Package ‘SimRepeat’

April 14, 2018

Type Package

Title Simulation of Correlated Systems of Equations with Multiple Variable Types

Version 0.1.0

Author Allison Cynthia Fialkowski

Maintainer Allison Cynthia Fialkowski <allijazz@uab.edu>

Description Generate correlated systems of statistical equations which represent repeated measurements or clustered data. These systems contain either: a) continuous normal, non-normal, and mixture variables based on the techniques of Headrick and Beasley (2004) <DOI:10.1081/SAC-120028431> or b) continuous (normal, non-normal and mixture), ordinal, and count (regular or zero-inflated, Poisson and Negative Binomial) variables based on the hierarchical linear models (HLM) approach. Headrick and Beasley's method for continuous variables calculates the beta (slope) coefficients based on the target correlations between independent variables and between outcomes and independent variables. The package provides functions to calculate the expected correlations between outcomes, between outcomes and error terms, and between outcomes and independent variables, extending Headrick and Beasley's equations to include mixture variables. These theoretical values can be compared to the simulated correlations. The HLM approach requires specification of the beta coefficients, but permits group and subject-level independent variables, interactions among independent variables, and fixed and random effects, providing more flexibility in the system of equations. Both methods permit simulation of data sets that mimic real-world clinical or genetic data sets (i.e. plasmodes, as in Vaughan et al., 2009, <10.1016/j.csda.2008.02.032>). The techniques extend those found in the 'SimMultiCorrData' and 'SimCorrMix' packages. Standard normal variables with an imposed intermediate correlation matrix are transformed to generate the desired distributions. Continuous variables are simulated using either Fleishman's third-order (<DOI:10.1007/BF02293811>) or Headrick's fifth-order (<DOI:10.1016/S0167-9473(02)00072-5>) power method transformation (PMT). Simulation occurs at the component-level for continuous mixture distributions. These components are transformed into the desired mixture variables using random multinomial variables based on the mixing probabilities. The target correlation matrices are specified in terms of correlations with components of continuous mixture variables. Binary and ordinal variables are simulated by discretizing the normal variables at quantiles defined by the marginal distributions. Count variables are simulated using the inverse CDF method. There are two simulation pathways for the multi-variable type systems which differ by intermediate correlations involving count variables. Correlation Method 1 adapts Yahav and Shmueli's 2012 method <DOI:10.1002/asmb.901> and performs best with large count variable means and positive correlations or small means and negative correlations. Correlation Method 2 adapts Barbiero and Ferrari's 2015 modification of the 'GenOrd' package <DOI:10.1002/asmb.2072> and performs best under the opposite scenarios. There are three methods available for correcting non-positive definite correlation matrices. The optional error loop may be used to improve the accuracy of the final correlation matrices. The package also provides function to check parameter inputs and summarize the simulated systems of equations.

Depends R (>= 3.4.0),
SimMultiCorrData (>= 0.2.1), SimCorrMix (>= 0.1.0)

License GPL-2

Imports BB, nleqslv, MASS, Matrix, VGAM, triangle, ggplot2, grid, stats, utils

Encoding UTF-8

LazyData true

Roxygen list(wrap = FALSE)

RoxygenNote 6.0.1

Suggests knitr,
rmarkdown, printr, bookdown, nlme, reshape2,
testthat

VignetteBuilder knitr

URL `https://github.com/AFialkowski/SimRepeat`

R topics documented:

`adj_grad` *Convert Non-Positive-Definite Correlation Matrix to Positive-Definite Matrix Using the Adjusted Gradient Updating Method*

Description

This function converts a non-positive-definite correlation matrix to a positive-definite matrix using the adjusted gradient updating method with initial matrix `B1`.

Usage

```
adj_grad(Sigma = NULL, B1 = NULL, tau = 0.5, tol = 0.1, steps = 100,
  msteps = 10)
```

Arguments

`Sigma`	the non-PD correlation matrix
`B1`	the initial matrix for algorithm; if NULL, uses a scaled initial matrix with diagonal elements `sqrt(nrow(Sigma))/2`
`tau`	parameter used to calculate theta
`tol`	maximum error for Frobenius norm distance between new matrix and original matrix
`steps`	maximum number of steps for k (default = 100)
`msteps`	maximum number of steps for m (default = 10)

Value

list with `Sigma2` the new correlation matrix, `dist` the Frobenius norm distance between `Sigma2` and `Sigma`, `eig0` original eigenvalues of `Sigma`, `eig2` eigenvalues of `Sigma2`

References

S Maree (2012). Correcting Non Positive Definite Correlation Matrices. BSc Thesis Applied Mathematics, TU Delft. `http://resolver.tudelft.nl/uuid:2175c274-ab03-4fd5-85a9-228fe421cdbf`.

JF Yin and Y Zhang (2013). Alternative gradient algorithms for computing the nearest correlation matrix. Applied Mathematics and Computation, 219(14): 7591-7599. `https://doi.org/10.1016/j.amc.2013.01.045`.

Y Zhang and JF Yin. Modified alternative gradients algorithm for computing the nearest correlation matrix. Internal paper of the Tongji University, Shanghai.

Examples

```
Sigma <- matrix(c(1, 0, 0.8, 0, 1, 0.8, 0.8, 0.8, 1), 3, 3, byrow = TRUE)
adj_grad(Sigma)
```

`calc_betas`	*Calculate Beta Coefficients for Correlated Systems of Continuous Variables*

Description

This function calculates the beta (slope) coefficients used in `nonnormsys` by the techniques of Headrick and Beasley (doi: 10.1081/SAC120028431). These coefficients are determined based on the correlations between independent variables $X_{(pj)}$ for a given outcome Y_p, for `p = 1, ..., M`, the correlations between that outcome Y_p and the $X_{(pj)}$ terms, and the variances. If there are continuous mixture variables and the matrices in `corr.yx` are specified in terms of correlations between outcomes and non-mixture and mixture variables, then the solutions are the slope coefficients for the non-mixture and mixture variables. In this case, the number of columns of the matrices of `corr.yx` should not match the dimensions of the matrices in `corr.x`. The correlations in `corr.x` will be calculated in terms of non-mixture and mixture variables using `rho_M1M2` and `rho_M1Y`. If there are continuous mixture variables and the matrices in `corr.yx` are specified in terms of correlations between outcomes and non-mixture and components of mixture variables, then the solutions are the slope coefficients for the non-mixture and components of mixture variables. In this case, the number of columns of the matrices of `corr.yx` should match the dimensions of the matrices in `corr.x`. The vignette **Theory and Equations for Correlated Systems of Continuous Variables** gives the equations, and the vignette **Correlated Systems of Statistical Equations with Non-Mixture and Mixture Continuous Variables** gives examples. There are also vignettes in `SimCorrMix` which provide more details on continuous non-mixture and mixture variables.

Usage

```
calc_betas(corr.yx = list(), corr.x = list(), vars = list(),
  mix_pis = list(), mix_mus = list(), mix_sigmas = list(),
  error_type = c("non_mix", "mix"), n = 25, seed = 1234)
```

Arguments

`corr.yx` a list of length `M` = # of equations, where the p-th component is a 1 row matrix of correlations between Y_p and $X_{(pj)}$; if there are mixture variables and the `betas` are desired in terms of these (and not the components), then `corr.yx` should be specified in terms of correlations between outcomes and non-mixture or mixture variables, and the number of columns of the matrices of `corr.yx` should not match the dimensions of the matrices in `corr.x`; if the `betas` are desired in terms of the components, then `corr.yx` should be specified in terms of correlations between outcomes and non-mixture or components of mixture variables, and the number of columns of the matrices of `corr.yx` should match the dimensions of the matrices in `corr.x`

`corr.x` list of length `M`, each component a list of length `M`; `corr.x[[p]][[q]]` is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); if p = q, `corr.x[[p]][[q]]` is a correlation matrix with `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p; if p != q, `corr.x[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that `ncol(corr.x[[p]][[q]])` = # $X_{(qj)}$ for outcome Y_q; order is 1st continuous non-mixture and 2nd components of continuous mixture variables

`vars` a list of same length as `corr.x` of vectors of variances for $X_{(pj)}, E$; E term should be last; order should be the same as in `corr.x`

`mix_pis` a list of same length as `corr.x`, where `mix_pis[[p]][[j]]` is a vector of mixing probabilities for $X_{mix(pj)}$ that sum to 1, the j-th mixture covariate for outcome Y_p; the last element of `mix_pis[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_pis[[p]] = NULL`

`mix_mus` a list of same length as `corr.x`, where `mix_mus[[p]][[j]]` is a vector of means for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of `mix_mus[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_mus[[p]] = NULL`

`mix_sigmas` a list of same length as `corr.x`, where `mix_sigmas[[p]][[j]]` is a vector of standard deviations for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of `mix_sigmas[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_sigmas[[p]] = NULL`

`error_type` "non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions, defaults to "non_mix"

n	the number of sets of random uniform(0, 1) numbers used as starting values in nleqslv to find the betas
seed	the seed for random number generation

Value

betas a matrix of slope coefficients where rows represent the outcomes; extra zeros are appended at the end of a row if that outcome has fewer $X_{(pj)}$ terms

References

Headrick TC, Beasley TM (2004). A Method for Simulating Correlated Non-Normal Systems of Linear Statistical Equations. Communications in Statistics - Simulation and Computation, 33(1). doi: 10.1081/SAC120028431

See Also

nonnormsys, rho_M1M2, rho_M1Y

Examples

```
# Example: system of three equations for 2 independent variables, where each
# error term has unit variance, from Headrick & Beasley (2002)
corr.yx <- list(matrix(c(0.4, 0.4), 1), matrix(c(0.5, 0.5), 1),
  matrix(c(0.6, 0.6), 1))
corr.x <- list()
corr.x[[1]] <- corr.x[[2]] <- corr.x[[3]] <- list()
corr.x[[1]][[1]] <- matrix(c(1, 0.1, 0.1, 1), 2, 2)
corr.x[[1]][[2]] <- matrix(c(0.1974318, 0.1859656, 0.1879483, 0.1858601),
  2, 2, byrow = TRUE)
corr.x[[1]][[3]] <- matrix(c(0.2873190, 0.2589830, 0.2682057, 0.2589542),
  2, 2, byrow = TRUE)
corr.x[[2]][[1]] <- t(corr.x[[1]][[2]])
corr.x[[2]][[2]] <- matrix(c(1, 0.35, 0.35, 1), 2, 2)
corr.x[[2]][[3]] <- matrix(c(0.5723303, 0.4883054, 0.5004441, 0.4841808),
  2, 2, byrow = TRUE)
corr.x[[3]][[1]] <- t(corr.x[[1]][[3]])
corr.x[[3]][[2]] <- t(corr.x[[2]][[3]])
corr.x[[3]][[3]] <- matrix(c(1, 0.7, 0.7, 1), 2, 2)
vars <- list(rep(1, 3), rep(1, 3), rep(1, 3))
calc_betas(corr.yx, corr.x, vars)
```

---

calc_corr_y *Calculate Expected Correlation Matrix of Outcomes (Y) for Correlated Systems of Continuous Variables*

Description

This function calculates the expected correlation matrix for outcomes (Y) in a correlated system of continuous variables. This system is generated with nonnormsys using the techniques of Headrick and Beasley (doi: 10.1081/SAC120028431). These correlations are determined based on the beta (slope) coefficients calculated with calc_betas, the correlations between independent variables

$X_{(pj)}$ for a given outcome Y_p, for p = 1, ..., M, the correlations between error terms, and the variances. The result can be used to compare the simulated correlation matrix to the theoretical correlation matrix. If there are continuous mixture variables and the betas are specified in terms of non-mixture and mixture variables and/or error_type = "mix", then the correlations in corr.x and/or corr.e will be calculated in terms of non-mixture and mixture variables using rho_M1M2 and rho_M1Y. In this case, the dimensions of the matrices in corr.x should not match the number of columns of betas. The vignette **Theory and Equations for Correlated Systems of Continuous Variables** gives the equations, and the vignette **Correlated Systems of Statistical Equations with Non-Mixture and Mixture Continuous Variables** gives examples. There are also vignettes in SimCorrMix which provide more details on continuous non-mixture and mixture variables.

Usage

```
calc_corr_y(betas = NULL, corr.x = list(), corr.e = NULL, vars = list(),
  mix_pis = list(), mix_mus = list(), mix_sigmas = list(),
  error_type = c("non_mix", "mix"))
```

Arguments

betas	a matrix of the slope coefficients calculated with calc_betas, rows represent the outcomes
corr.x	list of length M, each component a list of length M; corr.x[[p]][[q]] is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); if p = q, corr.x[[p]][[q]] is a correlation matrix with nrow(corr.x[[p]][[q]]) = # $X_{(pj)}$ for outcome Y_p; if p != q, corr.x[[p]][[q]] is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that nrow(corr.x[[p]][[q]]) = # $X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that ncol(corr.x[[p]][[q]]) = # $X_{(qj)}$ for outcome Y_q; order is 1st continuous non-mixture and 2nd components of continuous mixture variables
corr.e	correlation matrix for continuous non-mixture or components of mixture error terms
vars	a list of same length as corr.x of vectors of variances for $X_{(pj)}, E$; E term should be last; order should be the same as in corr.x
mix_pis	a list of same length as corr.x, where mix_pis[[p]][[j]] is a vector of mixing probabilities for $X_{mix(pj)}$ that sum to 1, the j-th mixture covariate for outcome Y_p; the last element of mix_pis[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_pis[[p]] = NULL
mix_mus	a list of same length as corr.x, where mix_mus[[p]][[j]] is a vector of means for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of mix_mus[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_mus[[p]] = NULL
mix_sigmas	a list of same length as corr.x, where mix_sigmas[[p]][[j]] is a vector of standard deviations for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of mix_sigmas[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_sigmas[[p]] = NULL
error_type	"non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions, defaults to "non_mix"

Value

corr.y the correlation matrix for the outcomes Y

References

Headrick TC, Beasley TM (2004). A Method for Simulating Correlated Non-Normal Systems of Linear Statistical Equations. Communications in Statistics - Simulation and Computation, 33(1). doi: 10.1081/SAC120028431

See Also

nonnormsys, calc_betas, rho_M1M2, rho_M1Y

Examples

```
# Example: system of three equations for 2 independent variables, where each
# error term has unit variance, from Headrick & Beasley (2002)
corr.yx <- list(matrix(c(0.4, 0.4), 1), matrix(c(0.5, 0.5), 1),
  matrix(c(0.6, 0.6), 1))
corr.x <- list()
corr.x[[1]] <- corr.x[[2]] <- corr.x[[3]] <- list()
corr.x[[1]][[1]] <- matrix(c(1, 0.1, 0.1, 1), 2, 2)
corr.x[[1]][[2]] <- matrix(c(0.1974318, 0.1859656, 0.1879483, 0.1858601),
  2, 2, byrow = TRUE)
corr.x[[1]][[3]] <- matrix(c(0.2873190, 0.2589830, 0.2682057, 0.2589542),
  2, 2, byrow = TRUE)
corr.x[[2]][[1]] <- t(corr.x[[1]][[2]])
corr.x[[2]][[2]] <- matrix(c(1, 0.35, 0.35, 1), 2, 2)
corr.x[[2]][[3]] <- matrix(c(0.5723303, 0.4883054, 0.5004441, 0.4841808),
  2, 2, byrow = TRUE)
corr.x[[3]][[1]] <- t(corr.x[[1]][[3]])
corr.x[[3]][[2]] <- t(corr.x[[2]][[3]])
corr.x[[3]][[3]] <- matrix(c(1, 0.7, 0.7, 1), 2, 2)
corr.e <- matrix(0.4, nrow = 3, ncol = 3)
diag(corr.e) <- 1
vars <- list(rep(1, 3), rep(1, 3), rep(1, 3))
betas <- calc_betas(corr.yx, corr.x, vars)
calc_corr_y(betas, corr.x, corr.e, vars)
```

---

calc_corr_ye — *Calculate Expected Matrix of Correlations between Outcomes (Y) and Error Terms (E) for Correlated Systems of Continuous Variables*

Description

This function calculates the expected correlation matrix between Outcomes (Y) and Error Terms (E) in a correlated system of continuous variables. This system is generated with `nonnormsys` using the techniques of Headrick and Beasley (doi: 10.1081/SAC120028431). These correlations are determined based on the beta (slope) coefficients calculated with `calc_betas`, the correlations between independent variables $X_{(pj)}$ for a given outcome Y_p, for `p = 1, ..., M`, the correlations between error terms, and the variances. The result can be used to compare the simulated correlation matrix to the theoretical correlation matrix. If there are continuous mixture variables and the betas are specified in terms of non-mixture and mixture variables, then correlations in `corr.x` will be recalculated in terms of non-mixture or mixture variables using `rho_M1M2` and `rho_M1Y`. In this case, the dimensions of the matrices in `corr.x` should not match the number of columns of `betas`. If `error_type = "mix"`, the correlations in `corr.e` will also be recalculated and the function result will be in terms of mixture error terms. If `error_type = "non_mix"`, the function result will be in terms of non-mixture error terms. The vignette **Theory and Equations for Correlated Systems of Continuous Variables** gives the equations, and the vignette **Correlated Systems of Statistical Equations with Non-Mixture and Mixture Continuous Variables** gives examples. There are also vignettes in `SimCorrMix` which provide more details on continuous non-mixture and mixture variables.

Usage

```
calc_corr_ye(betas = NULL, corr.x = list(), corr.e = NULL,
  vars = list(), mix_pis = list(), mix_mus = list(),
  mix_sigmas = list(), error_type = c("non_mix", "mix"))
```

Arguments

`betas`	a matrix of the slope coefficients calculated with `calc_betas`, rows represent the outcomes
`corr.x`	list of length `M`, each component a list of length `M`; `corr.x[[p]][[q]]` is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); if p = q, `corr.x[[p]][[q]]` is a correlation matrix with `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p; if p != q, `corr.x[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that `ncol(corr.x[[p]][[q]])` = # $X_{(qj)}$ for outcome Y_q; order is 1st continuous non-mixture and 2nd components of continuous mixture variables
`corr.e`	correlation matrix for continuous non-mixture or components of mixture error terms
`vars`	a list of same length as `corr.x` of vectors of variances for $X_{(pj)}, E$; E term should be last; order should be the same as in `corr.x`
`mix_pis`	a list of same length as `corr.x`, where `mix_pis[[p]][[j]]` is a vector of mixing probabilities for $X_{mix(pj)}$ that sum to 1, the j-th mixture covariate for outcome Y_p; the last element of `mix_pis[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_pis[[p]] = NULL`
`mix_mus`	a list of same length as `corr.x`, where `mix_mus[[p]][[j]]` is a vector of means for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of `mix_mus[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_mus[[p]] = NULL`
`mix_sigmas`	a list of same length as `corr.x`, where `mix_sigmas[[p]][[j]]` is a vector of standard deviations for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of `mix_sigmas[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_sigmas[[p]] = NULL`
`error_type`	"non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions, defaults to "non_mix"

Value

`corr.ye` the matrix of correlations between Y and E

References

Headrick TC, Beasley TM (2004). A Method for Simulating Correlated Non-Normal Systems of Linear Statistical Equations. Communications in Statistics - Simulation and Computation, 33(1). doi: 10.1081/SAC120028431

See Also

nonnormsys, calc_betas, rho_M1M2, rho_M1Y

Examples

```
# Example: system of three equations for 2 independent variables, where each
# error term has unit variance, from Headrick & Beasley (2002)
corr.yx <- list(matrix(c(0.4, 0.4), 1), matrix(c(0.5, 0.5), 1),
  matrix(c(0.6, 0.6), 1))
corr.x <- list()
corr.x[[1]] <- corr.x[[2]] <- corr.x[[3]] <- list()
corr.x[[1]][[1]] <- matrix(c(1, 0.1, 0.1, 1), 2, 2)
corr.x[[1]][[2]] <- matrix(c(0.1974318, 0.1859656, 0.1879483, 0.1858601),
  2, 2, byrow = TRUE)
corr.x[[1]][[3]] <- matrix(c(0.2873190, 0.2589830, 0.2682057, 0.2589542),
  2, 2, byrow = TRUE)
corr.x[[2]][[1]] <- t(corr.x[[1]][[2]])
corr.x[[2]][[2]] <- matrix(c(1, 0.35, 0.35, 1), 2, 2)
corr.x[[2]][[3]] <- matrix(c(0.5723303, 0.4883054, 0.5004441, 0.4841808),
  2, 2, byrow = TRUE)
corr.x[[3]][[1]] <- t(corr.x[[1]][[3]])
corr.x[[3]][[2]] <- t(corr.x[[2]][[3]])
corr.x[[3]][[3]] <- matrix(c(1, 0.7, 0.7, 1), 2, 2)
corr.e <- matrix(0.4, nrow = 3, ncol = 3)
diag(corr.e) <- 1
vars <- list(rep(1, 3), rep(1, 3), rep(1, 3))
betas <- calc_betas(corr.yx, corr.x, vars)
calc_corr_ye(betas, corr.x, corr.e, vars)
```

---

calc_corr_yx — *Calculate Expected Matrix of Correlations between Outcomes (Y) and Covariates (X) for Correlated Systems of Continuous Variables*

Description

This function calculates the expected correlation matrix between Outcomes (Y) and Covariates (X) in a correlated system of continuous variables. This system is generated with nonnormsys using the techniques of Headrick and Beasley (doi: 10.1081/SAC120028431). These correlations are determined based on the beta (slope) coefficients calculated with calc_betas, the correlations between independent variables $X_{(pj)}$ for a given outcome Y_p, for p = 1, ..., M, and the variances. The result can be used to compare the simulated correlation matrix to the theoretical correlation matrix. If there are continuous mixture variables and the betas are specified in terms of non-mixture and mixture variables, then the correlations in corr.x will be calculated in terms of non-mixture and mixture variables using rho_M1M2 and rho_M1Y. In this case, the dimensions of the matrices in corr.x should not match the number of columns of betas. The function result will be in terms of non-mixture and mixture variables. Otherwise, the result will be in terms of non-mixture and components of mixture variables. The vignette **Theory and Equations for Correlated Systems of Continuous Variables** gives the equations, and the vignette **Correlated Systems of Statistical Equations with Non-Mixture and Mixture Continuous Variables** gives examples. There are also vignettes in `SimCorrMix` which provide more details on continuous non-mixture and mixture variables.

Usage

```
calc_corr_yx(betas = NULL, corr.x = list(), vars = list(),
  mix_pis = list(), mix_mus = list(), mix_sigmas = list(),
  error_type = c("non_mix", "mix"))
```

Arguments

`betas`	a matrix of the slope coefficients calculated with `calc_betas`, rows represent the outcomes
`corr.x`	list of length `M`, each component a list of length `M`; `corr.x[[p]][[q]]` is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); if p = q, `corr.x[[p]][[q]]` is a correlation matrix with `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p; if p != q, `corr.x[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that `ncol(corr.x[[p]][[q]])` = # $X_{(qj)}$ for outcome Y_q; order is 1st continuous non-mixture and 2nd components of continuous mixture variables
`vars`	a list of same length as `corr.x` of vectors of variances for $X_{(pj)}, E$; E term should be last; order should be the same as in `corr.x`
`mix_pis`	a list of same length as `corr.x`, where `mix_pis[[p]][[j]]` is a vector of mixing probabilities for $X_{mix(pj)}$ that sum to 1, the j-th mixture covariate for outcome Y_p; the last element of `mix_pis[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_pis[[p]] = NULL`
`mix_mus`	a list of same length as `corr.x`, where `mix_mus[[p]][[j]]` is a vector of means for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of `mix_mus[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_mus[[p]] = NULL`
`mix_sigmas`	a list of same length as `corr.x`, where `mix_sigmas[[p]][[j]]` is a vector of standard deviations for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last element of `mix_sigmas[[p]]` is for E_p (if `error_type = "mix"`); if Y_p has no mixture variables, use `mix_sigmas[[p]] = NULL`
`error_type`	"non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions, defaults to "non_mix"

Value

`corr.yx` a list of length `M`, where `corr.yx[[p]]` is matrix of correlations between Y (rows) and X_p (columns); if the dimensions of `betas` match the dimensions of the matrices in `corr.x`, then the correlations will be in terms of non-mixture and components of mixture variables; otherwise, `mix_pis`, `mix_mus`, and `mix_sigmas` must be provided and the correlations will be in terms of non-mixture and mixture variables

References

Headrick TC, Beasley TM (2004). A Method for Simulating Correlated Non-Normal Systems of Linear Statistical Equations. Communications in Statistics - Simulation and Computation, 33(1). doi: 10.1081/SAC120028431

See Also

nonnormsys, calc_betas, rho_M1M2, rho_M1Y

Examples

```
# Example: system of three equations for 2 independent variables, where each
# error term has unit variance, from Headrick & Beasley (2002)
corr.yx <- list(matrix(c(0.4, 0.4), 1), matrix(c(0.5, 0.5), 1),
  matrix(c(0.6, 0.6), 1))
corr.x <- list()
corr.x[[1]] <- corr.x[[2]] <- corr.x[[3]] <- list()
corr.x[[1]][[1]] <- matrix(c(1, 0.1, 0.1, 1), 2, 2)
corr.x[[1]][[2]] <- matrix(c(0.1974318, 0.1859656, 0.1879483, 0.1858601),
  2, 2, byrow = TRUE)
corr.x[[1]][[3]] <- matrix(c(0.2873190, 0.2589830, 0.2682057, 0.2589542),
  2, 2, byrow = TRUE)
corr.x[[2]][[1]] <- t(corr.x[[1]][[2]])
corr.x[[2]][[2]] <- matrix(c(1, 0.35, 0.35, 1), 2, 2)
corr.x[[2]][[3]] <- matrix(c(0.5723303, 0.4883054, 0.5004441, 0.4841808),
  2, 2, byrow = TRUE)
corr.x[[3]][[1]] <- t(corr.x[[1]][[3]])
corr.x[[3]][[2]] <- t(corr.x[[2]][[3]])
corr.x[[3]][[3]] <- matrix(c(1, 0.7, 0.7, 1), 2, 2)
corr.e <- matrix(0.4, nrow = 3, ncol = 3)
diag(corr.e) <- 1
vars <- list(rep(1, 3), rep(1, 3), rep(1, 3))
betas <- calc_betas(corr.yx, corr.x, vars)
calc_corr_yx(betas, corr.x, vars)
```

---

checkpar *Parameter Check for Simulation Functions*

Description

This function checks the parameter inputs to the simulation functions nonnormsys, corrsys, and corrsys2. It should be used prior to execution of these functions to ensure all inputs are of the correct format. Those functions do not contain parameter checks in order to decrease simulation time. This would be important if the user is running several simulation repetitions so that the inputs only have to be checked once. Note that the inputs do not include all of the inputs to the simulation functions. See the appropriate function documentation for more details about parameter inputs. Since the parameter input list is extensive and this function does not check for all possible errors, if simulation gives an error, the user should still check the parameter inputs.

