Package 'SimCorrMix'

January 6, 2018

Type Package

Title Simulation of Correlated Data of Multiple Variable Types including Continuous and Count Mixture Distributions

Version 0.1.0

Author Allison Cynthia Fialkowski

Maintainer Allison Cynthia Fialkowski <allijazz@uab.edu>

Description Generate continuous (normal, non-normal, or mixture distributions), binary, ordinal, and count (regular or zero-inflated, Poisson or Negative Binomial) variables with a specified correlation matrix, or one continuous variable with a mixture distribution. This package can be used to simulate 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 methods extend those found in the SimMultiCorrData R package. Standard normal variables with an imposed intermediate correlation matrix are transformed to generate the desired distributions. Continuous variables are simulated using either Fleishman (1978)'s third order <DOI:10.1007/BF02293811> or Headrick (2002)'s fifth order <DOI:10.1016/S0167-9473(02)00072-5> polynomial transformation method (the power method transformation, PMT). Non-mixture distributions require the user to specify mean, variance, skewness, standardized kurtosis, and standardized fifth and sixth cumulants. Mixture distributions require these inputs for the component distributions plus the mixing probabilities. 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. However, the package provides functions to approximate expected correlations with continuous mixture variables given target correlations with the components. Binary and ordinal variables are simulated using a modification of package GenOrd's ordsample function. Count variables are simulated using the inverse CDF method. There are two simulation pathways which calculate intermediate correlations involving count variables differently. Correlation Method 1 adapts Yahav and Shmueli's 2012 method <DOI:10.1002/asmb.901>. Correlation Method 2 adapts Barbiero and Ferrari's 2015 modification of the GenOrd package <DOI:10.1002/asmb.2072>. The optional error loop may be used to improve the accuracy of the final correlation matrix. The package also contains functions to calculate the standardized cumulants of continuous mixture distributions, check parameter inputs, calculate feasible correlation boundaries, and summarize and plot simulated variables.

Depends R (>= 3.3.1),
SimMultiCorrData (>= 0.2.1)

License GPL-2

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

Encoding UTF-8

LazyData true

Roxygen list(wrap = FALSE)

RoxygenNote 6.0.1

Suggests knitr,
rmarkdown, printr,
testthat

VignetteBuilder knitr

URL https://github.com/AFialkowski/SimCorrMix

R topics documented:

calc_mixmoments . . . 3
contmixvar1 . . . 4
corrvar . . . 8
corrvar2 . . . 16
corr_error . . . 24
intercorr . . . 26
intercorr2 . . . 29
intercorr_cat_nb . . . 31
intercorr_cat_pois . . . 33
intercorr_cont . . . 34
intercorr_cont_nb . . . 35
intercorr_cont_nb2 . . . 37
intercorr_cont_pois . . . 38
intercorr_cont_pois2 . . . 40
intercorr_nb . . . 42
intercorr_pois . . . 43
intercorr_pois_nb . . . 44
maxcount_support . . . 46
norm_ord . . . 47
ord_norm . . . 48
plot_simpdf_theory . . . 50
plot_simtheory . . . 52
rho_M1M2 . . . 55
rho_M1Y . . . 56
SimCorrMix . . . 57
summary_var . . . 61
validcorr . . . 65
validcorr2 . . . 69
validpar . . . 74

Index **79**

`calc_mixmoments` *Find Standardized Cumulants of a Continuous Mixture Distribution by Method of Moments*

Description

This function uses the method of moments to calculate the expected mean, standard deviation, skewness, standardized kurtosis, and standardized fifth and sixth cumulants for a continuous mixture variable based on the distributions of its components. The result can be used as input to `find_constants` or for comparison to a simulated mixture variable from `contmixvar1`, `corrvar`, or `corrvar2`. See the **Expected Cumulants and Correlations for Continuous Mixture Variables** vignette for equations of the cumulants.

