replication rather than stopping the run.

Value

A data frame with one row per replication and columns dist, n, rep, converged, error, warning, followed by the extracted values. The seed and package versions are stored in attr(, "info").

Extractors for magmaan

A magmaan fit f gives the parameter table with summary(f)$coefficients (columns lhs, op, rhs, free, est, se, pvalue, ...), its score and likelihood ratio tests with summary(f)$tests, and fit measures with magmaan::fit_measures(f). Note that by default fit_measures() reports robust RMSEA and CFI and a bias-corrected SRMR under the plain names; magmaan::fit_measures(f, lavaan_compat = "MLR") gives lavaan's definitions and names (rmsea, rmsea.robust, cfi.robust, ...). magmaan's default standard errors are robust (sandwich), matching lavaan with estimator = "MLR" rather than plain ML. magmaan fits a mean structure, so its parameter table also contains the intercepts.

See Also

calibrate_ig, calibrate_plsim, calibrate_vm, calibrate_vita

Examples

```
model <- "F =~ x1 + x2 + x3 + x4"
sigma <- matrix(0.5, 4, 4); diag(sigma) <- 1
dimnames(sigma) <- list(paste0("x", 1:4), paste0("x", 1:4))
dists <- list(
  IG = calibrate_ig(sigma, skewness = rep(2, 4), excesskurtosis = rep(7, 4)),
  VM = calibrate_vm(sigma, skewness = rep(2, 4), excesskurtosis = rep(7, 4)))

# rejection rate of the ML chi-square test
chisq <- function(fit) lavaan::fitMeasures(fit, c("chisq", "df", "pvalue"))
res <- simulate_estimate(dists, model, n = 200, reps = 20,
                         extractor = chisq, seed = 1)
aggregate(pvalue ~ dist, data = res, FUN = function(p) mean(p < 0.05))
```

```
# RMSEA and CFI, ordinary and robust (MLR)
fitind <- function(fit)
  lavaan::fitMeasures(fit, c("rmsea", "cfi", "rmsea.robust", "cfi.robust"))
res <- simulate_estimate(dists, model, n = 200, reps = 20, estimator = "MLR",
                         extractor = fitind, seed = 1)
aggregate(cbind(rmsea, rmsea.robust) ~ dist, data = res, FUN = mean)

# estimate, standard error and p-value of one loading
loading <- function(fit) {
  pe <- lavaan::parameterEstimates(fit)
  pe <- pe[pe$lhs == "F" & pe$op == "=~" & pe$rhs == "x2", ]
  c(est = pe$est, se = pe$se, pvalue = pe$pvalue)
}
res <- simulate_estimate(dists$VM, model, n = 200, reps = 20,
                         extractor = loading, seed = 1)
colMeans(res[, c("est", "se")])   # mean estimate vs. mean standard error
sd(res$est)                       # empirical standard error

# the same with magmaan
if (requireNamespace("magmaan", quietly = TRUE) &&
    utils::packageVersion("magmaan") >= "0.2.0") {
  mag_loading <- function(fit) {
    cf <- summary(fit)$coefficients
    cf <- cf[cf$lhs == "F" & cf$op == "=~" & cf$rhs == "x2", ]
    c(est = cf$est, se = cf$se, pvalue = cf$pvalue)
  }
  mag_tests <- function(fit) {
    tests <- summary(fit)$tests
    stats::setNames(tests$pvalue, paste(tests$test, tests$reference, sep = "_"))
  }
  res <- simulate_estimate(dists$VM, model, n = 200, reps = 20, fit = "magmaan",
                           extractor = function(fit) c(mag_loading(fit), mag_tests(fit)),
                           seed = 1)
  colMeans(res[, c("est", "se", "score_peba4")])
}
```

`vita` *Calibrate a regular vine*

Description

`vita` implements the VITA (VIne-To-Anything) algorithm. Covariance matrix and margins are specified, and `vita` calibrates the pair-copulas in each node of the tree to match the target covariance.

Usage

```
vita(
  margins,
```

```
  sigma.target,
  vc = NULL,
  family_set = c("clayton", "gauss", "joe", "gumbel", "frank"),
  Nmax = 10^6,
  numrootpoints = 10,
  conflevel = 0.995,
  numpoints = 4,
  verbose = TRUE,
  cores = parallel::detectCores()
)
```

Arguments

margins	A list where each element corresponds to a margin. Each margin element is a list containing the distribution family ("distr") and additional parameters. Must be a distribution available in the stats package.
sigma.target	The target covariance matrix that is to be matched. The diagonal elements must contain the variances of marginal distributions.
vc	A vine dist object as specified by the rvinecopulib package. This object specifies the vine that is to be calibrated. If not provided, a D-vine is assumed.
family_set	A vector of one-parameter pair-copula families that is to be calibrated at each node in the vine. Possible entries are "gauss", "clayton", "joe", "gumbel" and "frank". Calibration of pair-copula families is attempted in the order provided.
Nmax	The sample size used for calibration. Reduce for faster calibration, at the cost of precision.
numrootpoints	The number of estimated roots at the initial calibration stage, which determines a search interval where Nmax samples are drawn
conflevel	Confidence level for determining search interval
numpoints	The number of samples drawn with size Nmax, to determine the root within search interval To increase precision increase this number. To calibrate faster (but less precisely), may be reduced to a number no lower than 2
verbose	If TRUE, outputs details of calibration of each bicopula
cores	Number of cores to use. If larger than 1, computations are done in parallel. May be determined with parallel:detectCores()

Value

If a feasible solution was found, a vine to be used for simulation

References

Grønneberg, S., Foldnes, N., & Marcoulides, K. M. (2021). covsim: An r package for simulating non-normal data for structural equation models using copulas. Journal of Statistical Software. doi:10.18637/jss.v102.i03

Examples

```
set.seed(1)# define a target covariance. 3 dimensions.
sigma.target <- cov(MASS::mvrnorm(10, mu=rep(0,3), Sigma=diag(1, 3)))

#normal margins that match the covariances:
marginsnorm <- lapply(X=sqrt(diag(sigma.target)),function(X) list(distr="norm", sd=X) )

#calibrate with a default D-vine, with rather low precision (default Nmax is 10^6)
# if cores=1 is removed, all cores are used, with a speed gain
calibrated.vine <- vita(marginsnorm, sigma.target =sigma.target, Nmax=10^5, cores=1)
#check
#round(cov(rvinecopulib::rvine(10^5, calibrated.vine))-sigma.target, 3)

#margins are normal but dependence structure is not
#pairs(rvinecopulib::rvine(500, calibrated.vine))
```

Index

calibrate_ig, 2, *7*, *10*, *13, 14*
calibrate_plsim, 3, *8*, *11*, *13, 14*
calibrate_vita, 4, *11*, *13, 14*
calibrate_vm, 6, *9*, *12–14*

mclapply, *14*

rIG, *3*, 7
rPLSIM, *4*, 8
rvine, *12*
rVM, *7*, 9

set.seed, *10–12*
simulate, *2–13*
simulate.covsim_calib_ig, 10
simulate.covsim_calib_plsim, 11
simulate.covsim_calib_vita, 11
simulate.covsim_calib_vm, 12
simulate_estimate, 13

vita, *4, 5*, 15