Usage

```
checkpar(M = NULL, method = c("Fleishman", "Polynomial"),
  error_type = c("non_mix", "mix"), means = list(), vars = list(),
  skews = list(), skurts = list(), fifths = list(), sixths = list(),
  Six = list(), mix_pis = list(), mix_mus = list(), mix_sigmas = list(),
  mix_skews = list(), mix_skurts = list(), mix_fifths = list(),
  mix_sixths = list(), mix_Six = list(), marginal = list(),
  support = list(), lam = list(), p_zip = list(), pois_eps = list(),
  size = list(), prob = list(), mu = list(), p_zinb = list(),
  nb_eps = list(), corr.x = list(), corr.yx = list(), corr.e = NULL,
  same.var = NULL, subj.var = NULL, int.var = NULL, tint.var = NULL,
  betas.0 = NULL, betas = list(), betas.subj = list(),
  betas.int = list(), betas.t = NULL, betas.tint = list(),
  rand.int = c("none", "non_mix", "mix"), rand.tsl = c("none", "non_mix",
  "mix"), rand.var = NULL, corr.u = list(), quiet = FALSE)
```

Arguments

`M` the number of dependent variables Y (outcomes); equivalently, the number of equations in the system

`method` the PMT method used to generate all continuous variables, including independent variables (covariates), error terms, and random effects; "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation

`error_type` "non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions

`means` if no random effects, a list of length `M` where `means[[p]]` contains a vector of means for the continuous independent variables in equation p with non-mixture (X_{cont}) or mixture (X_{mix}) distributions and for the error terms (E); order in vector is X_{cont}, X_{mix}, E

if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `means[M + 1]` or `means[(M + 1):(2 * M)]` are vectors of means for all random effects with continuous non-mixture or mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int != "none"`), 2nd random time slope U_1 (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions U_{cont}, 4th other random slopes with mixture distributions U_{mix}

`vars` a list of same length and order as `means` containing vectors of variances for the continuous variables, error terms, and any random effects

`skews` if no random effects, a list of length `M` where `skews[[p]]` contains a vector of skew values for the continuous independent variables in equation p with non-mixture (X_{cont}) distributions and for E if `error_type = "non_mix"`; order in vector is X_{cont}, E

if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `skews[M + 1]` or `skews[(M + 1):(2 * M)]` are vectors of skew values for all random effects with continuous non-mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int = "non_mix"`), 2nd random time slope U_1 (if `rand.tsl = "non_mix"`), 3rd other random slopes with non-mixture distributions U_{cont}

skurts
: a list of same length and order as skews containing vectors of standardized kurtoses (kurtosis - 3) for the continuous variables, error terms, and any random effects with non-mixture distributions

fifths
: a list of same length and order as skews containing vectors of standardized fifth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for method = "Fleishman"

sixths
: a list of same length and order as skews containing vectors of standardized sixth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for method = "Fleishman"

Six
: a list of length M, M + 1, or 2 * M, where Six[1:M] are for X_{cont}, E (if error_type = "non_mix") and Six[M + 1] or Six[(M + 1):(2 * M)] are for non-mixture U; if error_type = "mix" and there are only random effects (i.e., length(corr.x) = 0), use Six[1:M] = rep(list(NULL), M) so that Six[M + 1] or Six[(M + 1):(2 * M)] describes the non-mixture U;

Six[[p]][[j]] is a vector of sixth cumulant correction values to aid in finding a valid PDF for $X_{cont(pj)}$, the j-th continuous non-mixture covariate for outcome Y_p; the last vector in Six[[p]] is for E_p (if error_type = "non_mix"); use Six[[p]][[j]] = NULL if no correction desired for $X_{cont(pj)}$; use Six[[p]] = NULL if no correction desired for any continuous non-mixture covariate or error term in equation p

Six[[M + p]][[j]] is a vector of sixth cumulant correction values to aid in finding a valid PDF for $U_{(pj)}$, the j-th non-mixture random effect for outcome Y_p; use Six[[M + p]][[j]] = NULL if no correction desired for $U_{(pj)}$; use Six[[M + p]] = NULL if no correction desired for any continuous non-mixture random effect in equation p

keep Six = list() if no corrections desired for all equations or if method = "Fleishman"

mix_pis
: list of length M, M + 1 or 2 * M, where mix_pis[1:M] are for X_{cont}, E (if error_type = "mix") and mix_pis[M + 1] or mix_pis[(M + 1):(2 * M)] are for mixture U; use mix_pis[[p]] = NULL if equation p has no continuous mixture terms if error_type = "non_mix" and there are only random effects (i.e., length(corr.x) = 0), use mix_pis[1:M] = NULL so that mix_pis[M + 1] or mix_pis[(M + 1):(2 * M)] describes the mixture U;

mix_pis[[p]][[j]] is a vector of mixing probabilities of the component distributions for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last vector in mix_pis[[p]] is for E_p (if error_type = "mix"); components should be ordered as in corr.x

mix_pis[[M + p]][[j]] is a vector of mixing probabilities of the component distributions for $U_{(pj)}$, the j-th random effect with a mixture distribution for outcome Y_p; order is 1st random intercept (if rand.int = "mix"), 2nd random time slope (if rand.tsl = "mix"), 3rd other random slopes with mixture distributions; components should be ordered as in corr.u

mix_mus
: list of same length and order as mix_pis;

mix_mus[[p]][[j]] is a vector of means of the component distributions for $X_{mix(pj)}$, the last vector in mix_mus[[p]] is for E_p (if error_type = "mix")

mix_mus[[p]][[j]] is a vector of means of the component distributions for $U_{mix(pj)}$

mix_sigmas
: list of same length and order as mix_pis;

mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $X_{mix(pj)}$, the last vector in mix_sigmas[[p]] is for E_p (if error_type = "mix")

mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $U_{mix(pj)}$

mix_skews list of same length and order as mix_pis;

mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $X_{mix(pj)}$, the last vector in mix_skews[[p]] is for E_p (if error_type = "mix")

mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $U_{mix(pj)}$

mix_skurts list of same length and order as mix_pis;

mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $X_{mix(pj)}$, the last vector in mix_skurts[[p]] is for E_p (if error_type = "mix")

mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $U_{mix(pj)}$

mix_fifths list of same length and order as mix_pis; not necessary for method = "Fleishman";

mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_fifths[[p]] is for E_p (if error_type = "mix")

mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $U_{mix(pj)}$

mix_sixths list of same length and order as mix_pis; not necessary for method = "Fleishman";

mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_sixths[[p]] is for E_p (if error_type = "mix")

mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $U_{mix(pj)}$

mix_Six a list of same length and order as mix_pis; keep mix_Six = list() if no corrections desired for all equations or if method = "Fleishman"

p-th component of mix_Six[1:M] is a list of length equal to the total number of component distributions for the $X_{mix(p)}$ and E_p (if error_type = "mix"); mix_Six[[p]][[j]] is a vector of sixth cumulant corrections for the j-th component distribution (i.e., if there are 2 continuous mixture independent variables for Y_p, where $X_{mix(p1)}$ has 2 components and $X_{mix(p2)}$ has 3 components, then length(mix_Six[[p]]) = 5 and mix_Six[[p]][[3]] would correspond to the 1st component of $X_{mix(p2)}$); use mix_Six[[p]][[j]] = NULL if no correction desired for that component; use mix_Six[[p]] = NULL if no correction desired for any component of $X_{mix(p)}$ and E_p

q-th component of mix_Six[M + 1] or mix_Six[(M + 1):(2 * M)] is a list of length equal to the total number of component distributions for the $U_{mix(q)}$; mix_Six[[q]][[j]] is a vector of sixth cumulant corrections for the j-th component distribution; use mix_Six[[q]][[j]] = NULL if no correction desired for that component; use mix_Six[[q]] = NULL if no correction desired for any component of $U_{mix(q)}$

marginal a list of length M, with the p-th component a list of cumulative probabilities for the ordinal variables associated with outcome Y_p (use marginal[[p]] = NULL if outcome Y_p has no ordinal variables); marginal[[p]][[j]] is a vector of the cumulative probabilities defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1); for binary variables, the probability is the probability of the 1st category, which has the smaller support value; `length(marginal[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`support` a list of length `M`, with the p-th component a list of support values for the ordinal variables associated with outcome Y_p; use `support[[p]] = NULL` if outcome Y_p has no ordinal variables; `support[[p]][[j]]` is a vector of the support values defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if not provided, the default for r categories is 1, ..., r

`lam` list of length `M`, p-th component a vector of lambda (means > 0) values for Poisson variables for outcome Y_p (see `stats::dpois`); order is 1st regular Poisson and 2nd zero-inflated Poisson; use `lam[[p]] = NULL` if outcome Y_p has no Poisson variables; `length(lam[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`p_zip` a list of vectors of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `VGAM::dzipois`); if `p_zip` = 0, Y_{pois} has a regular Poisson distribution; if `p_zip` is in (0, 1), Y_{pois} has a zero-inflated Poisson distribution; if `p_zip` is in `(-(exp(lam) - 1)^(-1), 0)`, Y_{pois} has a zero-deflated Poisson distribution and `p_zip` is not a probability; if `p_zip = -(exp(lam) - 1)^(-1)`, Y_{pois} has a positive-Poisson distribution (see `VGAM::dpospois`); order is 1st regular Poisson and 2nd zero-inflated Poisson; if a single number, all Poisson variables given this value; if a vector of length `M`, all Poisson variables in equation p given `p_zip[p]`; otherwise, missing values are set to 0 and ordered 1st

`pois_eps` list of length `M`, p-th component a vector of length `lam[[p]]` containing cumulative probability truncation values used to calculate intermediate correlations involving Poisson variables; order is 1st regular Poisson and 2nd zero-inflated Poisson; if a single number, all Poisson variables given this value; if a vector of length `M`, all Poisson variables in equation p given `pois_eps[p]`; otherwise, missing values are set to 0.0001 and ordered 1st

`size` list of length `M`, p-th component a vector of size parameters for the Negative Binomial variables for outcome Y_p (see `stats::dnbinom`); order is 1st regular NB and 2nd zero-inflated NB; use `size[[p]] = NULL` if outcome Y_p has no Negative Binomial variables; `length(size[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`prob` list of length `M`, p-th component a vector of success probabilities for the Negative Binomial variables for outcome Y_p (see `stats::dnbinom`); order is 1st regular NB and 2nd zero-inflated NB; use `prob[[p]] = NULL` if outcome Y_p has no Negative Binomial variables; `length(prob[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`mu` list of length `M`, p-th component a vector of mean values for the Negative Binomial variables for outcome Y_p (see `stats::dnbinom`); order is 1st regular NB and 2nd zero-inflated NB; use `mu[[p]] = NULL` if outcome Y_p has no Negative Binomial variables; `length(mu[[p]])` can differ across outcomes; the order should be the same as in `corr.x`; for zero-inflated NB variables, this refers to the mean of the NB distribution (see `VGAM::dzinegbin`) (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture)

`p_zinb` a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see `VGAM::dzinegbin`); if `p_zinb` = 0, Y_{nb} has a regular NB distribution; if `p_zinb` is in `(-prob^size/(1 - prob^size), 0)`, Y_{nb} has a zero-deflated NB distribution and `p_zinb` is not a probability; if `p_zinb = -prob^size/(1 - prob^size)`, Y_{nb} has a positive-NB distribution (see `VGAM::dposnegbin`); order is 1st regular NB and 2nd zero-inflated NB; if a single number, all NB variables given this value; if a vector of length `M`, all NB variables in equation p given `p_zinb[p]`; otherwise, missing values are set to 0 and ordered 1st

`nb_eps` list of length `M`, p-th component a vector of length `size[[p]]` containing cumulative probability truncation values used to calculate intermediate correlations involving Negative Binomial variables; order is 1st regular NB and 2nd zero-inflated NB; if a single number, all NB variables given this value; if a vector of length `M`, all NB variables in equation p given `nb_eps[p]`; otherwise, missing values are set to 0.0001 and ordered 1st

`corr.x` list of length `M`, each component a list of length `M`; `corr.x[[p]][[q]]` is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); order: 1st ordinal (same order as in `marginal`), 2nd continuous non-mixture (same order as in `skews`), 3rd components of continuous mixture (same order as in `mix_pis`), 4th regular Poisson, 5th zero-inflated Poisson (same order as in `lam`), 6th regular NB, and 7th zero-inflated NB (same order as in `size`); if p = q, `corr.x[[p]][[q]]` is a correlation matrix with `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p; if p != q, `corr.x[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that `ncol(corr.x[[p]][[q]])` = # $X_{(qj)}$ for outcome Y_q; use `corr.x[[p]][[q]] = NULL` if equation q has no $X_{(qj)}$; use `corr.x[[p]] = NULL` if equation p has no $X_{(pj)}$

`corr.yx` input for `nonnormsys` only; a list of length `M`, where the p-th component is a 1 row matrix of correlations between Y_p and $X_{(pj)}$; if there are mixture variables and the `betas` are desired in terms of these (and not the components), then `corr.yx` should be specified in terms of correlations between outcomes and non-mixture or mixture variables, and the number of columns of the matrices of `corr.yx` should not match the dimensions of the matrices in `corr.x`; if the `betas` are desired in terms of the components, then `corr.yx` should be specified in terms of correlations between outcomes and non-mixture or components of mixture variables, and the number of columns of the matrices of `corr.yx` should match the dimensions of the matrices in `corr.x`; use `corr.yx[[p]] = NULL` if equation p has no $X_{(pj)}$

`corr.e` correlation matrix for continuous non-mixture or components of mixture error terms

`same.var` either a vector or a matrix; if a vector, `same.var` includes column numbers of `corr.x[[1]][[1]]` corresponding to independent variables that should be identical across equations; these terms must have the same indices for all `p = 1, ..., M`; i.e., if the 1st ordinal variable represents sex which should be the same for each equation, then `same.var[1] = 1` since ordinal variables are 1st in `corr.x[[1]][[1]]` and sex is the 1st ordinal variable, and the 1st term for all other outcomes must also be sex; if a matrix, columns 1 and 2 are outcome p and column index in `corr.x[[p]][[p]]` for 1st instance of variable, columns 3 and 4 are outcome q and column index in `corr.x[[q]][[q]]` for subsequent instances of variable; i.e., if 1st term for all outcomes is sex and `M = 3`, then `same.var = matrix(c(1, 1, 2, 1, 1, 1, 3, 1), 2, 4, byrow = TRUE)`; the independent variable index corresponds to ordinal, continuous non-mixture, component of continuous mixture, Poisson, or NB variable

`subj.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is independent variable index corresponding to covariate which is a a subject-level term (not including time), including time-varying covariates; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable; assumes all other variables are group-level terms; these subject-level terms are used to form interactions with the group level terms

`int.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd and 3rd columns are indices corresponding to independent variables to form interactions between; this includes all interactions that are not accounted for by a subject-group level interaction (as indicated by `subj.var`) or by a time-covariate interaction (as indicated by `tint.var`); ex: 1, 2, 3 indicates that for outcome 1, the 2nd and 3rd independent variables form an interaction term; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable

`tint.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is index of independent variable to form interaction with time; if `tint.var = NULL` or no $X_{(pj)}$ are indicated for outcome Y_p, this includes all group-level variables (variables not indicated as subject-level variables in `subj.var`), else includes only terms indicated by 2nd column (i.e., in order to include subject-level variables); ex: 1, 1 indicates that for outcome 1, the 1st independent variable has an interaction with time; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable

`betas.0` vector of length `M` containing intercepts, if `NULL` all set equal to 0; if length 1, all intercepts set to `betas.0`

`betas` list of length `M`, p-th component a vector of coefficients for outcome Y_p, including group and subject-level terms; order is order of variables in `corr.x[[p]][[p]]`; if `betas = list()`, all set to 0 so that all Y only have intercept and/or interaction terms plus error terms; if all outcomes have the same betas, use list of length 1; if Y_p only has intercept and/or interaction terms, set `betas[[p]] = NULL`; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.subj` list of length `M`, p-th component a vector of coefficients for interaction terms between group-level terms and subject-level terms given in `subj.var`; order is the same order as given in `subj.var`; if all outcomes have the same betas, use list of length 1; if Y_p only has group-level terms, set `betas.subj[[p]] = NULL`; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.int` list of length `M`, p-th component a vector of coefficients for interaction terms indicated in `int.var`; order is the same order as given in `int.var`; if all outcomes have the same betas, use list of length 1; if Y_p has none, set `betas.int[[p]] = NULL`; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.t` vector of length `M` of coefficients for time terms, if `NULL` all set equal to 1; if length 1, all intercepts set to `betas.t`

`betas.tint` list of length `M`, p-th component a vector of coefficients for interaction terms indicated in `tint.var`; order is the same order as given in `tint.var`; if all outcomes have the same betas, use list of length 1; if Y_p has none, set `betas.tint[[p]] = NULL`; if there are continuous mixture variables, beta is for mixture variable (not for components)

rand.int
: "none" (default) if no random intercept term for all outcomes, "non_mix" if all random intercepts have a continuous non-mixture distribution, "mix" if all random intercepts have a continuous mixture distribution; also can be a vector of length `M` containing a combination (i.e., `c("non_mix", "mix", "none")` if the 1st has a non-mixture distribution, the 2nd has a mixture distribution, and 3rd outcome has no random intercept)

rand.tsl
: "none" (default) if no random slope for time for all outcomes, "non_mix" if all random time slopes have a continuous non-mixture distribution, "mix" if all random time slopes have a continuous mixture distribution; also can be a vector of length `M` as in `rand.int`

rand.var
: matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is independent variable index corresponding to covariate to assign random effect to (not including the random intercept or time slope if present); the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable; order is 1st continuous non-mixture and 2nd continuous mixture random effects; note that the order of the rows corresponds to the order of the random effects in `corr.u` not the order of the independent variable so that a continuous mixture covariate with a non-mixture random effect would be ordered before a continuous non-mixture covariate with a mixture random effect (the 2nd column of `rand.var` indicates the specific covariate)

corr.u
: if the random effects are the same variables across equations, a matrix of correlations for U; if the random effects are different variables across equations, a list of length `M`, each component a list of length `M`; `corr.u[[p]][[q]]` is matrix of correlations for random effects in equations p ($U_{(pj)}$ for outcome Y_p) and q ($U_{(qj)}$ for outcome Y_q); if p = q, `corr.u[[p]][[q]]` is a correlation matrix with `nrow(corr.u[[p]][[q]])` = # $U_{(pj)}$ for outcome Y_p; if p != q, `corr.u[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to $U_{(pj)}$ for Y_p so that `nrow(corr.u[[p]][[q]])` = # $U_{(pj)}$ for outcome Y_p and columns correspond to $U_{(qj)}$ for Y_q so that `ncol(corr.u[[p]][[q]])` = # $U_{(qj)}$ for outcome Y_q; the number of random effects for Y_p is taken from `nrow(corr.u[[p]][[1]])` so that if there should be random effects, there must be entries for `corr.u`; use `corr.u[[p]][[q]] = NULL` if equation q has no $U_{(qj)}$; use `corr.u[[p]] = NULL` if equation p has no $U_{(pj)}$;

correlations are specified in terms of components of mixture variables (if present); order is 1st random intercept (if `rand.int != "none"`), 2nd random time slope (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions, 4th other random slopes with mixture distributions

quiet
: if FALSE prints messages, if TRUE suppresses messages

Value

TRUE if all inputs are correct, else it will stop with a correction message

References

Headrick TC, Beasley TM (2004). A Method for Simulating Correlated Non-Normal Systems of Linear Statistical Equations. Communications in Statistics - Simulation and Computation, 33(1). doi: 10.1081/SAC120028431

See Also

nonnormsys, corrsys, corrsys2

Examples

```
# Example: system of three equations for 2 independent variables, where each
# error term has unit variance, from Headrick & Beasley (2002)
# Y_1 = beta_10 + beta_11 * X_11 + beta_12 * X_12 + sigma_1 * e_1
# Y_2 = beta_20 + beta_21 * X_21 + beta_22 * X_22 + sigma_2 * e_2
# Y_3 = beta_30 + beta_31 * X_31 + beta_32 * X_32 + sigma_3 * e_3
# X_11 = X_21 = X_31 = Exponential(2)
# X_12 = X_22 = X_32 = Laplace(0, 1)
# e_1 = e_2 = e_3 = Cauchy(0, 1)
M <- 3
Stcum1 <- calc_theory("Exponential", 2)
Stcum2 <- calc_theory("Laplace", c(0, 1))
Stcum3 <- c(0, 1, 0, 25, 0, 1500) # taken from paper
means <- lapply(seq_len(M), function(x) c(0, 0, 0))
vars <- lapply(seq_len(M), function(x) c(1, 1, 1))
skews <- lapply(seq_len(M), function(x) c(Stcum1[3], Stcum2[3], Stcum3[3]))
skurts <- lapply(seq_len(M), function(x) c(Stcum1[4], Stcum2[4], Stcum3[4]))
fifths <- lapply(seq_len(M), function(x) c(Stcum1[5], Stcum2[5], Stcum3[5]))
sixths <- lapply(seq_len(M), function(x) c(Stcum1[6], Stcum2[6], Stcum3[6]))
corr.yx <- list(matrix(c(0.4, 0.4), 1), matrix(c(0.5, 0.5), 1),
  matrix(c(0.6, 0.6), 1))
corr.x <- list()
corr.x[[1]] <- corr.x[[2]] <- corr.x[[3]] <- list()
corr.x[[1]][[1]] <- matrix(c(1, 0.1, 0.1, 1), 2, 2)
corr.x[[1]][[2]] <- matrix(c(0.1974318, 0.1859656, 0.1879483, 0.1858601),
  2, 2, byrow = TRUE)
corr.x[[1]][[3]] <- matrix(c(0.2873190, 0.2589830, 0.2682057, 0.2589542),
  2, 2, byrow = TRUE)
corr.x[[2]][[1]] <- t(corr.x[[1]][[2]])
corr.x[[2]][[2]] <- matrix(c(1, 0.35, 0.35, 1), 2, 2)
corr.x[[2]][[3]] <- matrix(c(0.5723303, 0.4883054, 0.5004441, 0.4841808),
  2, 2, byrow = TRUE)
corr.x[[3]][[1]] <- t(corr.x[[1]][[3]])
corr.x[[3]][[2]] <- t(corr.x[[2]][[3]])
corr.x[[3]][[3]] <- matrix(c(1, 0.7, 0.7, 1), 2, 2)
corr.e <- matrix(0.4, nrow = 3, ncol = 3)
diag(corr.e) <- 1
checkpar(M, "Polynomial", "non_mix", means, vars, skews,
  skurts, fifths, sixths, corr.x = corr.x, corr.yx = corr.yx,
  corr.e = corr.e, quiet = TRUE)
```

---

corrsys — *Generate Correlated Systems of Equations with Ordinal, Continuous, and/or Count Variables: Correlation Method 1*

Description

This function generates a correlated system of M equations representing a **system of repeated measures** at M time points. The equations may contain 1) ordinal ($r \geq 2$ categories), continuous (normal, non-normal, and mixture distributions), count (regular and zero-inflated, Poisson and Negative Binomial) independent variables X; 2) continuous error terms E; 3) a discrete time variable $Time$; and 4) random effects U. The assumptions are that 1) there are at least 2 equations, 2) the independent variables, random effect terms, and error terms are uncorrelated, 3) each equation has an error term, 4) all error terms have a continuous non-mixture distribution or all have a continuous mixture distribution, 5) all random effects are continuous, and 6) growth is linear (with respect to time). The random effects may be a random intercept, a random slope for time, or a random slope for any of the X variables. Continuous variables are simulated using either Fleishman's third-order (`method = "Fleishman"`, doi: 10.1007/BF02293811) or Headrick's fifth-order (`method = "Polynomial"`, doi: 10.1016/S01679473(02)000725) power method transformation (PMT). Simulation occurs at the component-level for continuous mixture distributions. The target correlation matrix is specified in terms of correlations with components of continuous mixture variables. These components are transformed into the desired mixture variables using random multinomial variables based on the mixing probabilities. The X terms can be the same across equations (i.e., modeling sex or height) or may be time-varying covariates. The equations may contain different numbers of X terms (i.e., a covariate could be missing for a given equation).