Usage

```
calc_mixmoments(mix_pis = NULL, mix_mus = NULL, mix_sigmas = NULL,
  mix_skews = NULL, mix_skurts = NULL, mix_fifths = NULL,
  mix_sixths = NULL)
```

Arguments

`mix_pis`	a vector of mixing probabilities that sum to 1 for the component distributions
`mix_mus`	a vector of means for the component distributions
`mix_sigmas`	a vector of standard deviations for the component distributions
`mix_skews`	a vector of skew values for the component distributions
`mix_skurts`	a vector of standardized kurtoses for the component distributions
`mix_fifths`	a vector of standardized fifth cumulants for the component distributions; keep NULL if using `method` = "Fleishman" to generate continuous variables
`mix_sixths`	a vector of standardized sixth cumulants for the component distributions; keep NULL if using `method` = "Fleishman" to generate continuous variables

Value

A vector of the mean, standard deviation, skewness, standardized kurtosis, and standardized fifth and sixth cumulants

References

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

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, 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, 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.

Kendall M & Stuart A (1977). The Advanced Theory of Statistics, 4th Edition. Macmillan, New York.

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.

Examples

```
# Two mixture variables: 1st is mixture of Normal(-2, 1) and Normal(2, 1),
# 2nd is mixture of Logistic(0, 1), Chisq(4), and Beta(4, 1.5)
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
mix_skews <- list(rep(0, 2), c(L[3], C[3], B[3]))
mix_skurts <- list(rep(0, 2), c(L[4], C[4], B[4]))
mix_fifths <- list(rep(0, 2), c(L[5], C[5], B[5]))
mix_sixths <- list(rep(0, 2), c(L[6], C[6], B[6]))
Nstcum <- calc_mixmoments(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]],
  mix_skews[[1]], mix_skurts[[1]], mix_fifths[[1]], mix_sixths[[1]])
Nstcum
Mstcum <- calc_mixmoments(mix_pis[[2]], mix_mus[[2]], mix_sigmas[[2]],
  mix_skews[[2]], mix_skurts[[2]], mix_fifths[[2]], mix_sixths[[2]])
Mstcum
```

`contmixvar1` *Generation of One Continuous Variable with a Mixture Distribution Using the Power Method Transformation*

Description

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

$$h_Y(y) = \sum_{i=1}^{k} \pi_i f_{Yi}(y), \sum_{i=1}^{k} \pi_i = 1.$$

The π_i are mixing parameters which determine the weight of each component distribution $f_{Yi}(y)$ in the overall probability distribution. As long as each component has a valid PDF, the overall distribution $h_Y(y)$ has a valid PDF. The main assumption is statistical independence between the process of randomly selecting the component distribution and the distributions themselves. Each component Y_i is generated using either Fleishman's third-order (`method` = "Fleishman", doi: 10.1007/BF02293811) or Headrick's fifth-order (`method` = "Polynomial", doi: 10.1016/S01679473(02)00072-5) 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`. These components are then transformed to the desired mixture variable using a random multinomial variable generated based on the mixing probabilities. There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with `validpar`. Summaries for the simulation results can be obtained with `summary_var`.

Mixture distributions provide a useful way for describing heterogeneity in a population, especially when an outcome is a composite response from multiple sources. The vignette **Variable Types** provides more information about simulation of mixture variables and the required parameters. The vignette **Expected Cumulants and Correlations for Continuous Mixture Variables** gives the equations for the expected cumulants of a mixture variable. In addition, Headrick & Kowalchuk (2007, doi: 10.1080/10629360600605065) outlined a general method for comparing a simulated distribution Y to a given theoretical distribution Y^*. These steps can be found in the **Continuous Mixture Distributions** vignette.