The outcomes Y are generated using a hierarchical linear models (HLM) approach, which allows the data to be structured in at least two levels. **Level-1** is the repeated measure (time or condition) and other subject-level variables. **Level-1** is nested within **Level-2**, which describes the average of the outcome (the intercept) and growth (slope for time) as a function of group-level variables. The first level captures the within-subject variation, while the second level describes the between-subjects variability. Using a HLM provides a way to determine if: a) subjects differ at a specific time point with respect to the dependent variable, b) growth rates differ across conditions, or c) growth rates differ across subjects. Random effects describe deviation at the subject-level from the average (fixed) effect described by the slope coefficients (betas). See the **The Hierarchical Linear Models Approach for a System of Correlated Equations with Multiple Variable Types** vignette for a description of the HLM model. The user can specify subject-level X terms, and each subject-level X term is crossed with all group-level X terms. The equations may also contain interactions between X variables. Interactions specified in `int.var` between two group-level covariates are themselves considered group-level covariates and will be crossed with subject-level covariates. Interactions between two subject-level covariates are considered subject-level covariates and will be crossed with group-level covariates. Since `Time` is a subject-level variable, each group-level term is crossed with `Time` unless otherwise specified.

Random effects may be added for the intercept, time slope, or effects of any of the covariates. The type of random intercept and time slope (i.e., non-mixture or mixture) is specified in `rand.int` and `rand.tsl`. This type may vary by equation. The random effects for independent variables are specified in `rand.var` and may also contain a combination of non-mixture and mixture continuous distributions. If the parameter lists are of length `M + 1`, the random effects are the same variables across equations and the correlation for the effects `corr.u` is a matrix. If the parameter lists are of length `2 * M`, the random effects are different variables across equations and the correlation for the effects `corr.u` is a list.

The independent variables, interactions, `Time` effect, random effects, and error terms are summed together to produce the outcomes Y. The beta coefficients may be the same or differ across equations. The user specifies the betas for the independent variables in `betas`, for the interactions between two group-level or two subject-level covariates in `betas.int`, for the group-subject level interactions in `betas.subj`, and for the `Time` interactions in `betas.tint`. Setting a coefficient to 0 will eliminate that term. The user also provides the correlations 1) between E terms; 2) between X variables within each outcome Y_p, `p = 1, ..., M`, and between outcome pairs; and 3) between U variables within each outcome Y_p, `p = 1, ..., M`, and between outcome pairs. The order of the independent variables in `corr.x` must be 1st ordinal (same order as in `marginal`), 2nd continuous non-mixture (same order as in `skews`), 3rd components of continuous mixture (same order as in `mix_pis`), 4th regular Poisson, 5th zero-inflated Poisson (same order as in `lam`), 6th regular NB, and 7th zero-inflated NB (same order as in size). The order of the random effects in corr.u must be 1st random intercept, 2nd random time slope, 3rd continuous non-mixture random effects, and 4th components of continuous mixture random effects.

The variables are generated from multivariate normal variables with intermediate correlations calculated using intercorr, which employs **correlation method 1**. See SimCorrMix for a description of the correlation method and the techniques used to generate each variable type. The order of the variables returned is 1st covariates X (as specified in corr.x), 2nd group-group or subject-subject interactions (ordered as in int.var), 3rd subject-group interactions (1st by subject-level variable as specified in subj.var, 2nd by covariate as specified in corr.x), and 4th time interactions (either as specified in corr.x with group-level covariates or in tint.var).

This function contains no parameter checks in order to decrease simulation time. That should be done first using checkpar. Summaries of the system can be obtained using summary_sys. The **Correlated Systems of Statistical Equations with Multiple Variable Types** demonstrates examples.

Usage

```
corrsys(n = 10000, M = NULL, Time = NULL, method = c("Fleishman",
  "Polynomial"), error_type = c("non_mix", "mix"), means = list(),
  vars = list(), skews = list(), skurts = list(), fifths = list(),
  sixths = list(), Six = list(), mix_pis = list(), mix_mus = list(),
  mix_sigmas = list(), mix_skews = list(), mix_skurts = list(),
  mix_fifths = list(), mix_sixths = list(), mix_Six = list(),
  marginal = list(), support = list(), lam = list(), p_zip = list(),
  size = list(), prob = list(), mu = list(), p_zinb = list(),
  corr.x = list(), corr.e = NULL, same.var = NULL, subj.var = NULL,
  int.var = NULL, tint.var = NULL, betas.0 = NULL, betas = list(),
  betas.subj = list(), betas.int = list(), betas.t = NULL,
  betas.tint = list(), rand.int = c("none", "non_mix", "mix"),
  rand.tsl = c("none", "non_mix", "mix"), rand.var = NULL,
  corr.u = list(), seed = 1234, use.nearPD = TRUE, eigmin = 0,
  adjgrad = FALSE, B1 = NULL, tau = 0.5, tol = 0.1, steps = 100,
  msteps = 10, nrand = 1e+05, errorloop = FALSE, epsilon = 0.001,
  maxit = 1000, quiet = FALSE)
```

Arguments

n	the sample size (i.e. the length of each simulated variable; default = 10000)
M	the number of dependent variables Y (outcomes); equivalently, the number of equations in the system
Time	a vector of values to use for time; each subject receives the same time value; if NULL, Time = 1:M
method	the PMT method used to generate all continuous variables, including independent variables (covariates), error terms, and random effects; "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation
error_type	"non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions
means	if no random effects, a list of length M where means[[p]] contains a vector of means for the continuous independent variables in equation p with non-mixture

(X_{cont}) or mixture (X_{mix}) distributions and for the error terms (E); order in vector is X_{cont}, X_{mix}, E

if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `means[M + 1]` or `means[(M + 1):(2 * M)]` are vectors of means for all random effects with continuous non-mixture or mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int != "none"`), 2nd random time slope U_1 (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions U_{cont}, 4th other random slopes with mixture distributions U_{mix}

`vars` a list of same length and order as `means` containing vectors of variances for the continuous variables, error terms, and any random effects

`skews` if no random effects, a list of length `M` where `skews[[p]]` contains a vector of skew values for the continuous independent variables in equation p with non-mixture (X_{cont}) distributions and for E if `error_type = "non_mix"`; order in vector is X_{cont}, E

if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `skews[M + 1]` or `skews[(M + 1):(2 * M)]` are vectors of skew values for all random effects with continuous non-mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int = "non_mix"`), 2nd random time slope U_1 (if `rand.tsl = "non_mix"`), 3rd other random slopes with non-mixture distributions U_{cont}

`skurts` a list of same length and order as `skews` containing vectors of standardized kurtoses (kurtosis - 3) for the continuous variables, error terms, and any random effects with non-mixture distributions

`fifths` a list of same length and order as `skews` containing vectors of standardized fifth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for `method = "Fleishman"`

`sixths` a list of same length and order as `skews` containing vectors of standardized sixth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for `method = "Fleishman"`

`Six` a list of length `M`, `M + 1`, or `2 * M`, where `Six[1:M]` are for X_{cont}, E (if `error_type = "non_mix"`) and `Six[M + 1]` or `Six[(M + 1):(2 * M)]` are for non-mixture U; if `error_type = "mix"` and there are only random effects (i.e., `length(corr.x) = 0`), use `Six[1:M] = rep(list(NULL), M)` so that `Six[M + 1]` or `Six[(M + 1):(2 * M)]` describes the non-mixture U;

`Six[[p]][[j]]` is a vector of sixth cumulant correction values to aid in finding a valid PDF for $X_{cont(pj)}$, the j-th continuous non-mixture covariate for outcome Y_p; the last vector in `Six[[p]]` is for E_p (if `error_type = "non_mix"`); use `Six[[p]][[j]] = NULL` if no correction desired for $X_{cont(pj)}$; use `Six[[p]] = NULL` if no correction desired for any continuous non-mixture covariate or error term in equation p

`Six[[M + p]][[j]]` is a vector of sixth cumulant correction values to aid in finding a valid PDF for $U_{(pj)}$, the j-th non-mixture random effect for outcome Y_p; use `Six[[M + p]][[j]] = NULL` if no correction desired for $U_{(pj)}$; use `Six[[M + p]] = NULL` if no correction desired for any continuous non-mixture random effect in equation p

keep `Six = list()` if no corrections desired for all equations or if `method = "Fleishman"`

`mix_pis` list of length `M`, `M + 1` or `2 * M`, where `mix_pis[1:M]` are for X_{cont}, E (if `error_type = "mix"`) and `mix_pis[M + 1]` or `mix_pis[(M + 1):(2 * M)]` are for mixture U; use `mix_pis[[p]] = NULL` if equation p has no continuous mixture terms if error_type = "non_mix" and there are only random effects (i.e., length(corr.x) = 0), use mix_pis[1:M] = NULL so that mix_pis[M + 1] or mix_pis[(M + 1):(2 * M)] describes the mixture U;

mix_pis[[p]][[j]] is a vector of mixing probabilities of the component distributions for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last vector in mix_pis[[p]] is for E_p (if error_type = "mix"); components should be ordered as in corr.x

mix_pis[[M + p]][[j]] is a vector of mixing probabilities of the component distributions for $U_{(pj)}$, the j-th random effect with a mixture distribution for outcome Y_p; order is 1st random intercept (if rand.int = "mix"), 2nd random time slope (if rand.tsl = "mix"), 3rd other random slopes with mixture distributions; components should be ordered as in corr.u

mix_mus
: list of same length and order as mix_pis;

  mix_mus[[p]][[j]] is a vector of means of the component distributions for $X_{mix(pj)}$, the last vector in mix_mus[[p]] is for E_p (if error_type = "mix")

  mix_mus[[p]][[j]] is a vector of means of the component distributions for $U_{mix(pj)}$

mix_sigmas
: list of same length and order as mix_pis;

  mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $X_{mix(pj)}$, the last vector in mix_sigmas[[p]] is for E_p (if error_type = "mix")

  mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $U_{mix(pj)}$

mix_skews
: list of same length and order as mix_pis;

  mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $X_{mix(pj)}$, the last vector in mix_skews[[p]] is for E_p (if error_type = "mix")

  mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $U_{mix(pj)}$

mix_skurts
: list of same length and order as mix_pis;

  mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $X_{mix(pj)}$, the last vector in mix_skurts[[p]] is for E_p (if error_type = "mix")

  mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $U_{mix(pj)}$

mix_fifths
: list of same length and order as mix_pis; not necessary for method = "Fleishman";

  mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_fifths[[p]] is for E_p (if error_type = "mix")

  mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $U_{mix(pj)}$

mix_sixths
: list of same length and order as mix_pis; not necessary for method = "Fleishman";

  mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_sixths[[p]] is for E_p (if error_type = "mix")

  mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $U_{mix(pj)}$

mix_Six
: a list of same length and order as mix_pis; keep mix_Six = list() if no corrections desired for all equations or if method = "Fleishman"

p-th component of `mix_Six[1:M]` is a list of length equal to the total number of component distributions for the $X_{mix(p)}$ and E_p (if `error_type = "mix"`); `mix_Six[[p]][[j]]` is a vector of sixth cumulant corrections for the j-th component distribution (i.e., if there are 2 continuous mixture independent variables for Y_p, where $X_{mix(p1)}$ has 2 components and $X_{mix(p2)}$ has 3 components, then `length(mix_Six[[p]]) = 5` and `mix_Six[[p]][[3]]` would correspond to the 1st component of $X_{mix(p2)}$); use `mix_Six[[p]][[j]] = NULL` if no correction desired for that component; use `mix_Six[[p]] = NULL` if no correction desired for any component of $X_{mix(p)}$ and E_p

q-th component of `mix_Six[M + 1]` or `mix_Six[(M + 1):(2 * M)]` is a list of length equal to the total number of component distributions for the $U_{mix(q)}$; `mix_Six[[q]][[j]]` is a vector of sixth cumulant corrections for the j-th component distribution; use `mix_Six[[q]][[j]] = NULL` if no correction desired for that component; use `mix_Six[[q]] = NULL` if no correction desired for any component of $U_{mix(q)}$

`marginal` a list of length `M`, with the p-th component a list of cumulative probabilities for the ordinal variables associated with outcome Y_p (use `marginal[[p]] = NULL` if outcome Y_p has no ordinal variables); `marginal[[p]][[j]]` is a vector of the cumulative probabilities defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1); for binary variables, the probability is the probability of the 1st category, which has the smaller support value; `length(marginal[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`support` a list of length `M`, with the p-th component a list of support values for the ordinal variables associated with outcome Y_p; use `support[[p]] = NULL` if outcome Y_p has no ordinal variables; `support[[p]][[j]]` is a vector of the support values defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if not provided, the default for r categories is 1, ..., r

`lam` list of length `M`, p-th component a vector of lambda (means > 0) values for Poisson variables for outcome Y_p (see `stats::dpois`); order is 1st regular Poisson and 2nd zero-inflated Poisson; use `lam[[p]] = NULL` if outcome Y_p has no Poisson variables; `length(lam[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`p_zip` a list of vectors of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `VGAM::dzipois`); if `p_zip` = 0, Y_{pois} has a regular Poisson distribution; if `p_zip` is in (0, 1), Y_{pois} has a zero-inflated Poisson distribution; if `p_zip` is in `(-(exp(lam) - 1)^(-1), 0)`, Y_{pois} has a zero-deflated Poisson distribution and `p_zip` is not a probability; if `p_zip = -(exp(lam) - 1)^(-1)`, Y_{pois} has a positive-Poisson distribution (see `VGAM::dpospois`); order is 1st regular Poisson and 2nd zero-inflated Poisson; if a single number, all Poisson variables given this value; if a vector of length `M`, all Poisson variables in equation p given `p_zip[p]`; otherwise, missing values are set to 0 and ordered 1st

`size` list of length `M`, p-th component a vector of size parameters for the Negative Binomial variables for outcome Y_p (see `stats::dnbinom`); order is 1st regular NB and 2nd zero-inflated NB; use `size[[p]] = NULL` if outcome Y_p has no Negative Binomial variables; `length(size[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`prob` list of length `M`, p-th component a vector of success probabilities for the Negative Binomial variables for outcome Y_p (see `stats::dnbinom`); order is 1st regular

NB and 2nd zero-inflated NB; use `prob[[p]] = NULL` if outcome Y_p has no Negative Binomial variables; `length(prob[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`mu`
: list of length `M`, p-th component a vector of mean values for the Negative Binomial variables for outcome Y_p (see `stats::dnbinom`); order is 1st regular NB and 2nd zero-inflated NB; use `mu[[p]] = NULL` if outcome Y_p has no Negative Binomial variables; `length(mu[[p]])` can differ across outcomes; the order should be the same as in `corr.x`; for zero-inflated NB variables, this refers to the mean of the NB distribution (see `VGAM::dzinegbin`) (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture)

`p_zinb`
: a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see `VGAM::dzinegbin`); if `p_zinb` = 0, Y_{nb} has a regular NB distribution; if `p_zinb` is in `(-prob^size/(1 - prob^size), 0)`, Y_{nb} has a zero-deflated NB distribution and `p_zinb` is not a probability; if `p_zinb = -prob^size/(1 - prob^size)`, Y_{nb} has a positive-NB distribution (see `VGAM::dposnegbin`); order is 1st regular NB and 2nd zero-inflated NB; if a single number, all NB variables given this value; if a vector of length `M`, all NB variables in equation p given `p_zinb[p]`; otherwise, missing values are set to 0 and ordered 1st

`corr.x`
: list of length `M`, each component a list of length `M`; `corr.x[[p]][[q]]` is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); order: 1st ordinal (same order as in `marginal`), 2nd continuous non-mixture (same order as in `skews`), 3rd components of continuous mixture (same order as in `mix_pis`), 4th regular Poisson, 5th zero-inflated Poisson (same order as in `lam`), 6th regular NB, and 7th zero-inflated NB (same order as in `size`); if p = q, `corr.x[[p]][[q]]` is a correlation matrix with `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p; if p != q, `corr.x[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that `nrow(corr.x[[p]][[q]])` = # $X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that `ncol(corr.x[[p]][[q]])` = # $X_{(qj)}$ for outcome Y_q; use `corr.x[[p]][[q]] = NULL` if equation q has no $X_{(qj)}$; use `corr.x[[p]] = NULL` if equation p has no $X_{(pj)}$

`corr.e`
: correlation matrix for continuous non-mixture or components of mixture error terms

`same.var`
: either a vector or a matrix; if a vector, `same.var` includes column numbers of `corr.x[[1]][[1]]` corresponding to independent variables that should be identical across equations; these terms must have the same indices for all `p = 1, ..., M`; i.e., if the 1st ordinal variable represents sex which should be the same for each equation, then `same.var[1] = 1` since ordinal variables are 1st in `corr.x[[1]][[1]]` and sex is the 1st ordinal variable, and the 1st term for all other outcomes must also be sex; if a matrix, columns 1 and 2 are outcome p and column index in `corr.x[[p]][[p]]` for 1st instance of variable, columns 3 and 4 are outcome q and column index in `corr.x[[q]][[q]]` for subsequent instances of variable; i.e., if 1st term for all outcomes is sex and `M = 3`, then `same.var = matrix(c(1, 1, 2, 1, 1, 1, 3, 1), 2, 4, byrow = TRUE)`; the independent variable index corresponds to ordinal, continuous non-mixture, **component** of continuous mixture, Poisson, or NB variable

`subj.var`
: matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is independent variable index corresponding to covariate which is a a subject-level term (not including time), including time-varying covariates; the independent variable index corresponds to ordinal, continuous non-mixture, continuous

mixture (not mixture component), Poisson, or NB variable; assumes all other variables are group-level terms; these subject-level terms are used to form interactions with the group level terms

`int.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd and 3rd columns are indices corresponding to two group-level or two subject-level independent variables to form interactions between; this includes all interactions that are not accounted for by a subject-group level interaction (as indicated by `subj.var`) or by a time-covariate interaction (as indicated by `tint.var`); ex: 1, 2, 3 indicates that for outcome 1, the 2nd and 3rd independent variables form an interaction term; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable

`tint.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is index of independent variable to form interaction with time; if `tint.var = NULL` or no $X_{(pj)}$ are indicated for outcome Y_p, all group-level variables (variables not indicated as subject-level variables in `subj.var`) will be crossed with time, else includes only terms indicated by 2nd column (i.e., in order to include subject-level variables); ex: 1, 1 indicates that for outcome 1, the 1st independent variable has an interaction with time; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable

`betas.0` vector of length `M` containing intercepts, if `NULL` all set equal to 0; if length 1, all intercepts set to `betas.0`

`betas` list of length `M`, p-th component a vector of coefficients for outcome Y_p, including group and subject-level terms; order is order of variables in `corr.x[[p]][[p]]`; if `betas = list()`, all set to 0 so that all Y only have intercept and/or interaction terms plus error terms; if all outcomes have the same betas, use list of length 1; if Y_p only has intercept and/or interaction terms, set `betas[[p]] = NULL`; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.subj` list of length `M`, p-th component a vector of coefficients for interaction terms between group-level terms and subject-level terms given in `subj.var`; order is 1st by subject-level covariate as given in `subj.var` and 2nd by group-level covariate as given in `corr.x` or an interaction between group-level terms; if all outcomes have the same betas, use list of length 1; if Y_p only has group-level terms, set `betas.subj[[p]] = NULL`; since subject-subject interactions are treated as subject-level variables, these will also be crossed with all group-level variables and require coefficients; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.int` list of length `M`, p-th component a vector of coefficients for interaction terms indicated in `int.var`; order is the same order as given in `int.var`; if all outcomes have the same betas, use list of length 1; if Y_p has none, set `betas.int[[p]] = NULL`; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.t` vector of length `M` of coefficients for time terms, if `NULL` all set equal to 1; if length 1, all intercepts set to `betas.t`

`betas.tint` list of length `M`, p-th component a vector of coefficients for all interactions with time; this includes interactions with group-level covariates or terms indicated in `tint.var`; order is the same order as given in `corr.x` or `tint.var`; if all outcomes have the same betas, use list of length 1; if Y_p has none, set `betas.tint[[p]] = NULL`; since group-group interactions are treated as group-level variables, these will also be crossed with time (unless otherwise specified for that outcome in `tint.var`) and require coefficients; if there are continuous mixture variables, beta is for mixture variable (not for components)

`rand.int`
: "none" (default) if no random intercept term for all outcomes, "non_mix" if all random intercepts have a continuous non-mixture distribution, "mix" if all random intercepts have a continuous mixture distribution; also can be a vector of length `M` containing a combination (i.e., `c("non_mix", "mix", "none")` if the 1st has a non-mixture distribution, the 2nd has a mixture distribution, and 3rd outcome has no random intercept)

`rand.tsl`
: "none" (default) if no random slope for time for all outcomes, "non_mix" if all random time slopes have a continuous non-mixture distribution, "mix" if all random time slopes have a continuous mixture distribution; also can be a vector of length `M` as in `rand.int`

`rand.var`
: matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is independent variable index corresponding to covariate to assign random effect to (not including the random intercept or time slope if present); the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable; order is 1st continuous non-mixture and 2nd continuous mixture random effects; note that the order of the rows corresponds to the order of the random effects in `corr.u` not the order of the independent variable so that a continuous mixture covariate with a non-mixture random effect would be ordered before a continuous non-mixture covariate with a mixture random effect (the 2nd column of `rand.var` indicates the specific covariate)

`corr.u`
: if the random effects are the same variables across equations, a matrix of correlations for U; if the random effects are different variables across equations, a list of length `M`, each component a list of length `M`; `corr.u[[p]][[q]]` is matrix of correlations for random effects in equations p ($U_{(pj)}$ for outcome Y_p) and q ($U_{(qj)}$ for outcome Y_q); if p = q, `corr.u[[p]][[q]]` is a correlation matrix with `nrow(corr.u[[p]][[q]])` = # $U_{(pj)}$ for outcome Y_p; if p != q, `corr.u[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to $U_{(pj)}$ for Y_p so that `nrow(corr.u[[p]][[q]])` = # $U_{(pj)}$ for outcome Y_p and columns correspond to $U_{(qj)}$ for Y_q so that `ncol(corr.u[[p]][[q]])` = # $U_{(qj)}$ for outcome Y_q; the number of random effects for Y_p is taken from `nrow(corr.u[[p]][[1]])` so that if there should be random effects, there must be entries for `corr.u`; use `corr.u[[p]][[q]] = NULL` if equation q has no $U_{(qj)}$; use `corr.u[[p]] = NULL` if equation p has no $U_{(pj)}$;
: correlations are specified in terms of components of mixture variables (if present); order is 1st random intercept (if `rand.int != "none"`), 2nd random time slope (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions, 4th other random slopes with mixture distributions

`seed`
: the seed value for random number generation (default = 1234)

`use.nearPD`
: TRUE to convert the overall intermediate correlation matrix formed by the X (for all outcomes and independent variables), E, or the random effects to the nearest positive definite matrix with `Matrix::nearPD` if necessary; if FALSE and `adjgrad = FALSE` the negative eigenvalues are replaced with `eigmin` if necessary

`eigmin`
: minimum replacement eigenvalue if overall intermediate correlation matrix is not positive-definite (default = 0)

adjgrad	TRUE to use adj_grad to convert overall intermediate correlation matrix to a positive-definite matrix and next 5 inputs can be used
B1	the initial matrix for algorithm; if NULL, uses a scaled initial matrix with diagonal elements sqrt(nrow(Sigma))/2
tau	parameter used to calculate theta (default = 0.5)
tol	maximum error for Frobenius norm distance between new matrix and original matrix (default = 0.1)
steps	maximum number of steps for k (default = 100)
msteps	maximum number of steps for m (default = 10)
nrand	the number of random numbers to generate in calculating intermediate correlations (default = 10000)
errorloop	if TRUE, uses corr_error to attempt to correct the correlation of the independent variables within and across outcomes to be within epsilon of the target correlations corr.x until the number of iterations reaches maxit (default = FALSE)
epsilon	the maximum acceptable error between the final and target correlation matrices (default = 0.001) in the calculation of ordinal intermediate correlations with ord_norm or in the error loop
maxit	the maximum number of iterations to use (default = 1000) in the calculation of ordinal intermediate correlations with ord_norm or in the error loop
quiet	if FALSE prints messages, if TRUE suppresses messages

Value

A list with the following components:

Y matrix with n rows and M columns of outcomes

X list of length M containing $X_{ord(pj)}, X_{cont(pj)}, X_{comp(pj)}, X_{pois(pj)}, X_{nb(pj)}$

X_all list of length M containing $X_{ord(pj)}, X_{cont(pj)}, X_{mix(pj)}, X_{pois(pj)}, X_{nb(pj)}$, X interactions as indicated by int.var, subject-group level term interactions as indicated by subj.var, $Time_p$, and $Time$ interactions as indicated by tint.var; order is 1st covariates X (as specified in corr.x), 2nd group-group or subject-subject interactions (ordered as in int.var), 3rd subject-group interactions (1st by subject-level variable as specified in subj.var, 2nd by covariate as specified in corr.x), and 4th time interactions (either as specified in corr.x with group-level covariates or in tint.var)

E matrix with n rows containing continuous non-mixture or components of continuous mixture error terms

E_mix matrix with n rows containing continuous mixture error terms

Sigma_X0 matrix of intermediate correlations calculated by intercorr

Sigma_X matrix of intermediate correlations after nearPD or adj_grad function has been used; applied to generate the normal variables transformed to get the desired distributions

Error_Time the time in minutes required to use the error loop

Time the total simulation time in minutes

niter a matrix of the number of iterations used in the error loop

If **continuous variables** are produced: constants a list of maximum length 2 * M, the 1st M components are data.frames of the constants for the $X_{cont(pj)}$, $X_comp(pj)$ and E_p, the 2nd M components are for random effects (if present), `SixCorr` a list of maximum length `2 * M`, the 1st `M` components are lists of sixth cumulant correction values used to obtain valid *pdf*'s for the $X_{cont(pj)}$, $X_comp(pj)$, and E_p, the 2nd `M` components are for random effects (if present),

`valid.pdf` a list of maximum length `2 * M` of vectors where the i-th element is "TRUE" if the constants for the i-th continuous variable generate a valid pdf, else "FALSE"; the 1st `M` components are for the $X_{cont(pj)}$, $X_comp(pj)$, and E_p, the 2nd `M` components are for random effects (if present)

If **random effects** are produced: `U` a list of length `M` containing matrices of continuous non-mixture and components of mixture random effects,

`U_all` a list of length `M` containing matrices of continuous non-mixture and mixture random effects,

`V` a list of length `M` containing matrices of design matrices for random effects,

`rmeans2` and `rvars2` the means and variances of the non-mixture and components reordered in accordance with the random intercept and time slope types (input for `summary_sys`)

Reasons for Function Errors

1) The most likely cause for function errors is that the parameter inputs are mispecified. Using `checkpar` prior to simulation can help decrease these errors.

2) Another reason for error is that no solutions to `fleish` or `poly` converged when using `find_constants`. If this happens, the simulation will stop. It may help to first use `find_constants` for each continuous variable to determine if a sixth cumulant correction value is needed. If the standardized cumulants are obtained from `calc_theory`, the user may need to use rounded values as inputs (i.e. `skews = round(skews, 8)`). For example, in order to ensure that skew is exactly 0 for symmetric distributions.

3) The kurtosis for a continuous variable may be outside the region of possible values. There is an associated lower kurtosis boundary for associated with a given skew (for Fleishman's method) or skew and fifth and sixth cumulants (for Headrick's method). Use `calc_lower_skurt` to determine the boundary for a given set of cumulants.

References

Amatya A & Demirtas H (2015). Simultaneous generation of multivariate mixed data with Poisson and normal marginals. Journal of Statistical Computation and Simulation, 85(15):3129-39. doi: 10.1080/00949655.2014.953534.

Barbiero A & Ferrari PA (2015). GenOrd: Simulation of Discrete Random Variables with Given Correlation Matrix and Marginal Distributions. R package version 1.4.0.
https://CRAN.R-project.org/package=GenOrd

Davenport JW, Bezder JC, & Hathaway RJ (1988). Parameter Estimation for Finite Mixture Distributions. Computers & Mathematics with Applications, 15(10):819-28.

Demirtas H (2006). A method for multivariate ordinal data generation given marginal distributions and correlations. Journal of Statistical Computation and Simulation, 76(11):1017-1025.
doi: 10.1080/10629360600569246.

Demirtas H (2014). Joint Generation of Binary and Nonnormal Continuous Data. Biometrics & Biostatistics, S12.

Demirtas H, Hedeker D, & Mermelstein RJ (2012). Simulation of massive public health data by power polynomials. Statistics in Medicine, 31(27):3337-3346. doi: 10.1002/sim.5362.

Everitt BS (1996). An Introduction to Finite Mixture Distributions. Statistical Methods in Medical Research, 5(2):107-127. doi: 10.1177/096228029600500202.

Ferrari PA & Barbiero A (2012). Simulating ordinal data. Multivariate Behavioral Research, 47(4): 566-589. doi: 10.1080/00273171.2012.692630.

Fialkowski AC (2017). SimMultiCorrData: Simulation of Correlated Data with Multiple Variable Types. R package version 0.2.1. https://CRAN.R-project.org/package=SimMultiCorrData.

Fialkowski AC (2018). SimCorrMix: Simulation of Correlated Data of Multiple Variable Types including Continuous and Count Mixture Distributions. R package version 0.1.0. https://github.com/AFialkowski/SimCorrMix

Fleishman AI (1978). A Method for Simulating Non-normal Distributions. Psychometrika, 43:521-532. doi: 10.1007/BF02293811.

Headrick TC (2002). Fast Fifth-order Polynomial Transforms for Generating Univariate and Multivariate Non-normal Distributions. Computational Statistics & Data Analysis, 40(4):685-711. doi: 10.1016/S01679473(02)000725. (ScienceDirect)

Headrick TC (2004). On Polynomial Transformations for Simulating Multivariate Nonnormal Distributions. Journal of Modern Applied Statistical Methods, 3(1):65-71. doi: 10.22237/jmasm/1083370080.

Headrick TC, Kowalchuk RK (2007). The Power Method Transformation: Its Probability Density Function, Distribution Function, and Its Further Use for Fitting Data. Journal of Statistical Computation and Simulation, 77:229-249. doi: 10.1080/10629360600605065.

Headrick TC, Sawilowsky SS (1999). Simulating Correlated Non-normal Distributions: Extending the Fleishman Power Method. Psychometrika, 64:25-35. doi: 10.1007/BF02294317.

Headrick TC, Sheng Y, & Hodis FA (2007). Numerical Computing and Graphics for the Power Method Transformation Using Mathematica. Journal of Statistical Software, 19(3):1 - 17. doi: 10.18637/jss.v019.i03.

Higham N (2002). Computing the nearest correlation matrix - a problem from finance; IMA Journal of Numerical Analysis 22:329-343.

Ismail N & Zamani H (2013). Estimation of Claim Count Data Using Negative Binomial, Generalized Poisson, Zero-Inflated Negative Binomial and Zero-Inflated Generalized Poisson Regression Models. Casualty Actuarial Society E-Forum, 41(20):1-28.

Kincaid C (2005). Guidelines for Selecting the Covariance Structure in Mixed Model Analysis. Computational Statistics and Data Analysis, 198(30):1-8.

Lambert D (1992). Zero-Inflated Poisson Regression, with an Application to Defects in Manufacturing. Technometrics 34(1):1-14.

Lininger M, Spybrook J, & Cheatham CC (2015). Hierarchical Linear Model: Thinking Outside the Traditional Repeated-Measures Analysis-of-Variance Box. Journal of Athletic Training, 50(4):438-441. doi: 10.4085/1062605049.5.09.

McCulloch CE, Searle SR, Neuhaus JM (2008). *Generalized, Linear, and Mixed Models* (2nd ed.). Wiley Series in Probability and Statistics. Hoboken, New Jersey: John Wiley & Sons, Inc.

Olsson U, Drasgow F, & Dorans NJ (1982). The Polyserial Correlation Coefficient. Psychometrika, 47(3):337-47. doi: 10.1007/BF02294164.

Pearson RK (2011). Exploring Data in Engineering, the Sciences, and Medicine. In. New York: Oxford University Press.

Schork NJ, Allison DB, & Thiel B (1996). Mixture Distributions in Human Genetics Research. Statistical Methods in Medical Research, 5:155-178. doi: 10.1177/096228029600500204.

Vale CD & Maurelli VA (1983). Simulating Multivariate Nonnormal Distributions. Psychometrika, 48:465-471. doi: 10.1007/BF02293687.

Van Der Leeden R (1998). Multilevel Analysis of Repeated Measures Data. Quality & Quantity, 32(1):15-29.

Yahav I & Shmueli G (2012). On Generating Multivariate Poisson Data in Management Science Applications. Applied Stochastic Models in Business and Industry, 28(1):91-102. doi: 10.1002/asmb.901.

Yee TW (2017). VGAM: Vector Generalized Linear and Additive Models. https://CRAN.R-project.org/package=VGAM.

Zhang X, Mallick H, & Yi N (2016). Zero-Inflated Negative Binomial Regression for Differential Abundance Testing in Microbiome Studies. Journal of Bioinformatics and Genomics 2(2):1-9. doi: 10.18454/jbg.2016.2.2.1.

See Also

find_constants, intercorr, checkpar, summary_sys

Examples

```
M <- 3
B <- calc_theory("Beta", c(4, 1.5))
skews <- lapply(seq_len(M), function(x) B[3])
skurts <- lapply(seq_len(M), function(x) B[4])
fifths <- lapply(seq_len(M), function(x) B[5])
sixths <- lapply(seq_len(M), function(x) B[6])
Six <- lapply(seq_len(M), function(x) list(0.03))
corr.e <- matrix(c(1, 0.4, 0.4^2, 0.4, 1, 0.4, 0.4^2, 0.4, 1), M, M,
  byrow = TRUE)
means <- lapply(seq_len(M), function(x) B[1])
vars <- lapply(seq_len(M), function(x) B[2]^2)
marginal <- list(0.3, 0.4, 0.5)
support <- lapply(seq_len(M), function(x) list(0:1))
corr.x <- list(list(matrix(1, 1, 1), matrix(0.4, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(1, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(0.4, 1, 1), matrix(1, 1, 1)))
betas <- list(0.5)
betas.t <- 1
betas.tint <- list(0.25)
Sys1 <- corrsys(10000, M, Time = 1:M, "Polynomial", "non_mix", means, vars,
  skews, skurts, fifths, sixths, Six, marginal = marginal, support = support,
  corr.x = corr.x, corr.e = corr.e, betas = betas, betas.t = betas.t,
  betas.tint = betas.tint, quiet = TRUE)

## Not run:
seed <- 276
n <- 10000
M <- 3
Time <- 1:M

# Error terms have a beta(4, 1.5) distribution with an AR(1, p = 0.4)
correlation structure
B <- calc_theory("Beta", c(4, 1.5))
skews <- lapply(seq_len(M), function(x) B[3])
skurts <- lapply(seq_len(M), function(x) B[4])
fifths <- lapply(seq_len(M), function(x) B[5])
sixths <- lapply(seq_len(M), function(x) B[6])
Six <- lapply(seq_len(M), function(x) list(0.03))
error_type <- "non_mix"
corr.e <- matrix(c(1, 0.4, 0.4^2, 0.4, 1, 0.4, 0.4^2, 0.4, 1), M, M,
  byrow = TRUE)
```

```
1 continuous mixture of Normal(-2, 1) and Normal(2, 1) for each Y
mix_pis <- lapply(seq_len(M), function(x) list(c(0.4, 0.6)))
mix_mus <- lapply(seq_len(M), function(x) list(c(-2, 2)))
mix_sigmas <- lapply(seq_len(M), function(x) list(c(1, 1)))
mix_skews <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_skurts <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_fifths <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_sixths <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_Six <- list()
Nstcum <- calc_mixmoments(mix_pis[[1]][[1]], mix_mus[[1]][[1]],
  mix_sigmas[[1]][[1]], mix_skews[[1]][[1]], mix_skurts[[1]][[1]],
  mix_fifths[[1]][[1]], mix_sixths[[1]][[1]])

means <- lapply(seq_len(M), function(x) c(Nstcum[1], B[1]))
vars <- lapply(seq_len(M), function(x) c(Nstcum[2]^2, B[2]^2))

# 1 binary variable for each Y
marginal <- lapply(seq_len(M), function(x) list(0.4))
support <- list(NULL, list(c(0, 1)), NULL)

# 1 Poisson variable for each Y
lam <- list(1, 5, 10)
# Y2 and Y3 have zero-inflated Poisson variables
p_zip <- list(NULL, 0.05, 0.1)

# 1 NB variable for each Y
size <- list(10, 15, 20)
prob <- list(0.3, 0.4, 0.5)
# either prob or mu is required (not both)
mu <- mapply(function(x, y) x * (1 - y)/y, size, prob, SIMPLIFY = FALSE)
# Y2 and Y3 have zero-inflated NB variables
p_zinb <- list(NULL, 0.05, 0.1)

# The 2nd (the normal mixture) variable is the same across Y
same.var <- 2

# Create the correlation matrix in terms of the components of the normal
# mixture
K <- 5
corr.x <- list()
corr.x[[1]] <- list(matrix(0.1, K, K), matrix(0.2, K, K), matrix(0.3, K, K))
diag(corr.x[[1]][[1]]) <- 1
# set correlation between components to 0
corr.x[[1]][[1]][2:3, 2:3] <- diag(2)
# set correlations with the same variable equal across outcomes
corr.x[[1]][[2]][, same.var] <- corr.x[[1]][[3]][, same.var] <-
  corr.x[[1]][[1]][, same.var]
corr.x[[2]] <- list(t(corr.x[[1]][[2]]), matrix(0.35, K, K),
  matrix(0.4, K, K))
  diag(corr.x[[2]][[2]]) <- 1
  corr.x[[2]][[2]][2:3, 2:3] <- diag(2)
corr.x[[2]][[2]][, same.var] <- corr.x[[2]][[3]][, same.var] <-
  t(corr.x[[1]][[2]][same.var, ])
corr.x[[2]][[3]][same.var, ] <- corr.x[[1]][[3]][same.var, ]
corr.x[[2]][[2]][same.var, ] <- t(corr.x[[2]][[2]][, same.var])
corr.x[[3]] <- list(t(corr.x[[1]][[3]]), t(corr.x[[2]][[3]]),
```

```
  matrix(0.5, K, K))
diag(corr.x[[3]][[3]]) <- 1
corr.x[[3]][[3]][2:3, 2:3] <- diag(2)
corr.x[[3]][[3]][, same.var] <- t(corr.x[[1]][[3]][same.var, ])
corr.x[[3]][[3]][same.var, ] <- t(corr.x[[3]][[3]][, same.var])

# The 2nd and 3rd variables of each Y are subject-level variables
subj.var <- matrix(c(1, 2, 1, 3, 2, 2, 2, 3, 3, 2, 3, 3), 6, 2, byrow = TRUE)
int.var <- tint.var <- NULL
betas.0 <- 0
betas <- list(seq(0.5, 0.5 + (K - 2) * 0.25, 0.25))
betas.subj <- list(seq(0.5, 0.5 + (K - 2) * 0.1, 0.1))
betas.int <- list()
betas.t <- 1
betas.tint <- list(c(0.25, 0.5))

method <- "Polynomial"

# Check parameter inputs
checkpar(M, method, error_type, means, vars, skews, skurts, fifths, sixths,
  Six, mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts, mix_fifths,
  mix_sixths, mix_Six, marginal, support, lam, p_zip, pois_eps = list(),
  size, prob, mu, p_zinb, nb_eps = list(), corr.x, corr.yx = list(),
  corr.e, same.var, subj.var, int.var, tint.var, betas.0, betas,
  betas.subj, betas.int, betas.t, betas.tint)

# Simulated system using correlation method 1
N <- corrsys(n, M, Time, method, error_type, means, vars, skews, skurts,
  fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts,
  mix_fifths, mix_sixths, mix_Six, marginal, support, lam, p_zip, size,
  prob, mu, p_zinb, corr.x, corr.e, same.var, subj.var, int.var, tint.var,
  betas.0, betas, betas.subj, betas.int, betas.t, betas.tint, seed = seed,
  use.nearPD = FALSE)

# Summarize the results
S <- summary_sys(N$Y, N$E, E_mix = NULL, N$X, N$X_all, M, method, means,
  vars, skews, skurts, fifths, sixths, mix_pis, mix_mus, mix_sigmas,
  mix_skews, mix_skurts, mix_fifths, mix_sixths, marginal, support, lam,
  p_zip, size, prob, mu, p_zinb, corr.x, corr.e)

## End(Not run)
```

`corrsys2` *Generate Correlated Systems of Equations with Ordinal, Continuous, and/or Count Variables: Correlation Method 2*

Description

This function generates a correlated system of `M` equations representing a **system of repeated measures** at `M` time points. The equations may contain 1) ordinal ($r \geq 2$ categories), continuous (normal, non-normal, and mixture distributions), count (regular and zero-inflated, Poisson and Negative Binomial) independent variables X; 2) continuous error terms E; 3) a discrete time variable $Time$; and 4) random effects U. The assumptions are that 1) there are at least 2 equations, 2) the independent variables, random effect terms, and error terms are uncorrelated, 3) each equation has an error term, 4) all error terms have a continuous non-mixture distribution or all have a continuous mixture distribution, 5) all random effects are continuous, and 6) growth is linear (with respect to time). The random effects may be a random intercept, a random slope for time, or a random slope for any of the X variables. Continuous variables are simulated using either Fleishman's third-order (`method = "Fleishman"`, doi: 10.1007/BF02293811) or Headrick's fifth-order (`method = "Polynomial"`, doi: 10.1016/S01679473(02)000725) power method transformation (PMT). Simulation occurs at the component-level for continuous mixture distributions. The target correlation matrix is specified in terms of correlations with components of continuous mixture variables. These components are transformed into the desired mixture variables using random multinomial variables based on the mixing probabilities. The X terms can be the same across equations (i.e., modeling sex or height) or may be time-varying covariates. The equations may contain different numbers of X terms (i.e., a covariate could be missing for a given equation).

The outcomes Y are generated using a hierarchical linear models (HLM) approach, which allows the data to be structured in at least two levels. **Level-1** is the repeated measure (time or condition) and other subject-level variables. **Level-1** is nested within **Level-2**, which describes the average of the outcome (the intercept) and growth (slope for time) as a function of group-level variables. The first level captures the within-subject variation, while the second level describes the between-subjects variability. Using a HLM provides a way to determine if: a) subjects differ at a specific time point with respect to the dependent variable, b) growth rates differ across conditions, or c) growth rates differ across subjects. Random effects describe deviation at the subject-level from the average (fixed) effect described by the slope coefficients (betas). See the **The Hierarchical Linear Models Approach for a System of Correlated Equations with Multiple Variable Types** vignette for a description of the HLM model. The user can specify subject-level X terms, and each subject-level X term is crossed with all group-level X terms. The equations may also contain interactions between X variables. Interactions specified in `int.var` between two group-level covariates are themselves considered group-level covariates and will be crossed with subject-level covariates. Interactions between two subject-level covariates are considered subject-level covariates and will be crossed with group-level covariates. Since `Time` is a subject-level variable, each group-level term is crossed with `Time` unless otherwise specified.

Random effects may be added for the intercept, time slope, or effects of any of the covariates. The type of random intercept and time slope (i.e., non-mixture or mixture) is specified in `rand.int` and `rand.tsl`. This type may vary by equation. The random effects for independent variables are specified in `rand.var` and may also contain a combination of non-mixture and mixture continuous distributions. If the parameter lists are of length `M + 1`, the random effects are the same variables across equations and the correlation for the effects `corr.u` is a matrix. If the parameter lists are of length `2 * M`, the random effects are different variables across equations and the correlation for the effects `corr.u` is a list.

The independent variables, interactions, `Time` effect, random effects, and error terms are summed together to produce the outcomes Y. The beta coefficients may be the same or differ across equations. The user specifies the betas for the independent variables in `betas`, for the interactions between two group-level or two subject-level covariates in `betas.int`, for the group-subject level interactions in `betas.subj`, and for the `Time` interactions in `betas.tint`. Setting a coefficient to 0 will eliminate that term. The user also provides the correlations 1) between E terms; 2) between X variables within each outcome Y_p, `p = 1, ..., M`, and between outcome pairs; and 3) between U variables within each outcome Y_p, `p = 1, ..., M`, and between outcome pairs. The order of the independent variables in `corr.x` must be 1st ordinal (same order as in `marginal`), 2nd continuous non-mixture (same order as in `skews`), 3rd components of continuous mixture (same order as in `mix_pis`), 4th regular Poisson, 5th zero-inflated Poisson (same order as in `lam`), 6th regular NB, and 7th zero-inflated NB (same order as in `size`). The order of the random effects in `corr.u` must be 1st random intercept, 2nd random time slope, 3rd continuous non-mixture random effects, and 4th components of continuous mixture random effects.

The variables are generated from multivariate normal variables with intermediate correlations calculated using `intercorr2`, which employs **correlation method 2**. See `SimCorrMix` for a description of the correlation method and the techniques used to generate each variable type. The order of the variables returned is 1st covariates X (as specified in `corr.x`), 2nd group-group or subject-subject interactions (ordered as in `int.var`), 3rd subject-group interactions (1st by subject-level variable as specified in `subj.var`, 2nd by covariate as specified in `corr.x`), and 4th time interactions (either as specified in `corr.x` with group-level covariates or in `tint.var`).

This function contains no parameter checks in order to decrease simulation time. That should be done first using `checkpar`. Summaries of the system can be obtained using `summary_sys`. The **Correlated Systems of Statistical Equations with Multiple Variable Types** demonstrates examples.