Usage

```
contmixvar1(n = 10000, method = c("Fleishman", "Polynomial"), means = 0,
  vars = 1, mix_pis = NULL, mix_mus = NULL, mix_sigmas = NULL,
  mix_skews = NULL, mix_skurts = NULL, mix_fifths = NULL,
  mix_sixths = NULL, mix_Six = list(), seed = 1234, cstart = list())
```

Arguments

`n`	the sample size (i.e. the length of the simulated variable; default = 10000)
`method`	the method used to generate the component variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
`means`	mean for the mixture variable (default = 0)
`vars`	variance for the mixture variable (default = 1)
`mix_pis`	a vector of mixing probabilities that sum to 1 for the component distributions
`mix_mus`	a vector of means for the component distributions
`mix_sigmas`	a vector of standard deviations for the component distributions
`mix_skews`	a vector of skew values for the component distributions
`mix_skurts`	a vector of standardized kurtoses for the component distributions
`mix_fifths`	a vector of standardized fifth cumulants for the component distributions; keep NULL if using `method` = "Fleishman" to generate continuous variables
`mix_sixths`	a vector of standardized sixth cumulants for the component distributions; keep NULL if using `method` = "Fleishman" to generate continuous variables
`mix_Six`	a list of vectors of sixth cumulant correction values for the component distributions of Y_{mix}; use `NULL` if no correction is desired for a given component; if no correction is desired for any component keep as `mix_Six = list()` (not necessary for `method` = "Fleishman")
`seed`	the seed value for random number generation (default = 1234)

`cstart` a list of length equal to the total number of mixture components containing initial values for root-solving algorithm used in `find_constants`. If user specified, each list element must be input as a matrix. For `method = "Fleishman"`, each should have 3 columns for c_1, c_2, c_3; for `method = "Polynomial"`, each should have 5 columns for c_1, c_2, c_3, c_4, c_5. If no starting values are specified for a given component, that list element should be `NULL`.

Value

A list with the following components:

`constants` a data.frame of the constants

`Y_comp` a data.frame of the components of the mixture variable

`Y_mix` a data.frame of the generated mixture variable

`sixth_correction` the sixth cumulant correction values for `Y_comp`

`valid.pdf` "TRUE" if constants generate a valid PDF, else "FALSE"

`Time` the total simulation time in minutes

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.

Overview of Simulation Process

1) A check is performed to see if any distributions are repeated within the parameter inputs, i.e. if the mixture variable contains 2 components with the same standardized cumulants. These are noted so that the constants are only calculated once.

2) The constants are calculated for each component variable using `find_constants`. If no solutions are found that generate a valid power method PDF, the function will return constants that produce an invalid PDF (or a stop error if no solutions can be found). Possible solutions include: 1) changing the seed, or 2) using a `mix_Six` list with vectors of sixth cumulant correction values (if `method = "Polynomial"`). Errors regarding constant calculation are the most probable cause of function failure.

3) A matrix `X_cont` of dim `n x length(mix_pis)` of standard normal variables is generated and singular-value decomposition is done to remove any correlation. The `constants` are applied to `X_cont` to create the component variables `Y` with the desired distributions.

4) A random multinomial variable `M = rmultinom(n, size = 1, prob = mix_pis)` is generated using `rmultinom`. The continuous mixture variable `Y_mix` is created from the component variables `Y` based on this multinomial variable. That is, if `M[i, k_i] = 1`, then `Y_mix[i] = Y[i, k_i]`. A location-scale transformation is done on `Y_mix` to give it mean `means` and variance `vars`.

Reasons for Function Errors

1) The most likely cause for function errors 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 component variable to determine if a sixth cumulant correction value is needed. The solutions can be used as starting values (see `cstart` below). 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.

2) The kurtosis may be outside the region of possible values. There is an associated lower boundary for kurtosis 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

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.

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.