Usage

```
corrsys2(n = 10000, M = NULL, Time = NULL, method = c("Fleishman",
  "Polynomial"), error_type = c("non_mix", "mix"), means = list(),
  vars = list(), skews = list(), skurts = list(), fifths = list(),
  sixths = list(), Six = list(), mix_pis = list(), mix_mus = list(),
  mix_sigmas = list(), mix_skews = list(), mix_skurts = list(),
  mix_fifths = list(), mix_sixths = list(), mix_Six = list(),
  marginal = list(), support = list(), lam = list(), p_zip = list(),
  pois_eps = list(), size = list(), prob = list(), mu = list(),
  p_zinb = list(), nb_eps = list(), corr.x = list(), corr.e = NULL,
  same.var = NULL, subj.var = NULL, int.var = NULL, tint.var = NULL,
  betas.0 = NULL, betas = list(), betas.subj = list(),
  betas.int = list(), betas.t = NULL, betas.tint = list(),
  rand.int = c("none", "non_mix", "mix"), rand.tsl = c("none", "non_mix",
  "mix"), rand.var = NULL, corr.u = list(), seed = 1234,
  use.nearPD = TRUE, eigmin = 0, adjgrad = FALSE, B1 = NULL,
  tau = 0.5, tol = 0.1, steps = 100, msteps = 10, errorloop = FALSE,
  epsilon = 0.001, maxit = 1000, quiet = FALSE)
```

Arguments

`n`	the sample size (i.e. the length of each simulated variable; default = 10000)
`M`	the number of dependent variables Y (outcomes); equivalently, the number of equations in the system
`Time`	a vector of values to use for time; each subject receives the same time value; if `NULL`, `Time = 1:M`
`method`	the PMT method used to generate all continuous variables, including independent variables (covariates), error terms, and random effects; "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation
`error_type`	"non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions
`means`	if no random effects, a list of length `M` where `means[[p]]` contains a vector of means for the continuous independent variables in equation p with non-mixture (X_{cont}) or mixture (X_{mix}) distributions and for the error terms (E); order in vector is X_{cont}, X_{mix}, E if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `means[M + 1]` or `means[(M + 1):(2 * M)]`

are vectors of means for all random effects with continuous non-mixture or mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int != "none"`), 2nd random time slope U_1 (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions U_{cont}, 4th other random slopes with mixture distributions U_{mix}

`vars` a list of same length and order as `means` containing vectors of variances for the continuous variables, error terms, and any random effects

`skews` if no random effects, a list of length `M` where `skews[[p]]` contains a vector of skew values for the continuous independent variables in equation p with non-mixture (X_{cont}) distributions and for E if `error_type = "non_mix"`; order in vector is X_{cont}, E

if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `skews[M + 1]` or `skews[(M + 1):(2 * M)]` are vectors of skew values for all random effects with continuous non-mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int = "non_mix"`), 2nd random time slope U_1 (if `rand.tsl = "non_mix"`), 3rd other random slopes with non-mixture distributions U_{cont}

`skurts` a list of same length and order as `skews` containing vectors of standardized kurtoses (kurtosis - 3) for the continuous variables, error terms, and any random effects with non-mixture distributions

`fifths` a list of same length and order as `skews` containing vectors of standardized fifth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for `method = "Fleishman"`

`sixths` a list of same length and order as `skews` containing vectors of standardized sixth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for `method = "Fleishman"`

`Six` a list of length `M`, `M + 1`, or `2 * M`, where `Six[1:M]` are for X_{cont}, E (if `error_type = "non_mix"`) and `Six[M + 1]` or `Six[(M + 1):(2 * M)]` are for non-mixture U; if `error_type = "mix"` and there are only random effects (i.e., `length(corr.x) = 0`), use `Six[1:M] = rep(list(NULL), M)` so that `Six[M + 1]` or `Six[(M + 1):(2 * M)]` describes the non-mixture U;

`Six[[p]][[j]]` is a vector of sixth cumulant correction values to aid in finding a valid PDF for $X_{cont(pj)}$, the j-th continuous non-mixture covariate for outcome Y_p; the last vector in `Six[[p]]` is for E_p (if `error_type = "non_mix"`); use `Six[[p]][[j]] = NULL` if no correction desired for $X_{cont(pj)}$; use `Six[[p]] = NULL` if no correction desired for any continuous non-mixture covariate or error term in equation p

`Six[[M + p]][[j]]` is a vector of sixth cumulant correction values to aid in finding a valid PDF for $U_{(pj)}$, the j-th non-mixture random effect for outcome Y_p; use `Six[[M + p]][[j]] = NULL` if no correction desired for $U_{(pj)}$; use `Six[[M + p]] = NULL` if no correction desired for any continuous non-mixture random effect in equation p

keep `Six = list()` if no corrections desired for all equations or if `method = "Fleishman"`

`mix_pis` list of length `M`, `M + 1` or `2 * M`, where `mix_pis[1:M]` are for X_{cont}, E (if `error_type = "mix"`) and `mix_pis[M + 1]` or `mix_pis[(M + 1):(2 * M)]` are for mixture U; use `mix_pis[[p]] = NULL` if equation p has no continuous mixture terms if `error_type = "non_mix"` and there are only random effects (i.e., `length(corr.x) = 0`), use `mix_pis[1:M] = NULL` so that `mix_pis[M + 1]` or `mix_pis[(M + 1):(2 * M)]` describes the mixture U;

mix_pis[[p]][[j]] is a vector of mixing probabilities of the component distributions for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last vector in mix_pis[[p]] is for E_p (if error_type = "mix"); components should be ordered as in corr.x

mix_pis[[M + p]][[j]] is a vector of mixing probabilities of the component distributions for $U_{(pj)}$, the j-th random effect with a mixture distribution for outcome Y_p; order is 1st random intercept (if rand.int = "mix"), 2nd random time slope (if rand.tsl = "mix"), 3rd other random slopes with mixture distributions; components should be ordered as in corr.u

mix_mus
: list of same length and order as mix_pis;

  mix_mus[[p]][[j]] is a vector of means of the component distributions for $X_{mix(pj)}$, the last vector in mix_mus[[p]] is for E_p (if error_type = "mix")

  mix_mus[[p]][[j]] is a vector of means of the component distributions for $U_{mix(pj)}$

mix_sigmas
: list of same length and order as mix_pis;

  mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $X_{mix(pj)}$, the last vector in mix_sigmas[[p]] is for E_p (if error_type = "mix")

  mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $U_{mix(pj)}$

mix_skews
: list of same length and order as mix_pis;

  mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $X_{mix(pj)}$, the last vector in mix_skews[[p]] is for E_p (if error_type = "mix")

  mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $U_{mix(pj)}$

mix_skurts
: list of same length and order as mix_pis;

  mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $X_{mix(pj)}$, the last vector in mix_skurts[[p]] is for E_p (if error_type = "mix")

  mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $U_{mix(pj)}$

mix_fifths
: list of same length and order as mix_pis; not necessary for method = "Fleishman";

  mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_fifths[[p]] is for E_p (if error_type = "mix")

  mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $U_{mix(pj)}$

mix_sixths
: list of same length and order as mix_pis; not necessary for method = "Fleishman";

  mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_sixths[[p]] is for E_p (if error_type = "mix")

  mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $U_{mix(pj)}$

mix_Six
: a list of same length and order as mix_pis; keep mix_Six = list() if no corrections desired for all equations or if method = "Fleishman"

  p-th component of mix_Six[1:M] is a list of length equal to the total number of component distributions for the $X_{mix(p)}$ and E_p (if error_type = "mix"); mix_Six[[p]][[j]] is a vector of sixth cumulant corrections for the j-th component distribution (i.e., if there are 2 continuous mixture independent variables for Y_p, where $X_{mix(p1)}$ has 2 components and $X_{mix(p2)}$ has 3 components, then `length(mix_Six[[p]]) = 5` and `mix_Six[[p]][[3]]` would correspond to the 1st component of $X_{mix(p2)}$); use `mix_Six[[p]][[j]] = NULL` if no correction desired for that component; use `mix_Six[[p]] = NULL` if no correction desired for any component of $X_{mix(p)}$ and E_p

q-th component of `mix_Six[M + 1]` or `mix_Six[(M + 1):(2 * M)]` is a list of length equal to the total number of component distributions for the $U_{mix(q)}$; `mix_Six[[q]][[j]]` is a vector of sixth cumulant corrections for the j-th component distribution; use `mix_Six[[q]][[j]] = NULL` if no correction desired for that component; use `mix_Six[[q]] = NULL` if no correction desired for any component of $U_{mix(q)}$

`marginal` a list of length `M`, with the p-th component a list of cumulative probabilities for the ordinal variables associated with outcome Y_p (use `marginal[[p]] = NULL` if outcome Y_p has no ordinal variables); `marginal[[p]][[j]]` is a vector of the cumulative probabilities defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1); for binary variables, the probability is the probability of the 1st category, which has the smaller support value; `length(marginal[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`support` a list of length `M`, with the p-th component a list of support values for the ordinal variables associated with outcome Y_p; use `support[[p]] = NULL` if outcome Y_p has no ordinal variables; `support[[p]][[j]]` is a vector of the support values defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if not provided, the default for r categories is 1, ..., r

`lam` list of length `M`, p-th component a vector of lambda (means > 0) values for Poisson variables for outcome Y_p (see `stats::dpois`); order is 1st regular Poisson and 2nd zero-inflated Poisson; use `lam[[p]] = NULL` if outcome Y_p has no Poisson variables; `length(lam[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

`p_zip` a list of vectors of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `VGAM::dzipois`); if `p_zip` = 0, Y_{pois} has a regular Poisson distribution; if `p_zip` is in (0, 1), Y_{pois} has a zero-inflated Poisson distribution; if `p_zip` is in `(-(exp(lam) - 1)^(-1), 0)`, Y_{pois} has a zero-deflated Poisson distribution and `p_zip` is not a probability; if `p_zip = -(exp(lam) - 1)^(-1)`, Y_{pois} has a positive-Poisson distribution (see `VGAM::dpospois`); order is 1st regular Poisson and 2nd zero-inflated Poisson; if a single number, all Poisson variables given this value; if a vector of length `M`, all Poisson variables in equation p given `p_zip[p]`; otherwise, missing values are set to 0 and ordered 1st

`pois_eps` list of length `M`, p-th component a vector of length `lam[[p]]` containing cumulative probability truncation values used to calculate intermediate correlations involving Poisson variables; order is 1st regular Poisson and 2nd zero-inflated Poisson; if a single number, all Poisson variables given this value; if a vector of length `M`, all Poisson variables in equation p given `pois_eps[p]`; otherwise, missing values are set to 0.0001 and ordered 1st

`size` list of length `M`, p-th component a vector of size parameters for the Negative Binomial variables for outcome Y_p (see `stats::dnbinom`); order is 1st regular NB and 2nd zero-inflated NB; use `size[[p]] = NULL` if outcome Y_p has no Negative Binomial variables; `length(size[[p]])` can differ across outcomes; the order should be the same as in `corr.x`

prob
: list of length M, p-th component a vector of success probabilities for the Negative Binomial variables for outcome Y_p (see stats::dnbinom); order is 1st regular NB and 2nd zero-inflated NB; use prob[[p]] = NULL if outcome Y_p has no Negative Binomial variables; length(prob[[p]]) can differ across outcomes; the order should be the same as in corr.x

mu
: list of length M, p-th component a vector of mean values for the Negative Binomial variables for outcome Y_p (see stats::dnbinom); order is 1st regular NB and 2nd zero-inflated NB; use mu[[p]] = NULL if outcome Y_p has no Negative Binomial variables; length(mu[[p]]) can differ across outcomes; the order should be the same as in corr.x; for zero-inflated NB variables, this refers to the mean of the NB distribution (see VGAM::dzinegbin) (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture)

p_zinb
: a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see VGAM::dzinegbin); if p_zinb = 0, Y_{nb} has a regular NB distribution; if p_zinb is in (-prob^size/(1 - prob^size), 0), Y_{nb} has a zero-deflated NB distribution and p_zinb is not a probability; if p_zinb = -prob^size/(1 - prob^size), Y_{nb} has a positive-NB distribution (see VGAM::dposnegbin); order is 1st regular NB and 2nd zero-inflated NB; if a single number, all NB variables given this value; if a vector of length M, all NB variables in equation p given p_zinb[p]; otherwise, missing values are set to 0 and ordered 1st

nb_eps
: list of length M, p-th component a vector of length size[[p]] containing cumulative probability truncation values used to calculate intermediate correlations involving Negative Binomial variables; order is 1st regular NB and 2nd zero-inflated NB; if a single number, all NB variables given this value; if a vector of length M, all NB variables in equation p given nb_eps[p]; otherwise, missing values are set to 0.0001 and ordered 1st

corr.x
: list of length M, each component a list of length M; corr.x[[p]][[q]] is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); order: 1st ordinal (same order as in marginal), 2nd continuous non-mixture (same order as in skews), 3rd components of continuous mixture (same order as in mix_pis), 4th regular Poisson, 5th zero-inflated Poisson (same order as in lam), 6th regular NB, and 7th zero-inflated NB (same order as in size); if p = q, corr.x[[p]][[q]] is a correlation matrix with nrow(corr.x[[p]][[q]]) = # $X_{(pj)}$ for outcome Y_p; if p != q, corr.x[[p]][[q]] is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that nrow(corr.x[[p]][[q]]) = # $X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that ncol(corr.x[[p]][[q]]) = # $X_{(qj)}$ for outcome Y_q; use corr.x[[p]][[q]] = NULL if equation q has no $X_{(qj)}$; use corr.x[[p]] = NULL if equation p has no $X_{(pj)}$

corr.e
: correlation matrix for continuous non-mixture or components of mixture error terms

same.var
: either a vector or a matrix; if a vector, same.var includes column numbers of corr.x[[1]][[1]] corresponding to independent variables that should be identical across equations; these terms must have the same indices for all p = 1, ..., M; i.e., if the 1st ordinal variable represents sex which should be the same for each equation, then same.var[1] = 1 since ordinal variables are 1st in corr.x[[1]][[1]] and sex is the 1st ordinal variable, and the 1st term for all other outcomes must also be sex; if a matrix, columns 1 and 2 are outcome p and column index in corr.x[[p]][[p]] for 1st instance of variable, columns 3 and 4 are outcome q and column index in corr.x[[q]][[q]] for subsequent instances of variable;

i.e., if 1st term for all outcomes is sex and `M = 3`, then `same.var = matrix(c(1, 1, 2, 1, 1, 1, 3, 1), 2, 4, byrow = TRUE)`; the independent variable index corresponds to ordinal, continuous non-mixture, component of continuous mixture, Poisson, or NB variable

`subj.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is independent variable index corresponding to covariate which is a a subject-level term (not including time), including time-varying covariates; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable; assumes all other variables are group-level terms; these subject-level terms are used to form interactions with the group level terms

`int.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd and 3rd columns are indices corresponding to two group-level or two subject-level independent variables to form interactions between; this includes all interactions that are not accounted for by a subject-group level interaction (as indicated by `subj.var`) or by a time-covariate interaction (as indicated by `tint.var`); ex: 1, 2, 3 indicates that for outcome 1, the 2nd and 3rd independent variables form an interaction term; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable

`tint.var` matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is index of independent variable to form interaction with time; if `tint.var = NULL` or no $X_{(pj)}$ are indicated for outcome Y_p, all group-level variables (variables not indicated as subject-level variables in `subj.var`) will be crossed with time, else includes only terms indicated by 2nd column (i.e., in order to include subject-level variables); ex: 1, 1 indicates that for outcome 1, the 1st independent variable has an interaction with time; the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable

`betas.0` vector of length `M` containing intercepts, if `NULL` all set equal to 0; if length 1, all intercepts set to `betas.0`

`betas` list of length `M`, p-th component a vector of coefficients for outcome Y_p, including group and subject-level terms; order is order of variables in `corr.x[[p]][[p]]`; if `betas = list()`, all set to 0 so that all Y only have intercept and/or interaction terms plus error terms; if all outcomes have the same betas, use list of length 1; if Y_p only has intercept and/or interaction terms, set `betas[[p]] = NULL`; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.subj` list of length `M`, p-th component a vector of coefficients for interaction terms between group-level terms and subject-level terms given in `subj.var`; order is 1st by subject-level covariate as given in `subj.var` and 2nd by group-level covariate as given in `corr.x` or an interaction between group-level terms; if all outcomes have the same betas, use list of length 1; if Y_p only has group-level terms, set `betas.subj[[p]] = NULL`; since subject-subject interactions are treated as subject-level variables, these will also be crossed with all group-level variables and require coefficients; if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.int` list of length `M`, p-th component a vector of coefficients for interaction terms indicated in `int.var`; order is the same order as given in `int.var`; if all outcomes have the same betas, use list of length 1; if Y_p has none, set `betas.int[[p]] = NULL`;

if there are continuous mixture variables, beta is for mixture variable (not for components)

`betas.t`
: vector of length `M` of coefficients for time terms, if `NULL` all set equal to 1; if length 1, all intercepts set to `betas.t`

`betas.tint`
: list of length `M`, p-th component a vector of coefficients for all interactions with time; this includes interactions with group-level covariates or terms indicated in `tint.var`; order is the same order as given in `corr.x` or `tint.var`; if all outcomes have the same betas, use list of length 1; if Y_p has none, set `betas.tint[[p]] = NULL`; since group-group interactions are treated as group-level variables, these will also be crossed with time (unless otherwise specified for that outcome in `tint.var`) and require coefficients; if there are continuous mixture variables, beta is for mixture variable (not for components)

`rand.int`
: "none" (default) if no random intercept term for all outcomes, "non_mix" if all random intercepts have a continuous non-mixture distribution, "mix" if all random intercepts have a continuous mixture distribution; also can be a vector of length `M` containing a combination (i.e., `c("non_mix", "mix", "none")` if the 1st has a non-mixture distribution, the 2nd has a mixture distribution, and 3rd outcome has no random intercept)

`rand.tsl`
: "none" (default) if no random slope for time for all outcomes, "non_mix" if all random time slopes have a continuous non-mixture distribution, "mix" if all random time slopes have a continuous mixture distribution; also can be a vector of length `M` as in `rand.int`

`rand.var`
: matrix where 1st column is outcome index (`p = 1, ..., M`), 2nd column is independent variable index corresponding to covariate to assign random effect to (not including the random intercept or time slope if present); the independent variable index corresponds to ordinal, continuous non-mixture, continuous mixture (not mixture component), Poisson, or NB variable; order is 1st continuous non-mixture and 2nd continuous mixture random effects; note that the order of the rows corresponds to the order of the random effects in `corr.u` not the order of the independent variable so that a continuous mixture covariate with a non-mixture random effect would be ordered before a continuous non-mixture covariate with a mixture random effect (the 2nd column of `rand.var` indicates the specific covariate)

`corr.u`
: if the random effects are the same variables across equations, a matrix of correlations for U; if the random effects are different variables across equations, a list of length `M`, each component a list of length `M`; `corr.u[[p]][[q]]` is matrix of correlations for random effects in equations p ($U_{(pj)}$ for outcome Y_p) and q ($U_{(qj)}$ for outcome Y_q); if p = q, `corr.u[[p]][[q]]` is a correlation matrix with `nrow(corr.u[[p]][[q]]) = #` $U_{(pj)}$ for outcome Y_p; if p != q, `corr.u[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to $U_{(pj)}$ for Y_p so that `nrow(corr.u[[p]][[q]]) = #` $U_{(pj)}$ for outcome Y_p and columns correspond to $U_{(qj)}$ for Y_q so that `ncol(corr.u[[p]][[q]]) = #` $U_{(qj)}$ for outcome Y_q; the number of random effects for Y_p is taken from `nrow(corr.u[[p]][[1]])` so that if there should be random effects, there must be entries for `corr.u`; use `corr.u[[p]][[q]] = NULL` if equation q has no $U_{(qj)}$; use `corr.u[[p]] = NULL` if equation p has no $U_{(pj)}$;
: correlations are specified in terms of components of mixture variables (if present); order is 1st random intercept (if `rand.int != "none"`), 2nd random time slope (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions, 4th other random slopes with mixture distributions

`seed`
: the seed value for random number generation (default = 1234)

use.nearPD TRUE to convert the overall intermediate correlation matrix formed by the X (for all outcomes and independent variables), E, or the random effects to the nearest positive definite matrix with Matrix::nearPD if necessary; if FALSE and adjgrad = FALSE the negative eigenvalues are replaced with eigmin if necessary

eigmin minimum replacement eigenvalue if overall intermediate correlation matrix is not positive-definite (default = 0)

adjgrad TRUE to use adj_grad to convert overall intermediate correlation matrix to a positive-definite matrix and next 5 inputs can be used

B1 the initial matrix for algorithm; if NULL, uses a scaled initial matrix with diagonal elements sqrt(nrow(Sigma))/2

tau parameter used to calculate theta (default = 0.5)

tol maximum error for Frobenius norm distance between new matrix and original matrix (default = 0.1)

steps maximum number of steps for k (default = 100)

msteps maximum number of steps for m (default = 10)

errorloop if TRUE, uses corr_error to attempt to correct the correlation of the independent variables within and across outcomes to be within epsilon of the target correlations corr.x until the number of iterations reaches maxit (default = FALSE)

epsilon the maximum acceptable error between the final and target correlation matrices (default = 0.001) in the calculation of ordinal intermediate correlations with ord_norm or in the error loop

maxit the maximum number of iterations to use (default = 1000) in the calculation of ordinal intermediate correlations with ord_norm or in the error loop

quiet if FALSE prints messages, if TRUE suppresses messages

Value

A list with the following components:

Y matrix with n rows and M columns of outcomes

X list of length M containing $X_{ord(pj)}, X_{cont(pj)}, X_{comp(pj)}, X_{pois(pj)}, X_{nb(pj)}$

X_all list of length M containing $X_{ord(pj)}, X_{cont(pj)}, X_{mix(pj)}, X_{pois(pj)}, X_{nb(pj)}$, X interactions as indicated by int.var, subject-group level term interactions as indicated by subj.var, $Time_p$, and $Time$ interactions as indicated by tint.var; order is 1st covariates X (as specified in corr.x), 2nd group-group or subject-subject interactions (ordered as in int.var), 3rd subject-group interactions (1st by subject-level variable as specified in subj.var, 2nd by covariate as specified in corr.x), and 4th time interactions (either as specified in corr.x with group-level covariates or in tint.var)

E matrix with n rows containing continuous non-mixture or components of continuous mixture error terms

E_mix matrix with n rows containing continuous mixture error terms

Sigma_X0 matrix of intermediate correlations calculated by intercorr2

Sigma_X matrix of intermediate correlations after nearPD or adj_grad function has been used; applied to generate the normal variables transformed to get the desired distributions

Error_Time the time in minutes required to use the error loop

Time the total simulation time in minutes `niter` a matrix of the number of iterations used in the error loop

If **continuous variables** are produced: `constants` a list of maximum length `2 * M`, the 1st `M` components are data.frames of the constants for the $X_{cont(pj)}$, $X_comp(pj)$ and E_p, the 2nd `M` components are for random effects (if present),

`SixCorr` a list of maximum length `2 * M`, the 1st `M` components are lists of sixth cumulant correction values used to obtain valid *pdf*'s for the $X_{cont(pj)}$, $X_comp(pj)$, and E_p, the 2nd `M` components are for random effects (if present),

`valid.pdf` a list of maximum length `2 * M` of vectors where the i-th element is "TRUE" if the constants for the i-th continuous variable generate a valid pdf, else "FALSE"; the 1st `M` components are for the $X_{cont(pj)}$, $X_comp(pj)$, and E_p, the 2nd `M` components are for random effects (if present)

If **random effects** are produced: `U` a list of length `M` containing matrices of continuous non-mixture and components of mixture random effects,

`U_all` a list of length `M` containing matrices of continuous non-mixture and mixture random effects,

`V` a list of length `M` containing matrices of design matrices for random effects,

`rmeans2` and `rvars2` the means and variances of the non-mixture and components reordered in accordance with the random intercept and time slope types (input for `summary_sys`)

Reasons for Function Errors

1) The most likely cause for function errors is that the parameter inputs are mispecified. Using `checkpar` prior to simulation can help decrease these errors.

2) Another reason for error is that no solutions to `fleish` or `poly` converged when using `find_constants`. If this happens, the simulation will stop. It may help to first use `find_constants` for each continuous variable to determine if a sixth cumulant correction value is needed. If the standardized cumulants are obtained from `calc_theory`, the user may need to use rounded values as inputs (i.e. `skews = round(skews, 8)`). For example, in order to ensure that skew is exactly 0 for symmetric distributions.

3) The kurtosis for a continuous variable may be outside the region of possible values. There is an associated lower kurtosis boundary for associated with a given skew (for Fleishman's method) or skew and fifth and sixth cumulants (for Headrick's method). Use `calc_lower_skurt` to determine the boundary for a given set of cumulants.

References

Barbiero A & Ferrari PA (2015). Simulation of correlated Poisson variables. Applied Stochastic Models in Business and Industry, 31:669-80. doi: 10.1002/asmb.2072.

Barbiero A & Ferrari PA (2015). GenOrd: Simulation of Discrete Random Variables with Given Correlation Matrix and Marginal Distributions. R package version 1.4.0.
https://CRAN.R-project.org/package=GenOrd

Davenport JW, Bezder JC, & Hathaway RJ (1988). Parameter Estimation for Finite Mixture Distributions. Computers & Mathematics with Applications, 15(10):819-28.

Demirtas H (2006). A method for multivariate ordinal data generation given marginal distributions and correlations. Journal of Statistical Computation and Simulation, 76(11):1017-1025.
doi: 10.1080/10629360600569246.

Demirtas H (2014). Joint Generation of Binary and Nonnormal Continuous Data. Biometrics & Biostatistics, S12.

Demirtas H, Hedeker D, & Mermelstein RJ (2012). Simulation of massive public health data by power polynomials. Statistics in Medicine, 31(27):3337-3346. doi: 10.1002/sim.5362.

Everitt BS (1996). An Introduction to Finite Mixture Distributions. Statistical Methods in Medical Research, 5(2):107-127. doi: 10.1177/096228029600500202.

Ferrari PA & Barbiero A (2012). Simulating ordinal data. Multivariate Behavioral Research, 47(4): 566-589. doi: 10.1080/00273171.2012.692630.

Fialkowski AC (2017). SimMultiCorrData: Simulation of Correlated Data with Multiple Variable Types. R package version 0.2.1. `https://CRAN.R-project.org/package=SimMultiCorrData`.

Fialkowski AC (2018). SimCorrMix: Simulation of Correlated Data of Multiple Variable Types including Continuous and Count Mixture Distributions. R package version 0.1.0. `https://github.com/AFialkowski/SimCorrMix`

Fleishman AI (1978). A Method for Simulating Non-normal Distributions. Psychometrika, 43:521-532. doi: 10.1007/BF02293811.

Headrick TC (2002). Fast Fifth-order Polynomial Transforms for Generating Univariate and Multivariate Non-normal Distributions. Computational Statistics & Data Analysis, 40(4):685-711. doi: 10.1016/S01679473(02)000725. (ScienceDirect)

Headrick TC (2004). On Polynomial Transformations for Simulating Multivariate Nonnormal Distributions. Journal of Modern Applied Statistical Methods, 3(1):65-71. doi: 10.22237/jmasm/1083370080.

Headrick TC, Kowalchuk RK (2007). The Power Method Transformation: Its Probability Density Function, Distribution Function, and Its Further Use for Fitting Data. Journal of Statistical Computation and Simulation, 77:229-249. doi: 10.1080/10629360600605065.

Headrick TC, Sawilowsky SS (1999). Simulating Correlated Non-normal Distributions: Extending the Fleishman Power Method. Psychometrika, 64:25-35. doi: 10.1007/BF02294317.

Headrick TC, Sheng Y, & Hodis FA (2007). Numerical Computing and Graphics for the Power Method Transformation Using Mathematica. Journal of Statistical Software, 19(3):1 - 17. doi: 10.18637/jss.v019.i03.

Higham N (2002). Computing the nearest correlation matrix - a problem from finance; IMA Journal of Numerical Analysis 22:329-343.

Ismail N & Zamani H (2013). Estimation of Claim Count Data Using Negative Binomial, Generalized Poisson, Zero-Inflated Negative Binomial and Zero-Inflated Generalized Poisson Regression Models. Casualty Actuarial Society E-Forum 41(20):1-28.

Kincaid C (2005). Guidelines for Selecting the Covariance Structure in Mixed Model Analysis. Computational Statistics and Data Analysis, 198(30):1-8.

Lambert D (1992). Zero-Inflated Poisson Regression, with an Application to Defects in Manufacturing. Technometrics 34(1):1-14.

Lininger M, Spybrook J, & Cheatham CC (2015). Hierarchical Linear Model: Thinking Outside the Traditional Repeated-Measures Analysis-of-Variance Box. Journal of Athletic Training, 50(4):438-441. doi: 10.4085/1062605049.5.09.

McCulloch CE, Searle SR, Neuhaus JM (2008). *Generalized, Linear, and Mixed Models* (2nd ed.). Wiley Series in Probability and Statistics. Hoboken, New Jersey: John Wiley & Sons, Inc.

Olsson U, Drasgow F, & Dorans NJ (1982). The Polyserial Correlation Coefficient. Psychometrika, 47(3):337-47. doi: 10.1007/BF02294164.

Pearson RK (2011). Exploring Data in Engineering, the Sciences, and Medicine. In. New York: Oxford University Press.

Schork NJ, Allison DB, & Thiel B (1996). Mixture Distributions in Human Genetics Research. Statistical Methods in Medical Research, 5:155-178. doi: 10.1177/096228029600500204.

Vale CD & Maurelli VA (1983). Simulating Multivariate Nonnormal Distributions. Psychometrika, 48:465-471. doi: 10.1007/BF02293687.

Van Der Leeden R (1998). Multilevel Analysis of Repeated Measures Data. Quality & Quantity, 32(1):15-29.

Yee TW (2017). VGAM: Vector Generalized Linear and Additive Models. https://CRAN.R-project.org/package=VGAM.

Zhang X, Mallick H, & Yi N (2016). Zero-Inflated Negative Binomial Regression for Differential Abundance Testing in Microbiome Studies. Journal of Bioinformatics and Genomics 2(2):1-9. doi: 10.18454/jbg.2016.2.2.1.

See Also

find_constants, intercorr2, checkpar, summary_sys

Examples

```
M <- 3
B <- calc_theory("Beta", c(4, 1.5))
skews <- lapply(seq_len(M), function(x) B[3])
skurts <- lapply(seq_len(M), function(x) B[4])
fifths <- lapply(seq_len(M), function(x) B[5])
sixths <- lapply(seq_len(M), function(x) B[6])
Six <- lapply(seq_len(M), function(x) list(0.03))
corr.e <- matrix(c(1, 0.4, 0.4^2, 0.4, 1, 0.4, 0.4^2, 0.4, 1), M, M,
  byrow = TRUE)
means <- lapply(seq_len(M), function(x) B[1])
vars <- lapply(seq_len(M), function(x) B[2]^2)
marginal <- list(0.3, 0.4, 0.5)
support <- lapply(seq_len(M), function(x) list(0:1))
corr.x <- list(list(matrix(1, 1, 1), matrix(0.4, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(1, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(0.4, 1, 1), matrix(1, 1, 1)))
betas <- list(0.5)
betas.t <- 1
betas.tint <- list(0.25)
Sys2 <- corrsys2(10000, M, Time = 1:M, "Polynomial", "non_mix", means, vars,
  skews, skurts, fifths, sixths, Six, marginal = marginal, support = support,
  corr.x = corr.x, corr.e = corr.e, betas = betas, betas.t = betas.t,
  betas.tint = betas.tint, quiet = TRUE)

## Not run:
seed <- 276
n <- 10000
M <- 3
Time <- 1:M

# Error terms have a beta(4, 1.5) distribution with an AR(1, p = 0.4)
correlation structure
B <- calc_theory("Beta", c(4, 1.5))
skews <- lapply(seq_len(M), function(x) B[3])
skurts <- lapply(seq_len(M), function(x) B[4])
fifths <- lapply(seq_len(M), function(x) B[5])
sixths <- lapply(seq_len(M), function(x) B[6])
Six <- lapply(seq_len(M), function(x) list(0.03))
error_type <- "non_mix"
corr.e <- matrix(c(1, 0.4, 0.4^2, 0.4, 1, 0.4, 0.4^2, 0.4, 1), M, M,
  byrow = TRUE)
```

```
1 continuous mixture of Normal(-2, 1) and Normal(2, 1) for each Y
mix_pis <- lapply(seq_len(M), function(x) list(c(0.4, 0.6)))
mix_mus <- lapply(seq_len(M), function(x) list(c(-2, 2)))
mix_sigmas <- lapply(seq_len(M), function(x) list(c(1, 1)))
mix_skews <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_skurts <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_fifths <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_sixths <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_Six <- list()
Nstcum <- calc_mixmoments(mix_pis[[1]][[1]], mix_mus[[1]][[1]],
  mix_sigmas[[1]][[1]], mix_skews[[1]][[1]], mix_skurts[[1]][[1]],
  mix_fifths[[1]][[1]], mix_sixths[[1]][[1]])

means <- lapply(seq_len(M), function(x) c(Nstcum[1], B[1]))
vars <- lapply(seq_len(M), function(x) c(Nstcum[2]^2, B[2]^2))

# 1 binary variable for each Y
marginal <- lapply(seq_len(M), function(x) list(0.4))
support <- list(NULL, list(c(0, 1)), NULL)

# 1 Poisson variable for each Y
lam <- list(1, 5, 10)
# Y2 and Y3 have zero-inflated Poisson variables
p_zip <- list(NULL, 0.05, 0.1)

# 1 NB variable for each Y
size <- list(10, 15, 20)
prob <- list(0.3, 0.4, 0.5)
# either prob or mu is required (not both)
mu <- mapply(function(x, y) x * (1 - y)/y, size, prob, SIMPLIFY = FALSE)
# Y2 and Y3 have zero-inflated NB variables
p_zinb <- list(NULL, 0.05, 0.1)
pois_eps <- nb_eps <- list()

# The 2nd (the normal mixture) variable is the same across Y
same.var <- 2

# Create the correlation matrix in terms of the components of the normal
# mixture
K <- 5
corr.x <- list()
corr.x[[1]] <- list(matrix(0.1, K, K), matrix(0.2, K, K), matrix(0.3, K, K))
diag(corr.x[[1]][[1]]) <- 1
# set correlation between components to 0
corr.x[[1]][[1]][2:3, 2:3] <- diag(2)
# set correlations with the same variable equal across outcomes
corr.x[[1]][[2]][, same.var] <- corr.x[[1]][[3]][, same.var] <-
  corr.x[[1]][[1]][, same.var]
corr.x[[2]] <- list(t(corr.x[[1]][[2]]), matrix(0.35, K, K),
  matrix(0.4, K, K))
  diag(corr.x[[2]][[2]]) <- 1
  corr.x[[2]][[2]][2:3, 2:3] <- diag(2)
corr.x[[2]][[2]][, same.var] <- corr.x[[2]][[3]][, same.var] <-
  t(corr.x[[1]][[2]][same.var, ])
corr.x[[2]][[3]][same.var, ] <- corr.x[[1]][[3]][same.var, ]
corr.x[[2]][[2]][same.var, ] <- t(corr.x[[2]][[2]][, same.var])
corr.x[[3]] <- list(t(corr.x[[1]][[3]]), t(corr.x[[2]][[3]]),
```

```
  matrix(0.5, K, K))
diag(corr.x[[3]][[3]]) <- 1
corr.x[[3]][[3]][2:3, 2:3] <- diag(2)
corr.x[[3]][[3]][, same.var] <- t(corr.x[[1]][[3]][same.var, ])
corr.x[[3]][[3]][same.var, ] <- t(corr.x[[3]][[3]][, same.var])

# The 2nd and 3rd variables of each Y are subject-level variables
subj.var <- matrix(c(1, 2, 1, 3, 2, 2, 2, 3, 3, 2, 3, 3), 6, 2, byrow = TRUE)
int.var <- tint.var <- NULL
betas.0 <- 0
betas <- list(seq(0.5, 0.5 + (K - 2) * 0.25, 0.25))
betas.subj <- list(seq(0.5, 0.5 + (K - 2) * 0.1, 0.1))
betas.int <- list()
betas.t <- 1
betas.tint <- list(c(0.25, 0.5))

method <- "Polynomial"

# Check parameter inputs
checkpar(M, method, error_type, means, vars, skews, skurts, fifths, sixths,
  Six, mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts, mix_fifths,
  mix_sixths, mix_Six, marginal, support, lam, p_zip, pois_eps, size, prob,
  mu, p_zinb, nb_eps, corr.x, corr.yx = list(), corr.e, same.var, subj.var,
  int.var, tint.var, betas.0, betas, betas.subj, betas.int, betas.t,
  betas.tint)

# Simulated system using correlation method 2
N <- corrsys2(n, M, Time, method, error_type, means, vars, skews, skurts,
  fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts,
  mix_fifths, mix_sixths, mix_Six, marginal, support, lam, p_zip, pois_eps,
  size, prob, mu, p_zinb, nb_eps, corr.x, corr.e, same.var, subj.var,
  int.var, tint.var, betas.0, betas, betas.subj, betas.int, betas.t,
  betas.tint, seed = seed, use.nearPD = FALSE)

# Summarize the results
S <- summary_sys(N$Y, N$E, E_mix = NULL, N$X, N$X_all, M, method, means,
  vars, skews, skurts, fifths, sixths, mix_pis, mix_mus, mix_sigmas,
  mix_skews, mix_skurts, mix_fifths, mix_sixths, marginal, support, lam,
  p_zip, size, prob, mu, p_zinb, corr.x, corr.e)

## End(Not run)
```

nonnormsys — *Generate Correlated Systems of Equations Containing Normal, Non-Normal, and Mixture Continuous Variables*

Description

This function extends the techniques of Headrick and Beasley (2004, doi: 10.1081/SAC120028431) to create correlated systems of statistical equations containing continuous variables with normal, non-normal, or mixture distributions. The method allows the user to control the distributions for the stochastic disturbance (error) terms E and independent variables X. The user specifies the correlation structure between X terms within an outcome and across outcomes. For a given equation, the user also specifies the correlation between the outcome Y and X terms. These correlations are used to calculate the beta (slope) coefficients for the equations with `calc_betas`. If the system contains mixture variables and `corr.yx` is specified in terms of non-mixture and mixture variables, the `betas` will be calculated in terms of non-mixture and mixture independent variables. If `corr.yx` Finally, the user specifies the correlations across error terms. The assumptions are that 1) there are at least 2 equations and a total of at least 1 independent variable, 2) the independent variables are uncorrelated with the error terms, 3) each equation has an error term, and 4) all error terms have either a non-mixture or mixture distribution. The outcomes Y are calculated as the E terms added to the products of the beta coefficients and the X terms. There are no interactions between independent variables or distinction between subject and group-level terms (as in the hierarchical linear models approach of `corrsys` or `corrsys2`). However, the user does not have to provide the beta coefficients (except for the intercepts). Since the equations do not include random slopes (i.e. for the X terms), the effects of the independent variables are all considered "fixed." However, a random intercept or a "time" effect with a random slope could be added by specifying them as independent variables. There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with `checkpar`. Summaries of the simulation results can be found with `summary_sys`. The functions `calc_corr_y`, `calc_corr_yx`, and `calc_corr_ye` use equations from Headrick and Beasley (2004) to calculate the expected correlations for the outcomes, among a given outcome and covariates from the other outcomes, and for the error terms. The vignette **Theory and Equations for Correlated Systems of Continuous Variables** gives the equations, and the vignette **Correlated Systems of Statistical Equations with Non-Mixture and Mixture Continuous Variables** gives examples. There are also vignettes in `SimCorrMix` which provide more details on continuous non-mixture and mixture variables.

Usage

```
nonnormsys(n = 10000, M = NULL, method = c("Fleishman", "Polynomial"),
  error_type = c("non_mix", "mix"), means = list(), vars = list(),
  skews = list(), skurts = list(), fifths = list(), sixths = list(),
  Six = list(), mix_pis = list(), mix_mus = list(), mix_sigmas = list(),
  mix_skews = list(), mix_skurts = list(), mix_fifths = list(),
  mix_sixths = list(), mix_Six = list(), same.var = NULL,
  betas.0 = NULL, corr.x = list(), corr.yx = list(), corr.e = NULL,
  seed = 1234, use.nearPD = TRUE, eigmin = 0, adjgrad = FALSE,
  B1 = NULL, tau = 0.5, tol = 0.1, steps = 100, msteps = 10,
  errorloop = FALSE, epsilon = 0.001, maxit = 1000, quiet = FALSE)
```

Arguments

`n`	the sample size (i.e. the length of each simulated variable; default = 10000)
`M`	the number of dependent variables Y (outcomes); equivalently, the number of equations in the system
`method`	the PMT method used to generate all continuous variables, including independent variables (covariates) and error terms; "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation
`error_type`	"non_mix" if all error terms have continuous non-mixture distributions, "mix" if all error terms have continuous mixture distributions
`means`	a list of length `M` of vectors of means for the non-mixture (X_{cont}) and mixture (X_{mix}) independent variables and for the error terms (E); the order in each vector should be: X_{cont}, X_{mix}, E so that the order for X_{cont}, X_{mix} is the same as in `corr.x` (considering the components of mixture variables)

vars
: a list of length M of vectors of variances for X_{cont}, X_{mix}, E; same order and dimension as means

skews
: a list of length M of vectors of skew values for X_{cont} and E (if error_type = "non_mix"); same order as in corr.x and means

skurts
: a list of length M of vectors of standardized kurtoses (kurtosis - 3) for X_{cont} and E (if error_type = "non_mix"); same order and dimension as skews

fifths
: a list of length M of vectors of standardized fifth cumulants for X_{cont} and E (if error_type = "non_mix"); same order and dimension as skews; not necessary for method = "Fleishman"

sixths
: a list of length M of vectors of standardized sixth cumulants for X_{cont} and E (if error_type = "non_mix"); same order and dimension as skews; not necessary for method = "Fleishman"

Six
: a list of length M, where Six[[p]][[j]] is a vector of sixth cumulant correction values to aid in finding a valid PDF for $X_{cont(pj)}$, the j-th continuous non-mixture covariate for outcome Y_p; the last element of Six[[p]] is for E_p (if error_type = "non_mix"); use Six[[p]][[j]] = NULL if no correction desired for $X_{cont(pj)}$; use Six[[p]] = NULL if no correction desired for any non-mixture covariate or error term in equation p; keep Six = list() if no corrections desired for all covariates or if method = "Fleishman"

mix_pis
: a list of length M, where mix_pis[[p]][[j]] is a vector of mixing probabilities that sum to 1 for $X_{mix(pj)}$, the j-th continuous mixture covariate for outcome Y_p; the last element of mix_pis[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_pis[[p]] = NULL; components should be ordered as in corr.x

mix_mus
: a list of length M, where mix_mus[[p]][[j]] is a vector of means of the component distributions for $X_{mix(pj)}$; the last element of mix_mus[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_mus[[p]] = NULL

mix_sigmas
: a list of length M, where mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $X_{mix(pj)}$; the last element of mix_sigmas[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_sigmas[[p]] = NULL

mix_skews
: a list of length M, where mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $X_{mix(pj)}$; the last element of mix_skews[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_skews[[p]] = NULL

mix_skurts
: a list of length M, where mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $X_{mix(pj)}$; the last element of mix_skurts[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_skurts[[p]] = NULL

mix_fifths
: a list of length M, where mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $X_{mix(pj)}$; the last element of mix_fifths[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_fifths[[p]] = NULL; not necessary for method = "Fleishman"

mix_sixths
: a list of length M, where mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $X_{mix(pj)}$; the last element of mix_sixths[[p]] is for E_p (if error_type = "mix"); if Y_p has no mixture variables, use mix_sixths[[p]] = NULL; not necessary for method = "Fleishman"

mix_Six
: a list of length M, where mix_Six[[p]] is a list of length equal to the total number of component distributions for the $X_{mix(p)}$ and E_p (if error_type = "mix"); mix_Six[[p]][[j]] is a vector of sixth cumulant corrections for the j-th component distribution (i.e., if there are 2 continuous mixture independent variables for Y_p, where $X_{mix(p1)}$ has 2 components and $X_{mix(p2)}$ has 3 components,

then `length(mix_Six[[p]]) = 5` and `mix_Six[[p]][[3]]` would correspond to the 1st component of $X_{mix(p2)}$); use `mix_Six[[p]][[j]] = NULL` if no correction desired for that component; use `mix_Six[[p]] = NULL` if no correction desired for any component of $X_{mix(p)}$ and E_p; keep `mix_Six = list()` if no corrections desired for all covariates or if `method = "Fleishman"`

`same.var` either a vector or a matrix; if a vector, `same.var` includes column numbers of `corr.x[[1]][[1]]` corresponding to independent variables that should be identical across equations; these terms must have the same indices for all `p = 1, ..., M`; i.e., if the 1st variable represents height which should be the same for each equation, then `same.var[1] = 1` and the 1st term for all other outcomes must also be height; if a matrix, columns 1 and 2 are outcome p and column index in `corr.x[[p]][[p]]` for 1st instance of variable, columns 3 and 4 are outcome q and column index in `corr.x[[q]][[q]]` for subsequent instances of variable; i.e., if 1st term for all outcomes is height and `M = 3`, then `same.var = matrix(c(1, 1, 2, 1, 1, 1, 3, 1), 2, 4, byrow = TRUE)`; the independent variable index corresponds to continuous non-mixture and component of continuous mixture covariate

`betas.0` vector of length `M` containing intercepts, if `NULL` all set equal to 0; if length 1, all intercepts set to `betas.0`

`corr.x` list of length `M`, each component a list of length `M`; `corr.x[[p]][[q]]` is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); order: 1st continuous non-mixture (same order as in `skews`) and 2nd components of continuous mixture (same order as in `mix_pis`); if p = q, `corr.x[[p]][[q]]` is a correlation matrix with `nrow(corr.x[[p]][[q]])` = # of non-mixture + # of mixture components for outcome Y_p; if p != q, `corr.x[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that `nrow(corr.x[[p]][[q]])` = # of non-mixture + # of mixture components for outcome Y_p and columns correspond to covariates for Y_q so that `ncol(corr.x[[p]][[q]])` = # of non-mixture + # of mixture components for outcome Y_q; use `corr.x[[p]][[q]] = NULL` if equation q has no $X_{(qj)}$; use `corr.x[[p]] = NULL` if equation p has no $X_{(pj)}$

`corr.yx` a list of length `M`, where the p-th component is a 1 row matrix of correlations between Y_p and $X_{(pj)}$; if there are mixture variables and the `betas` are desired in terms of these (and not the components), then `corr.yx` should be specified in terms of correlations between outcomes and non-mixture or mixture variables, and the number of columns of the matrices of `corr.yx` should not match the dimensions of the matrices in `corr.x`; if the `betas` are desired in terms of the components, then `corr.yx` should be specified in terms of correlations between outcomes and non-mixture or components of mixture variables, and the number of columns of the matrices of `corr.yx` should match the dimensions of the matrices in `corr.x`; use `corr.yx[[p]] = NULL` if equation p has no $X_{(pj)}$

`corr.e` correlation matrix for continuous non-mixture or components of mixture error terms

`seed` the seed value for random number generation (default = 1234)

`use.nearPD` TRUE to convert the overall intermediate correlation matrix formed by the X (for all outcomes and independent variables) or E to the nearest positive definite matrix with `Matrix::nearPD` if necessary; if FALSE and `adjgrad = FALSE` the negative eigenvalues are replaced with `eigmin` if necessary

`eigmin` minimum replacement eigenvalue if overall intermediate correlation matrix is not positive-definite (default = 0)

adjgrad	TRUE to use adj_grad to convert overall intermediate correlation matrix to a positive-definite matrix and next 5 inputs can be used
B1	the initial matrix for algorithm; if NULL, uses a scaled initial matrix with diagonal elements sqrt(nrow(Sigma))/2
tau	parameter used to calculate theta (default = 0.5)
tol	maximum error for Frobenius norm distance between new matrix and original matrix (default = 0.1)
steps	maximum number of steps for k (default = 100)
msteps	maximum number of steps for m (default = 10)
errorloop	if TRUE, uses corr_error to attempt to correct the correlation of the independent variables within and across outcomes to be within epsilon of the target correlations corr.x until the number of iterations reaches maxit (default = FALSE)
epsilon	the maximum acceptable error between the final and target correlation matrices (default = 0.001) in the error loop
maxit	the maximum number of iterations to use (default = 1000) in the error loop
quiet	if FALSE prints messages, if TRUE suppresses messages

Value

A list with the following components:

Y matrix with n rows and M columns of outcomes

X list of length M containing $X_{cont(pj)}, X_{comp(pj)}$

X_all list of length M containing $X_{cont(pj)}, X_{mix(pj)}$

E matrix with n rows containing continuous non-mixture or components of continuous mixture error terms

E_mix matrix with n rows containing continuous mixture error terms

betas a matrix of the slope coefficients calculated with calc_betas, rows represent the outcomes

constants a list of length M with data.frames of the constants for the $X_{cont(pj)}$, $X_comp(pj)$ and E_p

SixCorr a list of length M of lists of sixth cumulant correction values used to obtain valid *pdf*'s for the $X_{cont(pj)}$, $X_comp(pj)$, and E_p

valid.pdf a list of length M of vectors where the i-th element is "TRUE" if the constants for the i-th continuous variable generate a valid pdf, else "FALSE"

Sigma_X0 matrix of intermediate correlations calculated by intercorr

Sigma_X matrix of intermediate correlations after nearPD or adj_grad function has been used; applied to generate the normal variables transformed to get the desired distributions

Error_Time the time in minutes required to use the error loop

Time the total simulation time in minutes

niter a matrix of the number of iterations used in the error loop

Generation of Continuous Non-Mixture and Mixture Variables

Mixture distributions describe random variables that are drawn from more than one component distribution. For a random variable X_{mix} from a finite continuous mixture distribution with k components, the probability density function (PDF) can be described by:

$$h_X(x) = \sum_{i=1}^{k} \pi_i f_{X_{comp_i}}(x), \sum_{i=1}^{k} \pi_i = 1.$$

The π_i are mixing parameters which determine the weight of each component distribution $f_{X_{comp_i}}(x)$ in the overall probability distribution. As long as each component has a valid PDF, the overall distribution $h_X()$ has a valid PDF. The main assumption is statistical independence between the process of randomly selecting the component distribution and the distributions themselves. Simulation is done at the component-level for mixture variables.