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

See Also

`find_constants`, `validpar`, `summary_var`

Examples

```
## Not run:
# Mixture of Beta(6, 3), Beta(4, 1.5), and Beta(10, 20)
Stcum1 <- calc_theory("Beta", c(6, 3))
Stcum2 <- calc_theory("Beta", c(4, 1.5))
Stcum3 <- calc_theory("Beta", c(10, 20))
mix_pis <- c(0.5, 0.2, 0.3)
```

```
mix_mus <- c(Stcum1[1], Stcum2[1], Stcum3[1])
mix_sigmas <- c(Stcum1[2], Stcum2[2], Stcum3[2])
mix_skews <- c(Stcum1[3], Stcum2[3], Stcum3[3])
mix_skurts <- c(Stcum1[4], Stcum2[4], Stcum3[4])
mix_fifths <- c(Stcum1[5], Stcum2[5], Stcum3[5])
mix_sixths <- c(Stcum1[6], Stcum2[6], Stcum3[6])
mix_Six <- list(seq(0.01, 10, 0.01), c(0.01, 0.02, 0.03),
  seq(0.01, 10, 0.01))
Bstcum <- calc_mixmoments(mix_pis, mix_mus, mix_sigmas, mix_skews,
  mix_skurts, mix_fifths, mix_sixths)
Bmix <- contmixvar1(n = 10000, "Polynomial", Bstcum[1], Bstcum[2]^2,
  mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts, mix_fifths,
  mix_sixths, mix_Six)
Bsum <- summary_var(Y_comp = Bmix$Y_comp, Y_mix = Bmix$Y_mix, means = means,
  vars = vars, mix_pis = mix_pis, mix_mus = mix_mus,
  mix_sigmas = mix_sigmas, mix_skews = mix_skews, mix_skurts = mix_skurts,
  mix_fifths = mix_fifths, mix_sixths = mix_sixths)

## End(Not run)
```

`corrvar` *Generation of Correlated Ordinal, Continuous (mixture and non-mixture), and/or Count (Poisson and Negative Binomial, regular and zero-inflated) Variables: Correlation Method 1*

Description

This function simulates `k_cat` ordinal ($r \geq 2$ categories), `k_cont` continuous non-mixture, `k_mix` continuous mixture, `k_pois` Poisson (regular and zero-inflated), and/or `k_nb` Negative Binomial (regular and zero-inflated) variables with a specified correlation matrix `rho`. The variables are generated from multivariate normal variables with intermediate correlation matrix `Sigma`, calculated by `intercorr`, and then transformed. The intermediate correlations involving count variables are determined using **correlation method 1**. The *ordering* of the variables in `rho` must be 1st ordinal, 2nd continuous non-mixture, 3rd components of the continuous mixture, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, and 7th zero-inflated NB. Note that it is possible for `k_cat`, `k_cont`, `k_mix`, `k_pois`, and/or `k_nb` to be 0. The target correlations are specified with respect to the components of the continuous mixture variables. There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with `validpar` and `validcorr`. Summaries for the simulation results can be obtained with `summary_var`.

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. 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`. Continuous mixture variables are generated componentwise and then transformed to the desired mixture variables. Ordinal variables ($r \geq 2$ categories) are generated by

discretizing the standard normal variables at quantiles. These quantiles are determined by evaluating the inverse standard normal cdf at the cumulative probabilities defined by each variable's marginal distribution. Count variables are generated using the inverse cdf method. The CDF of a standard normal variable has a uniform distribution. The appropriate quantile function (F_Y)^(-1) is applied to this uniform variable with the designated parameters to generate the count variable: Y = (F_Y)^(-1)(Phi(Z)). The Negative Binomial variable represents the number of failures which occur in a sequence of Bernoulli trials before the target number of successes is achieved. Zero-inflated Poisson or NB variables are obtained by setting the probability of a structural zero to be greater than 0. The optional error loop attempts to correct the final pairwise correlations to be within a user-specified precision value (`epsilon`) of the target correlations.

The vignette **Variable Types** discusses how each of the different variables are generated and describes the required parameters.

The vignette **Overall Workflow for Generation of Correlated Data** provides a detailed example discussing the step-by-step simulation process and comparing correlation methods 1 and 2.