All continuous variables are simulated using either Fleishman's third-order (`method` = "Fleishman", doi: 10.1007/BF02293811) or Headrick's fifth-order (`method` = "Polynomial", doi: 10.1016/S0167-9473(02)000725) power method transformation (PMT). It works by matching standardized cumulants – the first four (mean, variance, skew, and standardized kurtosis) for Fleishman's method, or the first six (mean, variance, skew, standardized kurtosis, and standardized fifth and sixth cumulants) for Headrick's method. The transformation is expressed as follows:

$$Y = c_0 + c_1 * Z + c_2 * Z^2 + c_3 * Z^3 + c_4 * Z^4 + c_5 * Z^5, Z \sim N(0, 1),$$

where c_4 and c_5 both equal 0 for Fleishman's method. The real constants are calculated by `find_constants` for non-mixture and components of mixture variables. Continuous mixture variables are generated componentwise and then transformed to the desired mixture variables using random multinomial variables generated based on mixing probabilities. The correlation matrices are specified in terms of correlations with components of the mixture variables.

Choice of Fleishman's third-order or Headrick's fifth-order method

Using the fifth-order approximation allows additional control over the fifth and sixth moments of the generated distribution, improving accuracy. In addition, the range of feasible standardized kurtosis values, given skew and standardized fifth (γ_3) and sixth (γ_4) cumulants, is larger than with Fleishman's method (see `calc_lower_skurt`). For example, the Fleishman method can not be used to generate a non-normal distribution with a ratio of $\gamma_3^2/\gamma_4 > 9/14$ (see Headrick & Kowalchuk, 2007). This eliminates the Chi-squared family of distributions, which has a constant ratio of $\gamma_3^2/\gamma_4 = 2/3$. The fifth-order method also generates more distributions with valid PDF's. However, if the fifth and sixth cumulants are unknown or do not exist, the Fleishman approximation should be used.

Reasons for Function Errors

1) The most likely cause for function errors is that the parameter inputs are mispecified. Using `checkpar` prior to simulation can help decrease these errors.

2) No solutions to `fleish` or `poly` converged when using `find_constants`. If this happens, the simulation will stop. It may help to first use `find_constants` for each continuous variable to determine if a sixth cumulant correction value is needed. If the standardized cumulants are obtained from `calc_theory`, the user may need to use rounded values as inputs (i.e. `skews = round(skews, 8)`). For example, in order to ensure that skew is exactly 0 for symmetric distributions.

3) The kurtosis for a continuous variable may be outside the region of possible values. There is an associated lower kurtosis boundary for associated with a given skew (for Fleishman's method) or

skew and fifth and sixth cumulants (for Headrick's method). Use calc_lower_skurt to determine the boundary for a given set of cumulants.

4) No solutions to calc_betas converged when trying to find the beta coefficients. Try different correlation matrices.

References

Davenport JW, Bezder JC, & Hathaway RJ (1988). Parameter Estimation for Finite Mixture Distributions. Computers & Mathematics with Applications, 15(10):819-28.

Everitt BS (1996). An Introduction to Finite Mixture Distributions. Statistical Methods in Medical Research, 5(2):107-127. doi: 10.1177/096228029600500202.

Fialkowski AC (2017). SimMultiCorrData: Simulation of Correlated Data with Multiple Variable Types. R package version 0.2.1. https://CRAN.R-project.org/package=SimMultiCorrData.

Fialkowski AC (2018). SimCorrMix: Simulation of Correlated Data of Multiple Variable Types including Continuous and Count Mixture Distributions. R package version 0.1.0. https://github.com/AFialkowski/SimCorrMix

Fleishman AI (1978). A Method for Simulating Non-normal Distributions. Psychometrika, 43:521-532. doi: 10.1007/BF02293811.

Headrick TC (2002). Fast Fifth-order Polynomial Transforms for Generating Univariate and Multivariate Non-normal Distributions. Computational Statistics & Data Analysis, 40(4):685-711. doi: 10.1016/S01679473(02)000725. (ScienceDirect)

Headrick TC (2004). On Polynomial Transformations for Simulating Multivariate Nonnormal Distributions. Journal of Modern Applied Statistical Methods, 3(1):65-71. doi: 10.22237/jmasm/1083370080.

Headrick TC, Beasley TM (2004). A Method for Simulating Correlated Non-Normal Systems of Linear Statistical Equations. Communications in Statistics - Simulation and Computation, 33(1). doi: 10.1081/SAC120028431

Headrick TC, Kowalchuk RK (2007). The Power Method Transformation: Its Probability Density Function, Distribution Function, and Its Further Use for Fitting Data. Journal of Statistical Computation and Simulation, 77:229-249. doi: 10.1080/10629360600605065.

Headrick TC, Sawilowsky SS (1999). Simulating Correlated Non-normal Distributions: Extending the Fleishman Power Method. Psychometrika, 64:25-35. doi: 10.1007/BF02294317.

Headrick TC, Sheng Y, & Hodis FA (2007). Numerical Computing and Graphics for the Power Method Transformation Using Mathematica. Journal of Statistical Software, 19(3):1 - 17. doi: 10.18637/jss.v019.i03.

Higham N (2002). Computing the nearest correlation matrix - a problem from finance; IMA Journal of Numerical Analysis 22:329-343.

McCulloch CE, Searle SR, Neuhaus JM (2008). *Generalized, Linear, and Mixed Models* (2nd ed.). Wiley Series in Probability and Statistics. Hoboken, New Jersey: John Wiley & Sons, Inc.

Pearson RK (2011). Exploring Data in Engineering, the Sciences, and Medicine. In. New York: Oxford University Press.

Schork NJ, Allison DB, & Thiel B (1996). Mixture Distributions in Human Genetics Research. Statistical Methods in Medical Research, 5:155-178. doi: 10.1177/096228029600500204.

Vale CD & Maurelli VA (1983). Simulating Multivariate Nonnormal Distributions. Psychometrika, 48:465-471. doi: 10.1007/BF02293687.

See Also

calc_betas, calc_corr_y, calc_corr_yx, calc_corr_ye, checkpar, summary_sys

Examples

```
M <- 3
B <- calc_theory("Beta", c(4, 1.5))
skews <- lapply(seq_len(M), function(x) c(0, B[3]))
skurts <- lapply(seq_len(M), function(x) c(0, B[4]))
fifths <- lapply(seq_len(M), function(x) c(0, B[5]))
sixths <- lapply(seq_len(M), function(x) c(0, B[6]))
Six <- lapply(seq_len(M), function(x) list(NULL, 0.03))
corr.e <- matrix(c(1, 0.4, 0.4^2, 0.4, 1, 0.4, 0.4^2, 0.4, 1), M, M,
  byrow = TRUE)
means <- lapply(seq_len(M), function(x) c(0, B[1]))
vars <- lapply(seq_len(M), function(x) c(1, B[2]^2))
corr.x <- list(list(matrix(1, 1, 1), matrix(0.4, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(1, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(0.4, 1, 1), matrix(1, 1, 1)))
corr.yx <- list(matrix(0.4, 1), matrix(0.5, 1), matrix(0.6, 1))
Sys1 <- nonnormsys(10000, M, "Polynomial", "non_mix", means, vars,
  skews, skurts, fifths, sixths, Six, corr.x = corr.x, corr.yx = corr.yx,
  corr.e = corr.e)

## Not run:
# Example: system of three equations for 2 independent variables, where each
# error term has unit variance, from Headrick & Beasley (2002)
# Y_1 = beta_10 + beta_11 * X_11 + beta_12 * X_12 + sigma_1 * e_1
# Y_2 = beta_20 + beta_21 * X_21 + beta_22 * X_22 + sigma_2 * e_2
# Y_3 = beta_30 + beta_31 * X_31 + beta_32 * X_32 + sigma_3 * e_3

# X_11 = X_21 = X_31 = Exponential(2)
# X_12 = X_22 = X_32 = Laplace(0, 1)
# e_1 = e_2 = e_3 = Cauchy(0, 1)

seed <- 1234
M <- 3
Stcum1 <- calc_theory("Exponential", 2)
Stcum2 <- calc_theory("Laplace", c(0, 1))
Stcum3 <- c(0, 1, 0, 25, 0, 1500) # taken from paper
means <- lapply(seq_len(M), function(x) c(0, 0, 0))
vars <- lapply(seq_len(M), function(x) c(1, 1, 1))
skews <- lapply(seq_len(M), function(x) c(Stcum1[3], Stcum2[3], Stcum3[3]))
skurts <- lapply(seq_len(M), function(x) c(Stcum1[4], Stcum2[4], Stcum3[4]))
fifths <- lapply(seq_len(M), function(x) c(Stcum1[5], Stcum2[5], Stcum3[5]))
sixths <- lapply(seq_len(M), function(x) c(Stcum1[6], Stcum2[6], Stcum3[6]))

# No sixth cumulant corrections will be used in order to match the results
# from the paper.  Otherwise, the following should be used in order to
# produce variables with valid PDF's:
# Six <- lapply(seq_len(M), function(x) list(NULL, 25.14, NULL))

corr.yx <- list(matrix(c(0.4, 0.4), 1), matrix(c(0.5, 0.5), 1),
  matrix(c(0.6, 0.6), 1))
corr.x <- list()
corr.x[[1]] <- corr.x[[2]] <- corr.x[[3]] <- list()
corr.x[[1]][[1]] <- matrix(c(1, 0.1, 0.1, 1), 2, 2)
corr.x[[1]][[2]] <- matrix(c(0.1974318, 0.1859656, 0.1879483, 0.1858601),
  2, 2, byrow = TRUE)
corr.x[[1]][[3]] <- matrix(c(0.2873190, 0.2589830, 0.2682057, 0.2589542),
```

```
  2, 2, byrow = TRUE)
corr.x[[2]][[1]] <- t(corr.x[[1]][[2]])
corr.x[[2]][[2]] <- matrix(c(1, 0.35, 0.35, 1), 2, 2)
corr.x[[2]][[3]] <- matrix(c(0.5723303, 0.4883054, 0.5004441, 0.4841808),
  2, 2, byrow = TRUE)
corr.x[[3]][[1]] <- t(corr.x[[1]][[3]])
corr.x[[3]][[2]] <- t(corr.x[[2]][[3]])
corr.x[[3]][[3]] <- matrix(c(1, 0.7, 0.7, 1), 2, 2)
corr.e <- matrix(0.4, nrow = 3, ncol = 3)
diag(corr.e) <- 1

# Check the parameter inputs
checkpar(M, "Polynomial", "non_mix", means, vars, skews,
  skurts, fifths, sixths, corr.x = corr.x, corr.yx = corr.yx,
  corr.e = corr.e)
# Generate the system
Sys1 <- nonnormsys(10000, M, "Polynomial", "non_mix", means, vars, skews,
  skurts, fifths, sixths, corr.x = corr.x, corr.yx = corr.yx,
  corr.e = corr.e, seed = seed)
# Summarize the results
Sum1 <- summary_sys(Sys1$Y, Sys1$E, E_mix = NULL, Sys1$X, X_all = list(), M,
  "Polynomial", means, vars, skews, skurts, fifths, sixths, corr.x = corr.x,
  corr.e = corr.e)

# Calculate theoretical correlations for comparison to simulated values
calc_corr_y(Sys1$betas, corr.x, corr.e, vars)
Sum1$rho.y
calc_corr_ye(Sys1$betas, corr.x, corr.e, vars)
Sum1$rho.ye
calc_corr_yx(Sys1$betas, corr.x, vars)
Sum1$rho.yx

## End(Not run)
```

`SimRepeat` *Simulation of Correlated Systems of Statistical Equations with Multiple Variable Types*

Description

Generate correlated systems of statistical equations which represent **repeated measurements** or clustered data. These systems contain either: *a)* continuous normal, non-normal, and mixture variables based on the techniques of Headrick and Beasley (2004, doi: 10.1081/SAC120028431) or *b)* continuous (normal, non-normal and mixture), ordinal, and count (regular or zero-inflated, Poisson and Negative Binomial) variables based on the hierarchical linear models (HLM) approach. Headrick and Beasley's method for continuous variables calculates the beta (slope) coefficients based on the target correlations between independent variables and between outcomes and independent variables. The package provides functions to calculate the expected correlations between outcomes, between outcomes and error terms, and between outcomes and independent variables, extending Headrick and Beasley's equations to include mixture variables. These theoretical values can be compared to the simulated correlations. The HLM approach requires specification of the beta coefficients, but permits group and subject-level independent variables, interactions among independent variables, and fixed and random effects, providing more flexibility in the system of equations. Both methods permit simulation of data sets that mimic real-world clinical or genetic data sets (i.e. plasmodes, as in Vaughan et al., 2009, doi: 10.1016/j.csda.2008.02.032).

The techniques extend those found in the **SimMultiCorrData** and **SimCorrMix** packages. Standard normal variables with an imposed intermediate correlation matrix are transformed to generate the desired distributions. Continuous variables are simulated using either Fleishman's third-order (doi: 10.1007/BF02293811) or Headrick's fifth-order (doi: 10.1016/S01679473(02)000725) power method transformation (PMT). Simulation occurs at the component-level for continuous mixture distributions. These components are transformed into the desired mixture variables using random multinomial variables based on the mixing probabilities. The target correlation matrices are specified in terms of correlations with components of continuous mixture variables. Binary and ordinal variables are simulated by discretizing the normal variables at quantiles defined by the marginal distributions. Count variables are simulated using the inverse CDF method.

There are two simulation pathways for the multi-variable type systems which differ by intermediate correlations involving count variables. Correlation Method 1 adapts Yahav and Shmueli's 2012 method (doi: 10.1002/asmb.901) and performs best with large count variable means and positive correlations or small means and negative correlations. Correlation Method 2 adapts Barbiero and Ferrari's 2015 modification of `GenOrd-package` (doi: 10.1002/asmb.2072) and performs best under the opposite scenarios. There are three methods available for correcting non-positive definite correlation matrices. The optional error loop may be used to improve the accuracy of the final correlation matrices. The package also provides function to check parameter inputs and summarize the generated systems of equations.

Vignettes

There are vignettes which accompany this package that may help the user understand the simulation and analysis methods.

1) **Theory and Equations for Correlated Systems of Continuous Variables** describes the system of continuous variables generated with `nonnormsys` and derives the equations used in `calc_betas`, `calc_corr_y`, `calc_corr_ye`, and `calc_corr_yx`.

2) **Correlated Systems of Statistical Equations with Non-Mixture and Mixture Continuous Variables** provides examples of using `nonnormsys`.

3) **The Hierarchical Linear Models Approach for a System of Correlated Equations with Multiple Variable Types** describes the system of ordinal, continuous, and count variables generated with `corrsys` and `corrsys2`.

4) **Correlated Systems of Statistical Equations with Multiple Variable Types** provides examples of using `corrsys` and `corrsys2`.

Functions

This package contains 3 *simulation* functions:

`nonnormsys`, `corrsys`, `corrsys2`

4 *support* functions for `nonnormsys`:

`calc_betas`, `calc_corr_y`, `calc_corr_ye`, `calc_corr_yx`

1 *parameter check* function:

`checkpar`

1 *summary* function:

`summary_sys`

1 *correction* function for non-PD correlation matrices:

`adj_grad`

References

Amatya A & Demirtas H (2015). Simultaneous generation of multivariate mixed data with Poisson and normal marginals. Journal of Statistical Computation and Simulation, 85(15):3129-39. doi: 10.1080/00949655.2014.953534.

Barbiero A & Ferrari PA (2015). Simulation of correlated Poisson variables. Applied Stochastic Models in Business and Industry, 31:669-80. doi: 10.1002/asmb.2072.

Barbiero A & Ferrari PA (2015). GenOrd: Simulation of Discrete Random Variables with Given Correlation Matrix and Marginal Distributions. R package version 1.4.0.
`https://CRAN.R-project.org/package=GenOrd`

Berend H (2017). nleqslv: Solve Systems of Nonlinear Equations. R package version 3.2. `https://CRAN.R-project.org/package=nleqslv`

Davenport JW, Bezder JC, & Hathaway RJ (1988). Parameter Estimation for Finite Mixture Distributions. Computers & Mathematics with Applications, 15(10):819-28.

Demirtas H (2006). A method for multivariate ordinal data generation given marginal distributions and correlations. Journal of Statistical Computation and Simulation, 76(11):1017-1025.
doi: 10.1080/10629360600569246.

Demirtas H (2014). Joint Generation of Binary and Nonnormal Continuous Data. Biometrics & Biostatistics, S12.

Demirtas H & Hedeker D (2011). A practical way for computing approximate lower and upper correlation bounds. American Statistician, 65(2):104-109. doi: 10.1198/tast.2011.10090.

Demirtas H, Hedeker D, & Mermelstein RJ (2012). Simulation of massive public health data by power polynomials. Statistics in Medicine, 31(27):3337-3346. doi: 10.1002/sim.5362.

Emrich LJ & Piedmonte MR (1991). A Method for Generating High-Dimensional Multivariate Binary Variables. The American Statistician, 45(4): 302-4. doi: 10.1080/00031305.1991.10475828.

Everitt BS (1996). An Introduction to Finite Mixture Distributions. Statistical Methods in Medical Research, 5(2):107-127. doi: 10.1177/096228029600500202.

Ferrari PA & Barbiero A (2012). Simulating ordinal data. Multivariate Behavioral Research, 47(4): 566-589. doi: 10.1080/00273171.2012.692630.

Fialkowski AC (2017). SimMultiCorrData: Simulation of Correlated Data with Multiple Variable Types. R package version 0.2.1. `https://CRAN.R-project.org/package=SimMultiCorrData`.

Fialkowski AC (2018). SimCorrMix: Simulation of Correlated Data of Multiple Variable Types including Continuous and Count Mixture Distributions. R package version 0.1.0. `https://github.com/AFialkowski/SimCorrMix`

Fleishman AI (1978). A Method for Simulating Non-normal Distributions. Psychometrika, 43:521-532. doi: 10.1007/BF02293811.

Frechet M (1951). Sur les tableaux de correlation dont les marges sont donnees. Ann. l'Univ. Lyon SectA, 14:53-77.

Headrick TC (2002). Fast Fifth-order Polynomial Transforms for Generating Univariate and Multivariate Non-normal Distributions. Computational Statistics & Data Analysis, 40(4):685-711. doi: 10.1016/S01679473(02)000725. (ScienceDirect)

Headrick TC (2004). On Polynomial Transformations for Simulating Multivariate Nonnormal Distributions. Journal of Modern Applied Statistical Methods, 3(1):65-71. doi: 10.22237/jmasm/1083370080.

Headrick TC, Beasley TM (2004). A Method for Simulating Correlated Non-Normal Systems of Linear Statistical Equations. Communications in Statistics - Simulation and Computation, 33(1). doi: 10.1081/SAC120028431

Headrick TC, Kowalchuk RK (2007). The Power Method Transformation: Its Probability Density Function, Distribution Function, and Its Further Use for Fitting Data. Journal of Statistical Computation and Simulation, 77:229-249. doi: 10.1080/10629360600605065.

Headrick TC, Sawilowsky SS (1999). Simulating Correlated Non-normal Distributions: Extending the Fleishman Power Method. Psychometrika, 64:25-35. doi: 10.1007/BF02294317.

Headrick TC, Sawilowsky SS (2002). Weighted Simplex Procedures for Determining Boundary Points and Constants for the Univariate and Multivariate Power Methods. Journal of Educational and Behavioral Statistics, 25:417-436. doi: 10.3102/10769986025004417.

Headrick TC, Sheng Y, & Hodis FA (2007). Numerical Computing and Graphics for the Power Method Transformation Using Mathematica. Journal of Statistical Software, 19(3):1 - 17. doi: 10.18637/jss.v019.i03.

Higham N (2002). Computing the nearest correlation matrix - a problem from finance; IMA Journal of Numerical Analysis 22:329-343.

Hoeffding W. Scale-invariant correlation theory. In: Fisher NI, Sen PK, editors. The collected works of Wassily Hoeffding. New York: Springer-Verlag; 1994. p. 57-107.

Ismail N & Zamani H (2013). Estimation of Claim Count Data Using Negative Binomial, Generalized Poisson, Zero-Inflated Negative Binomial and Zero-Inflated Generalized Poisson Regression Models. Casualty Actuarial Society E-Forum 41(20):1-28.

Kincaid C (2005). Guidelines for Selecting the Covariance Structure in Mixed Model Analysis. Computational Statistics and Data Analysis, 198(30):1-8.

Lambert D (1992). Zero-Inflated Poisson Regression, with an Application to Defects in Manufacturing. Technometrics 34(1):1-14.

Lininger M, Spybrook J, & Cheatham CC (2015). Hierarchical Linear Model: Thinking Outside the Traditional Repeated-Measures Analysis-of-Variance Box. Journal of Athletic Training, 50(4):438-441. doi: 10.4085/1062605049.5.09.

McCulloch CE, Searle SR, Neuhaus JM (2008). *Generalized, Linear, and Mixed Models* (2nd ed.). Wiley Series in Probability and Statistics. Hoboken, New Jersey: John Wiley & Sons, Inc.

Olsson U, Drasgow F, & Dorans NJ (1982). The Polyserial Correlation Coefficient. Psychometrika, 47(3):337-47. doi: 10.1007/BF02294164.

Pearson RK (2011). Exploring Data in Engineering, the Sciences, and Medicine. In. New York: Oxford University Press.

Schork NJ, Allison DB, & Thiel B (1996). Mixture Distributions in Human Genetics Research. Statistical Methods in Medical Research, 5:155-178. doi: 10.1177/096228029600500204.

Vale CD & Maurelli VA (1983). Simulating Multivariate Nonnormal Distributions. Psychometrika, 48:465-471. doi: 10.1007/BF02293687.

Van Der Leeden R (1998). Multilevel Analysis of Repeated Measures Data. Quality & Quantity, 32(1):15-29.

Varadhan R, Gilbert PD (2009). BB: An R Package for Solving a Large System of Nonlinear Equations and for Optimizing a High-Dimensional Nonlinear Objective Function, J. Statistical Software, 32(4). doi: 10.18637/jss.v032.i04. `http://www.jstatsoft.org/v32/i04/`

Vaughan LK, Divers J, Padilla M, Redden DT, Tiwari HK, Pomp D, Allison DB (2009). The use of plasmodes as a supplement to simulations: A simple example evaluating individual admixture estimation methodologies. Comput Stat Data Anal, 53(5):1755-66. doi: 10.1016/j.csda.2008.02.032.

Yahav I & Shmueli G (2012). On Generating Multivariate Poisson Data in Management Science Applications. Applied Stochastic Models in Business and Industry, 28(1):91-102. doi: 10.1002/asmb.901.

Yee TW (2017). VGAM: Vector Generalized Linear and Additive Models. https://CRAN.R-project.org/package=VGAM.

Zhang X, Mallick H, & Yi N (2016). Zero-Inflated Negative Binomial Regression for Differential Abundance Testing in Microbiome Studies. Journal of Bioinformatics and Genomics 2(2):1-9. doi: 10.18454/jbg.2016.2.2.1.

See Also

Useful link: https://github.com/AFialkowski/SimMultiCorrData, https://github.com/AFialkowski/SimCorrMix, https://github.com/AFialkowski/SimRepeat

---

summary_sys — *Summary of Correlated Systems of Variables*

Description

This function summarizes the results of nonnormsys, corrsys, or corrsys2. The inputs are either the simulated variables or inputs for those functions. See their documentation for more information. If only selected descriptions are desired, keep the non-relevant parameter inputs at their defaults. For example, if only a description of the error terms are desired, error_type = "non_mix", and method = "Polynomial", specify E, M, method, means, vars, skews, skurts, fifths, sixths, corr.e.