Usage

```
corrvar(n = 10000, k_cat = 0, k_cont = 0, k_mix = 0, k_pois = 0,
  k_nb = 0, method = c("Fleishman", "Polynomial"), means = NULL,
  vars = NULL, skews = NULL, skurts = NULL, fifths = NULL,
  sixths = NULL, 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 = NULL, p_zip = 0,
  size = NULL, prob = NULL, mu = NULL, p_zinb = 0, rho = NULL,
  seed = 1234, errorloop = FALSE, epsilon = 0.001, maxit = 1000,
  use.nearPD = TRUE, nrand = 1e+05, Sigma = NULL, cstart = list())
```

Arguments

`n`	the sample size (i.e. the length of each simulated variable; default = 10000)
`k_cat`	the number of ordinal (r >= 2 categories) variables (default = 0)
`k_cont`	the number of continuous non-mixture variables (default = 0)
`k_mix`	the number of continuous mixture variables (default = 0)
`k_pois`	the number of regular Poisson and zero-inflated Poisson variables (default = 0)
`k_nb`	the number of regular Negative Binomial and zero-inflated Negative Binomial variables (default = 0)
`method`	the method used to generate the `k_cont` non-mixture and `k_mix` mixture continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
`means`	a vector of means for the `k_cont` non-mixture and `k_mix` mixture continuous variables (i.e. `rep(0, (k_cont + k_mix))`)
`vars`	a vector of variances for the `k_cont` non-mixture and `k_mix` mixture continuous variables (i.e. `rep(1, (k_cont + k_mix))`)
`skews`	a vector of skewness values for the `k_cont` non-mixture continuous variables
`skurts`	a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0) for the `k_cont` non-mixture continuous variables
`fifths`	a vector of standardized fifth cumulants for the `k_cont` non-mixture continuous variables (not necessary for `method` = "Fleishman")

`sixths` a vector of standardized sixth cumulants for the `k_cont` non-mixture continuous variables (not necessary for `method` = "Fleishman")

`Six` a list of vectors of sixth cumulant correction values for the `k_cont` non-mixture continuous variables if no valid PDF constants are found,
ex: `Six = list(seq(0.01, 2, 0.01), seq(1, 10, 0.5))`; if no correction is desired for Y_{cont_i}, set the i-th list component equal to `NULL`; if no correction is desired for any of the Y_{cont} keep as `Six = list()` (not necessary for `method` = "Fleishman")

`mix_pis` a list of length `k_mix` with i-th component a vector of mixing probabilities that sum to 1 for component distributions of Y_{mix_i}

`mix_mus` a list of length `k_mix` with i-th component a vector of means for component distributions of Y_{mix_i}

`mix_sigmas` a list of length `k_mix` with i-th component a vector of standard deviations for component distributions of Y_{mix_i}

`mix_skews` a list of length `k_mix` with i-th component a vector of skew values for component distributions of Y_{mix_i}

`mix_skurts` a list of length `k_mix` with i-th component a vector of standardized kurtoses for component distributions of Y_{mix_i}

`mix_fifths` a list of length `k_mix` with i-th component a vector of standardized fifth cumulants for component distributions of Y_{mix_i} (not necessary for `method` = "Fleishman")

`mix_sixths` a list of length `k_mix` with i-th component a vector of standardized sixth cumulants for component distributions of Y_{mix_i} (not necessary for `method` = "Fleishman")

`mix_Six` a list of length `k_mix` with i-th component a list of vectors of sixth cumulant correction values for component distributions of Y_{mix_i}; use `NULL` if no correction is desired for a given component or mixture variable; if no correction is desired for any of the Y_{mix} keep as `mix_Six = list()` (not necessary for `method` = "Fleishman")

`marginal` a list of length equal to `k_cat`; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; 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, these should be input the same as for ordinal variables with more than 2 categories (i.e. the user-specified probability is the probability of the 1st category, which has the smaller support value)

`support` a list of length equal to `k_cat`; the i-th element is a vector containing the r ordered support values; if not provided (i.e. `support = list()`), the default is for the i-th element to be the vector 1, ..., r

`lam` a vector of lambda (mean > 0) constants for the Poisson variables (see `dpois`); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

`p_zip` a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `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 `dpospois`); if `length(p_zip) < length(lam)`, the missing values are set to 0 (and ordered 1st)

size	a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
prob	a vector of success probability parameters for the NB variables; order the same as in size
mu	a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)
p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)
rho	the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that rho is specified in terms of the components of Y_{mix}
seed	the seed value for random number generation (default = 1234)
errorloop	if TRUE, uses corr_error to attempt to correct final pairwise correlations to be within epsilon of target pairwise correlations (default = FALSE)
epsilon	the maximum acceptable error between the final and target pairwise correlations (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
use.nearPD	TRUE to convert the overall intermediate correlation matrix to the nearest positive definite matrix with Matrix::nearPD if necessary; if FALSE the negative eigenvalues are replaced with 0 if necessary
nrand	the number of random numbers to generate in calculating intermediate correlations with intercorr (default = 10000)
Sigma	an intermediate correlation matrix to use if the user wants to provide one, else it is calculated within by intercorr
cstart	a list of length equal to k_cont + the total number of mixture components containing initial values for root-solving algorithm used in find_constants. If user specified, each list element must be input as a matrix. For method = "Fleishman", each should have 3 columns for c_1, c_2, c_3; for method = "Polynomial", each should have 5 columns for c_1, c_2, c_3, c_4, c_5. If no starting values are specified for a given component, that list element should be NULL.

Value

A list whose components vary based on the type of simulated variables.

If **ordinal variables** are produced: Y_cat the ordinal variables,

If **continuous variables** are produced:

constants a data.frame of the constants,

Y_cont the continuous non-mixture variables,

`Y_comp` the components of the continuous mixture variables,

`Y_mix` the continuous mixture variables,

`sixth_correction` a list of sixth cumulant correction values,

`valid.pdf` a vector where the i-th element is "TRUE" if the constants for the i-th continuous variable generate a valid PDF, else "FALSE"

If **Poisson variables** are produced: `Y_pois` the regular and zero-inflated Poisson variables,

If **Negative Binomial variables** are produced: `Y_nb` the regular and zero-inflated Negative Binomial variables,

Additionally, the following elements:

`Sigma` the intermediate correlation matrix (after the error loop),

`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 for each variable in the error loop,

Overview of Correlation Method 1

The intermediate correlations used in method 1 are more simulation based than those in method 2, which means that accuracy increases with sample size and the number of repetitions. In addition, specifying the seed allows for reproducibility. In addition, method 1 differs from method 2 in the following ways:

1) The intermediate correlation for **count variables** is based on the method of Yahav & Shmueli (2012, doi: 10.1002/asmb.901), which uses a simulation based, logarithmic transformation of the target correlation. This method becomes less accurate as the variable mean gets closer to zero.

2) The **ordinal - count variable** correlations are based on an extension of the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534), in which the correlation correction factor is the product of the upper Frechet-Hoeffding bound on the correlation between the count variable and the normal variable used to generate it and a simulated upper bound on the correlation between an ordinal variable and the normal variable used to generate it (see Demirtas & Hedeker, 2011, doi: 10.1198/tast.2011.10090).

3) The **continuous - count variable** correlations are based on an extension of the methods of Amatya & Demirtas (2015) and Demirtas et al. (2012, doi: 10.1002/sim.5362), in which the correlation correction factor is the product of the upper Frechet-Hoeffding bound on the correlation between the count variable and the normal variable used to generate it and the power method correlation between the continuous variable and the normal variable used to generate it (see Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065). The intermediate correlations are the ratio of the target correlations to the correction factor.

Please see the **Comparison of Correlation Methods 1 and 2** vignette for more information and a step-by-step overview of the simulation process.

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 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. The solutions can be used as starting values (see `cstart` below). 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.

2) The kurtosis may be outside the region of possible values. There is an associated lower boundary for kurtosis 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.

3) The feasibility of the final correlation matrix `rho`, given the distribution parameters, should be checked first using `validcorr`. This function either checks if a given `rho` is plausible or returns the lower and upper final correlation limits. It should be noted that even if a target correlation matrix is within the "plausible range," it still may not be possible to achieve the desired matrix. This happens most frequently when generating ordinal variables or using negative correlations. The error loop frequently fixes these problems.