Usage

```
summary_sys(Y = NULL, E = NULL, E_mix = NULL, X = list(),
  X_all = list(), M = NULL, method = c("Fleishman", "Polynomial"),
  means = list(), vars = list(), skews = list(), skurts = list(),
  fifths = list(), sixths = list(), mix_pis = list(), mix_mus = list(),
  mix_sigmas = list(), mix_skews = list(), mix_skurts = list(),
  mix_fifths = list(), mix_sixths = list(), marginal = list(),
  support = list(), lam = list(), p_zip = list(), size = list(),
  prob = list(), mu = list(), p_zinb = list(), corr.x = list(),
  corr.e = NULL, U = list(), U_all = list(), rand.int = c("none",
  "non_mix", "mix"), rand.tsl = c("none", "non_mix", "mix"),
  corr.u = list(), rmeans2 = list(), rvars2 = list())
```

Arguments

Y	the matrix of outcomes simulated with corrsys or corrsys2
E	the matrix of continuous non-mixture or components of mixture error terms
E_mix	the matrix of continuous mixture error terms
X	a list of length M where X[[p]] = cbind(X_cat(pj), X_cont(pj), X_comp(pj), X_pois(pj), X keep X[[p]] = NULL if Y_p has no independent variables
X_all	a list of length M where X_all[[p]] contains all independent variables, interactions, and time for Y_p; keep X_all[[p]] = NULL if Y_p has no independent variables
M	the number of dependent variables Y (outcomes); equivalently, the number of equations in the system

`method` the PMT method used to generate all continuous variables, including independent variables (covariates), error terms, and random effects; "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation

`means` if no random effects, a list of length `M` where `means[[p]]` contains a vector of means for the continuous independent variables in equation p with non-mixture (X_{cont}) or mixture (X_{mix}) distributions and for the error terms (E); order in vector is X_{cont}, X_{mix}, E

if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `means[M + 1]` or `means[(M + 1):(2 * M)]` are vectors of means for all random effects with continuous non-mixture or mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int != "none"`), 2nd random time slope U_1 (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions U_{cont}, 4th other random slopes with mixture distributions U_{mix}

`vars` a list of same length and order as `means` containing vectors of variances for the continuous variables, error terms, and any random effects

`skews` if no random effects, a list of length `M` where `skews[[p]]` contains a vector of skew values for the continuous independent variables in equation p with non-mixture (X_{cont}) distributions and for E if `error_type = "non_mix"`; order in vector is X_{cont}, E

if there are random effects, a list of length `M + 1` if the effects are the same across equations or `2 * M` if they differ; where `skews[M + 1]` or `skews[(M + 1):(2 * M)]` are vectors of skew values for all random effects with continuous non-mixture distributions; order in vector is 1st random intercept U_0 (if `rand.int = "non_mix"`), 2nd random time slope U_1 (if `rand.tsl = "non_mix"`), 3rd other random slopes with non-mixture distributions U_{cont}

`skurts` a list of same length and order as `skews` containing vectors of standardized kurtoses (kurtosis - 3) for the continuous variables, error terms, and any random effects with non-mixture distributions

`fifths` a list of same length and order as `skews` containing vectors of standardized fifth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for `method = "Fleishman"`

`sixths` a list of same length and order as `skews` containing vectors of standardized sixth cumulants for the continuous variables, error terms, and any random effects with non-mixture distributions; not necessary for `method = "Fleishman"`

`mix_pis` list of length `M`, `M + 1` or `2 * M`, where `mix_pis[1:M]` are for X_{cont}, E (if `error_type = "mix"`) and `mix_pis[M + 1]` or `mix_pis[(M + 1):(2 * M)]` are for mixture U; use `mix_pis[[p]] = NULL` if equation p has no continuous mixture terms if `error_type = "non_mix"` and there are only random effects (i.e., `length(corr.x) = 0`), use `mix_pis[1:M] = NULL` so that `mix_pis[M + 1]` or `mix_pis[(M + 1):(2 * M)]` describes the mixture U;

`mix_pis[[p]][[j]]` is a vector of mixing probabilities of the component distributions for $X_{mix(pj)}$, the j-th mixture covariate for outcome Y_p; the last vector in `mix_pis[[p]]` is for E_p (if `error_type = "mix"`); components should be ordered as in `corr.x`

`mix_pis[[M + p]][[j]]` is a vector of mixing probabilities of the component distributions for $U_{(pj)}$, the j-th random effect with a mixture distribution for outcome Y_p; order is 1st random intercept (if `rand.int = "mix"`), 2nd random time slope (if `rand.tsl = "mix"`), 3rd other random slopes with mixture distributions; components should be ordered as in `corr.u`

mix_mus
: list of same length and order as mix_pis;

mix_mus[[p]][[j]] is a vector of means of the component distributions for $X_{mix(pj)}$, the last vector in mix_mus[[p]] is for E_p (if error_type = "mix")

mix_mus[[p]][[j]] is a vector of means of the component distributions for $U_{mix(pj)}$

mix_sigmas
: list of same length and order as mix_pis;

mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $X_{mix(pj)}$, the last vector in mix_sigmas[[p]] is for E_p (if error_type = "mix")

mix_sigmas[[p]][[j]] is a vector of standard deviations of the component distributions for $U_{mix(pj)}$

mix_skews
: list of same length and order as mix_pis;

mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $X_{mix(pj)}$, the last vector in mix_skews[[p]] is for E_p (if error_type = "mix")

mix_skews[[p]][[j]] is a vector of skew values of the component distributions for $U_{mix(pj)}$

mix_skurts
: list of same length and order as mix_pis;

mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $X_{mix(pj)}$, the last vector in mix_skurts[[p]] is for E_p (if error_type = "mix")

mix_skurts[[p]][[j]] is a vector of standardized kurtoses of the component distributions for $U_{mix(pj)}$

mix_fifths
: list of same length and order as mix_pis; not necessary for method = "Fleishman";

mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_fifths[[p]] is for E_p (if error_type = "mix")

mix_fifths[[p]][[j]] is a vector of standardized fifth cumulants of the component distributions for $U_{mix(pj)}$

mix_sixths
: list of same length and order as mix_pis; not necessary for method = "Fleishman";

mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $X_{mix(pj)}$, the last vector in mix_sixths[[p]] is for E_p (if error_type = "mix")

mix_sixths[[p]][[j]] is a vector of standardized sixth cumulants of the component distributions for $U_{mix(pj)}$

marginal
: a list of length M, with the p-th component a list of cumulative probabilities for the ordinal variables associated with outcome Y_p (use marginal[[p]] = NULL if outcome Y_p has no ordinal variables); marginal[[p]][[j]] is a vector of the cumulative probabilities defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1); for binary variables, the probability is the probability of the 1st category, which has the smaller support value; length(marginal[[p]]) can differ across outcomes; the order should be the same as in corr.x

support
: a list of length M, with the p-th component a list of support values for the ordinal variables associated with outcome Y_p; use support[[p]] = NULL if outcome Y_p has no ordinal variables; support[[p]][[j]] is a vector of the support values defining the marginal distribution of $X_{ord(pj)}$, the j-th ordinal variable for outcome Y_p; if not provided, the default for r categories is 1, ..., r

lam
: list of length M, p-th component a vector of lambda (means > 0) values for Poisson variables for outcome Y_p (see stats::dpois); order is 1st regular Poisson and 2nd zero-inflated Poisson; use lam[[p]] = NULL if outcome Y_p has no Poisson variables; length(lam[[p]]) can differ across outcomes; the order should be the same as in corr.x

p_zip
: a list of vectors of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see VGAM::dzipois); if p_zip = 0, Y_{pois} has a regular Poisson distribution; if p_zip is in (0, 1), Y_{pois} has a zero-inflated Poisson distribution; if p_zip is in (-(exp(lam) - 1)^(-1), 0), Y_{pois} has a zero-deflated Poisson distribution and p_zip is not a probability; if p_zip = -(exp(lam) - 1)^(-1), Y_{pois} has a positive-Poisson distribution (see VGAM::dpospois); order is 1st regular Poisson and 2nd zero-inflated Poisson; if a single number, all Poisson variables given this value; if a vector of length M, all Poisson variables in equation p given p_zip[p]; otherwise, missing values are set to 0 and ordered 1st

size
: list of length M, p-th component a vector of size parameters for the Negative Binomial variables for outcome Y_p (see stats::nbinom); order is 1st regular NB and 2nd zero-inflated NB; use size[[p]] = NULL if outcome Y_p has no Negative Binomial variables; length(size[[p]]) can differ across outcomes; the order should be the same as in corr.x

prob
: list of length M, p-th component a vector of success probabilities for the Negative Binomial variables for outcome Y_p (see stats::nbinom); order is 1st regular NB and 2nd zero-inflated NB; use prob[[p]] = NULL if outcome Y_p has no Negative Binomial variables; length(prob[[p]]) can differ across outcomes; the order should be the same as in corr.x

mu
: list of length M, p-th component a vector of mean values for the Negative Binomial variables for outcome Y_p (see stats::nbinom); order is 1st regular NB and 2nd zero-inflated NB; use mu[[p]] = NULL if outcome Y_p has no Negative Binomial variables; length(mu[[p]]) can differ across outcomes; the order should be the same as in corr.x; for zero-inflated NB variables, this refers to the mean of the NB distribution (see VGAM::dzinegbin) (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture)

p_zinb
: a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see VGAM::dzinegbin); if p_zinb = 0, Y_{nb} has a regular NB distribution; if p_zinb is in (-prob^size/(1 - prob^size), 0), Y_{nb} has a zero-deflated NB distribution and p_zinb is not a probability; if p_zinb = -prob^size/(1 - prob^size), Y_{nb} has a positive-NB distribution (see VGAM::dposnegbin); order is 1st regular NB and 2nd zero-inflated NB; if a single number, all NB variables given this value; if a vector of length M, all NB variables in equation p given p_zinb[p]; otherwise, missing values are set to 0 and ordered 1st

corr.x
: list of length M, each component a list of length M; corr.x[[p]][[q]] is matrix of correlations for independent variables in equations p ($X_{(pj)}$ for outcome Y_p) and q ($X_{(qj)}$ for outcome Y_q); order: 1st ordinal (same order as in marginal), 2nd continuous non-mixture (same order as in skews), 3rd components of continuous mixture (same order as in mix_pis), 4th regular Poisson, 5th zero-inflated Poisson (same order as in lam), 6th regular NB, and 7th zero-inflated NB (same order as in size); if p = q, corr.x[[p]][[q]] is a correlation matrix with nrow(corr.x[[p]][[q]]) = # $X_{(pj)}$ for outcome Y_p; if p != q, corr.x[[p]][[q]] is a non-symmetric matrix of correlations where rows correspond to covariates for Y_p so that nrow(corr.x[[p]][[q]]) = #

	$X_{(pj)}$ for outcome Y_p and columns correspond to covariates for Y_q so that `ncol(corr.x[[p]][[q]])` = # $X_{(qj)}$ for outcome Y_q; use `corr.x[[p]][[q]] = NULL` if equation q has no $X_{(qj)}$; use `corr.x[[p]] = NULL` if equation p has no $X_{(pj)}$
`corr.e`	correlation matrix for continuous non-mixture or components of mixture error terms
`U`	a list of length `M` of continuous non-mixture and components of mixture random effects
`U_all`	a list of length `M` of continuous non-mixture and mixture random effects
`rand.int`	"none" (default) if no random intercept term for all outcomes, "non_mix" if all random intercepts have a continuous non-mixture distribution, "mix" if all random intercepts have a continuous mixture distribution; also can be a vector of length `M` containing a combination (i.e., `c("non_mix", "mix", "none")` if the 1st has a non-mixture distribution, the 2nd has a mixture distribution, and 3rd outcome has no random intercept)
`rand.tsl`	"none" (default) if no random slope for time for all outcomes, "non_mix" if all random time slopes have a continuous non-mixture distribution, "mix" if all random time slopes have a continuous mixture distribution; also can be a vector of length `M` as in `rand.int`
`corr.u`	if the random effects are the same variables across equations, a matrix of correlations for U; if the random effects are different variables across equations, a list of length `M`, each component a list of length `M`; `corr.u[[p]][[q]]` is matrix of correlations for random effects in equations p ($U_{(pj)}$ for outcome Y_p) and q ($U_{(qj)}$ for outcome Y_q); if p = q, `corr.u[[p]][[q]]` is a correlation matrix with `nrow(corr.u[[p]][[q]])` = # $U_{(pj)}$ for outcome Y_p; if p != q, `corr.u[[p]][[q]]` is a non-symmetric matrix of correlations where rows correspond to $U_{(pj)}$ for Y_p so that `nrow(corr.u[[p]][[q]])` = # $U_{(pj)}$ for outcome Y_p and columns correspond to $U_{(qj)}$ for Y_q so that `ncol(corr.u[[p]][[q]])` = # $U_{(qj)}$ for outcome Y_q; the number of random effects for Y_p is taken from `nrow(corr.u[[p]][[1]])` so that if there should be random effects, there must be entries for `corr.u`; use `corr.u[[p]][[q]] = NULL` if equation q has no $U_{(qj)}$; use `corr.u[[p]] = NULL` if equation p has no $U_{(pj)}$; correlations are specified in terms of components of mixture variables (if present); order is 1st random intercept (if `rand.int != "none"`), 2nd random time slope (if `rand.tsl != "none"`), 3rd other random slopes with non-mixture distributions, 4th other random slopes with mixture distributions
`rmeans2`	a list returned from `corrsys` or `corrsys2` which has the non-mixture and component means ordered according to types of random intercept and time slope
`rvars2`	a list returned like `rmeans`

Value

A list with the following components:

`cont_sum_y` a data.frame summarizing the simulated distributions of the Y_p,

`cont_sum_e` a data.frame summarizing the simulated distributions of the non-mixture or components of mixture E_p,

`target_sum_e` a data.frame summarizing the target distributions of the non-mixture or components of mixture E_p,

`mix_sum_e` a data.frame summarizing the simulated distributions of the mixture E_p,

`target_mix_e` a data.frame summarizing the target distributions of the mixture E_p, rho.y correlation matrix of dimension M x M for Y_p

rho.e correlation matrix for the non-mixture or components of mixture E_p

rho.emix correlation matrix for the mixture E_p

rho.ye matrix with correlations between Y_p (rows) and the non-mixture or components of mixture E_p (columns)

rho.yemix matrix with correlations between Y_p (rows) and the mixture E_p (columns)

sum_xall a data.frame summarizing X_all without the Time variable,

rho.yx a list of length M, where rho.yx[[p]] is matrix of correlations between Y (rows) and X[[p]] = $X_ord(pj), X_cont(pj), X_comp(pj), X_pois(pj), X_nb(pj)$ (columns)

rho.yxall a list of length M, where rho.yx[[p]] is matrix of correlations between Y (rows) and X_all[[p]] (columns) not including Time

rho.x a list of length M of lists of length M where rho.x[[p]][[q]] = cor(cbind(X[[p]], X[[q]])) if p!= q or rho.x[[p]][[q]] = cor(X[[p]])) if p = q, where X[[p]] = $X_ord(pj), X_cont(pj), X_comp(pj), X_pois(pj)$

rho.xall a list of length M of lists of length M where rho.xall[[p]][[q]] = cor(cbind(X_all[[p]], X_all[[q]])) if p!= q or rho.xall[[p]][[q]] = cor(X_all[[p]])) if p = q, not including Time

maxerr a list of length M containing a vector of length M with the maximum correlation errors between outcomes, maxerr[[p]]][q] = abs(max(corr.x[[p]][[q]] - rho.x[[p]][[q]]))

Additional components vary based on the type of simulated variables:

If **ordinal variables** are produced: ord_sum_x a list where ord_sum_x[[j]] is a data.frame summarizing $X_{ord(pj)}$ for all p = 1, ..., M

If **continuous variables** are produced: cont_sum_x a data.frame summarizing the simulated distributions of the $X_{cont(pj)}$ and $X_comp(pj)$,

target_sum_x a data.frame summarizing the target distributions of the $X_{cont(pj)}$ and $X_comp(pj)$,

mix_sum_x a data.frame summarizing the simulated distributions of the $X_{mix(pj)}$,

target_mix_x a data.frame summarizing the target distributions of the $X_{mix(pj)}$

If **Poisson variables** are produced: pois_sum_x a data.frame summarizing the simulated distributions of the $X_{pois(pj)}$

If **Negative Binomial variables** are produced: nb_sum_x a data.frame summarizing the simulated distributions of the $X_{nb(pj)}$

If **random effects** are produced: cont_sum_u a data.frame summarizing the simulated distributions of the $U_{cont(pj)}$ and $U_{comp(pj)}$,

target_sum_u a data.frame summarizing the target distributions of the $U_{cont(pj)}$ and $U_{comp(pj)}$,

sum_uall a data.frame summarizing the simulated distributions of U_all,

mix_sum_u a data.frame summarizing the simulated distributions of the $U_{mix(pj)}$,

target_mix_u a data.frame summarizing the target distributions of the $U_{mix(pj)}$,

rho.u list of length M, each component a list of length M; rho.u[[p]][[q]] = cor(cbind(U[[p]], U[[q]])) if p != q or rho.u[[p]][[q]] = cor(U[[p]])) if p = q

rho.uall list of length M, each component a list of length M; rho.uall[[p]][[q]] = cor(cbind(U_all[[p]], U_all[if p != q or rho.uall[[p]][[q]] = cor(U_all[[p]])) if p = q

maxerr_u list of length M containing a vector of length M with the maximum correlation errors for U between outcomes maxerr_u[[p]]][q] = abs(max(corr.u[[p]][[q]] - rho.u[[p]][[q]]))

References

See references for SimRepeat.

See Also

nonnormsys, corrsys, corrsys2

Examples

```
M <- 3
B <- calc_theory("Beta", c(4, 1.5))
skews <- lapply(seq_len(M), function(x) B[3])
skurts <- lapply(seq_len(M), function(x) B[4])
fifths <- lapply(seq_len(M), function(x) B[5])
sixths <- lapply(seq_len(M), function(x) B[6])
Six <- lapply(seq_len(M), function(x) list(0.03))
corr.e <- matrix(c(1, 0.4, 0.4^2, 0.4, 1, 0.4, 0.4^2, 0.4, 1), M, M,
  byrow = TRUE)
means <- lapply(seq_len(M), function(x) B[1])
vars <- lapply(seq_len(M), function(x) B[2]^2)
marginal <- list(0.3, 0.4, 0.5)
support <- lapply(seq_len(M), function(x) list(0:1))
corr.x <- list(list(matrix(1, 1, 1), matrix(0.4, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(1, 1, 1), matrix(0.4, 1, 1)),
  list(matrix(0.4, 1, 1), matrix(0.4, 1, 1), matrix(1, 1, 1)))
betas <- list(0.5)
betas.t <- 1
betas.tint <- list(0.25)
Sys1 <- corrsys(10000, M, Time = 1:M, "Polynomial", "non_mix", means, vars,
  skews, skurts, fifths, sixths, Six, marginal = marginal, support = support,
  corr.x = corr.x, corr.e = corr.e, betas = betas, betas.t = betas.t,
  betas.tint = betas.tint, quiet = TRUE)
Sum1 <- summary_sys(Sys1$Y, Sys1$E, E_mix = NULL, Sys1$X, Sys1$X_all, M,
  "Polynomial", means, vars, skews, skurts, fifths, sixths,
  marginal = marginal, support = support, corr.x = corr.x, corr.e = corr.e)

## Not run:
seed <- 276
n <- 10000
M <- 3
Time <- 1:M

# Error terms have a beta(4, 1.5) distribution with an AR(1, p = 0.4)
correlation structure
B <- calc_theory("Beta", c(4, 1.5))
skews <- lapply(seq_len(M), function(x) B[3])
skurts <- lapply(seq_len(M), function(x) B[4])
fifths <- lapply(seq_len(M), function(x) B[5])
sixths <- lapply(seq_len(M), function(x) B[6])
Six <- lapply(seq_len(M), function(x) list(0.03))
error_type <- "non_mix"
corr.e <- matrix(c(1, 0.4, 0.4^2, 0.4, 1, 0.4, 0.4^2, 0.4, 1), M, M,
  byrow = TRUE)

1 continuous mixture of Normal(-2, 1) and Normal(2, 1) for each Y
mix_pis <- lapply(seq_len(M), function(x) list(c(0.4, 0.6)))
mix_mus <- lapply(seq_len(M), function(x) list(c(-2, 2)))
mix_sigmas <- lapply(seq_len(M), function(x) list(c(1, 1)))
mix_skews <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_skurts <- lapply(seq_len(M), function(x) list(c(0, 0)))
```

```
mix_fifths <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_sixths <- lapply(seq_len(M), function(x) list(c(0, 0)))
mix_Six <- list()
Nstcum <- calc_mixmoments(mix_pis[[1]][[1]], mix_mus[[1]][[1]],
  mix_sigmas[[1]][[1]], mix_skews[[1]][[1]], mix_skurts[[1]][[1]],
  mix_fifths[[1]][[1]], mix_sixths[[1]][[1]])

means <- lapply(seq_len(M), function(x) c(Nstcum[1], B[1]))
vars <- lapply(seq_len(M), function(x) c(Nstcum[2]^2, B[2]^2))

# 1 binary variable for each Y
marginal <- lapply(seq_len(M), function(x) list(0.4))
support <- list(NULL, list(c(0, 1)), NULL)

# 1 Poisson variable for each Y
lam <- list(1, 5, 10)
# Y2 and Y3 have zero-inflated Poisson variables
p_zip <- list(NULL, 0.05, 0.1)

# 1 NB variable for each Y
size <- list(10, 15, 20)
prob <- list(0.3, 0.4, 0.5)
# either prob or mu is required (not both)
mu <- mapply(function(x, y) x * (1 - y)/y, size, prob, SIMPLIFY = FALSE)
# Y2 and Y3 have zero-inflated NB variables
p_zinb <- list(NULL, 0.05, 0.1)

# The 2nd (the normal mixture) variable is the same across Y
same.var <- 2

# Create the correlation matrix in terms of the components of the normal
# mixture
K <- 5
corr.x <- list()
corr.x[[1]] <- list(matrix(0.1, K, K), matrix(0.2, K, K), matrix(0.3, K, K))
diag(corr.x[[1]][[1]]) <- 1
# set correlation between components to 0
corr.x[[1]][[1]][2:3, 2:3] <- diag(2)
# set correlations with the same variable equal across outcomes
corr.x[[1]][[2]][, same.var] <- corr.x[[1]][[3]][, same.var] <-
  corr.x[[1]][[1]][, same.var]
corr.x[[2]] <- list(t(corr.x[[1]][[2]]), matrix(0.35, K, K),
  matrix(0.4, K, K))
  diag(corr.x[[2]][[2]]) <- 1
  corr.x[[2]][[2]][2:3, 2:3] <- diag(2)
corr.x[[2]][[2]][, same.var] <- corr.x[[2]][[3]][, same.var] <-
  t(corr.x[[1]][[2]][same.var, ])
corr.x[[2]][[3]][same.var, ] <- corr.x[[1]][[3]][same.var, ]
corr.x[[2]][[2]][same.var, ] <- t(corr.x[[2]][[2]][, same.var])
corr.x[[3]] <- list(t(corr.x[[1]][[3]]), t(corr.x[[2]][[3]]),
  matrix(0.5, K, K))
diag(corr.x[[3]][[3]]) <- 1
corr.x[[3]][[3]][2:3, 2:3] <- diag(2)
corr.x[[3]][[3]][, same.var] <- t(corr.x[[1]][[3]][same.var, ])
corr.x[[3]][[3]][same.var, ] <- t(corr.x[[3]][[3]][, same.var])

# The 2nd and 3rd variables of each Y are subject-level variables
```

```
subj.var <- matrix(c(1, 2, 1, 3, 2, 2, 2, 3, 3, 2, 3, 3), 6, 2, byrow = TRUE)
int.var <- tint.var <- NULL
betas.0 <- 0
betas <- list(seq(0.5, 0.5 + (K - 2) * 0.25, 0.25))
betas.subj <- list(seq(0.5, 0.5 + (K - 2) * 0.1, 0.1))
betas.int <- list()
betas.t <- 1
betas.tint <- list(c(0.25, 0.5))

method <- "Polynomial"

# Check parameter inputs
checkpar(M, method, error_type, means, vars, skews, skurts, fifths, sixths,
  Six, mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts, mix_fifths,
  mix_sixths, mix_Six, marginal, support, lam, p_zip, pois_eps = list(),
  size, prob, mu, p_zinb, nb_eps = list(), corr.x, corr.yx = list(),
  corr.e, same.var, subj.var, int.var, tint.var, betas.0, betas,
  betas.subj, betas.int, betas.t, betas.tint)

# Simulated system using correlation method 1
N <- corrsys(n, M, Time, method, error_type, means, vars, skews, skurts,
  fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts,
  mix_fifths, mix_sixths, mix_Six, marginal, support, lam, p_zip, size,
  prob, mu, p_zinb, corr.x, corr.e, same.var, subj.var, int.var, tint.var,
  betas.0, betas, betas.subj, betas.int, betas.t, betas.tint, seed = seed,
  use.nearPD = FALSE)

# Summarize the results
S <- summary_sys(N$Y, N$E, E_mix = NULL, N$X, N$X_all, M, method, means,
  vars, skews, skurts, fifths, sixths, mix_pis, mix_mus, mix_sigmas,
  mix_skews, mix_skurts, mix_fifths, mix_sixths, marginal, support, lam,
  p_zip, size, prob, mu, p_zinb, corr.x, corr.e)
S$sum_xall
S$maxerr

## End(Not run)
```

Index