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.

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.

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

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.

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, validpar, validcorr, intercorr, corr_error, summary_var

Examples

```
## Not run:

# 2 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable
n <- 10000
```

```
seed <- 1234

# Mixture variables: Normal mixture with 2 components;
# mixture of Logistic(0, 1), Chisq(4), Beta(4, 1.5)
# Find cumulants of components of 2nd mixture variable
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))

skews <- skurts <- fifths <- sixths <- NULL
Six <- list()
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
mix_skews <- list(rep(0, 2), c(L[3], C[3], B[3]))
mix_skurts <- list(rep(0, 2), c(L[4], C[4], B[4]))
mix_fifths <- list(rep(0, 2), c(L[5], C[5], B[5]))
mix_sixths <- list(rep(0, 2), c(L[6], C[6], B[6]))
mix_Six <- list(list(NULL, NULL), list(1.75, NULL, 0.03))
Nstcum <- calc_mixmoments(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]],
  mix_skews[[1]], mix_skurts[[1]], mix_fifths[[1]], mix_sixths[[1]])
Mstcum <- calc_mixmoments(mix_pis[[2]], mix_mus[[2]], mix_sigmas[[2]],
  mix_skews[[2]], mix_skurts[[2]], mix_fifths[[2]], mix_sixths[[2]])
means <- c(Nstcum[1], Mstcum[1])
vars <- c(Nstcum[2]^2, Mstcum[2]^2)

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
size <- 2
prob <- 0.75
p_zinb <- 0.2

k_cat <- k_pois <- k_nb <- 1
k_cont <- 0
k_mix <- 2
Rey <- matrix(0.39, 8, 8)
diag(Rey) <- 1
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "M2_1", "M2_2",
  "M2_3", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0
Rey["M2_1", "M2_2"] <- Rey["M2_2", "M2_1"] <- Rey["M2_1", "M2_3"] <- 0
Rey["M2_3", "M2_1"] <- Rey["M2_2", "M2_3"] <- Rey["M2_3", "M2_2"] <- 0

# check parameter inputs
validpar(k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, rho = Rey)

# check to make sure Rey is within the feasible correlation boundaries
validcorr(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", means,
  vars, skews, skurts, fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas,
  mix_skews, mix_skurts, mix_fifths, mix_sixths, mix_Six, marginal,
```

```
  lam, p_zip, size, prob, mu = NULL, p_zinb, Rey, seed)

# simulate without the error loop
Sim1 <- corrvar(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, Rey, seed, epsilon = 0.01)

names(Sim1)

# simulate with the error loop
Sim1_EL <- corrvar(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial",
  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 = NULL, p_zinb, Rey,
  seed, errorloop = TRUE, epsilon = 0.01)

names(Sim1_EL)

## End(Not run)
```

`corrvar2` *Generation of Correlated Ordinal, Continuous (mixture and non-mixture), and/or Count (Poisson and Negative Binomial, regular and zero-inflated) Variables: Correlation Method 2*

Description

This function simulates `k_cat` ordinal ($r \geq 2$ categories), `k_cont` continuous non-mixture, `k_mix` continuous mixture, `k_pois` Poisson (regular and zero-inflated), and/or `k_nb` Negative Binomial (regular and zero-inflated) variables with a specified correlation matrix `rho`. The variables are generated from multivariate normal variables with intermediate correlation matrix `Sigma`, calculated by `intercorr2`, and then transformed. The intermediate correlations involving count variables are determined using **correlation method 2**. The *ordering* of the variables in `rho` must be 1st ordinal, 2nd continuous non-mixture, 3rd components of the continuous mixture, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, and 7th zero-inflated NB. Note that it is possible for `k_cat`, `k_cont`, `k_mix`, `k_pois`, and/or `k_nb` to be 0. The target correlations are specified with respect to the components of the continuous mixture variables. There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with `validpar` and `validcorr2`. Summaries for the simulation results can be obtained with `summary_var`.

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. 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`. Continuous mixture variables are generated componentwise and then transformed to the desired mixture variables. Ordinal variables ($r \geq 2$ categories) are generated by discretizing the standard normal variables at quantiles. These quantiles are determined by evaluating the inverse standard normal cdf at the cumulative probabilities defined by each variable's marginal distribution. Count variables are generated using the inverse cdf method. The CDF of a standard normal variable has a uniform distribution. The appropriate quantile function (F_Y)^(-1) is applied to this uniform variable with the designated parameters to generate the count variable: Y = (F_Y)^(-1)(Phi(Z)). The Negative Binomial variable represents the number of failures which occur in a sequence of Bernoulli trials before the target number of successes is achieved. Zero-inflated Poisson or NB variables are obtained by setting the probability of a structural zero to be greater than 0. The optional error loop attempts to correct the final pairwise correlations to be within a user-specified precision value (`epsilon`) of the target correlations.

The vignette **Variable Types** discusses how each of the different variables are generated and describes the required parameters.

The vignette **Overall Workflow for Generation of Correlated Data** provides a detailed example discussing the step-by-step simulation process and comparing correlation methods 1 and 2.

Usage

```
corrvar2(n = 10000, k_cat = 0, k_cont = 0, k_mix = 0, k_pois = 0,
  k_nb = 0, method = c("Fleishman", "Polynomial"), means = NULL,
  vars = NULL, skews = NULL, skurts = NULL, fifths = NULL,
  sixths = NULL, 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 = NULL, p_zip = 0,
  size = NULL, prob = NULL, mu = NULL, p_zinb = 0, pois_eps = 1e-04,
  nb_eps = 1e-04, rho = NULL, seed = 1234, errorloop = FALSE,
  epsilon = 0.001, maxit = 1000, use.nearPD = TRUE, Sigma = NULL,
  cstart = list())
```

Arguments

`n`	the sample size (i.e. the length of each simulated variable; default = 10000)
`k_cat`	the number of ordinal (r >= 2 categories) variables (default = 0)
`k_cont`	the number of continuous non-mixture variables (default = 0)
`k_mix`	the number of continuous mixture variables (default = 0)
`k_pois`	the number of regular Poisson and zero-inflated Poisson variables (default = 0)
`k_nb`	the number of regular Negative Binomial and zero-inflated Negative Binomial variables (default = 0)
`method`	the method used to generate the `k_cont` non-mixture and `k_mix` mixture continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
`means`	a vector of means for the `k_cont` non-mixture and `k_mix` mixture continuous variables (i.e. `rep(0, (k_cont + k_mix))`)
`vars`	a vector of variances for the `k_cont` non-mixture and `k_mix` mixture continuous variables (i.e. `rep(1, (k_cont + k_mix))`)
`skews`	a vector of skewness values for the `k_cont` non-mixture continuous variables

skurts — a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0) for the k_cont non-mixture continuous variables

fifths — a vector of standardized fifth cumulants for the k_cont non-mixture continuous variables (not necessary for method = "Fleishman")

sixths — a vector of standardized sixth cumulants for the k_cont non-mixture continuous variables (not necessary for method = "Fleishman")

Six — a list of vectors of sixth cumulant correction values for the k_cont non-mixture continuous variables if no valid PDF constants are found,
ex: Six = list(seq(0.01, 2, 0.01), seq(1, 10, 0.5)); if no correction is desired for Y_{cont_i}, set the i-th list component equal to NULL; if no correction is desired for any of the Y_{cont} keep as Six = list() (not necessary for method = "Fleishman")

mix_pis — a list of length k_mix with i-th component a vector of mixing probabilities that sum to 1 for component distributions of Y_{mix_i}

mix_mus — a list of length k_mix with i-th component a vector of means for component distributions of Y_{mix_i}

mix_sigmas — a list of length k_mix with i-th component a vector of standard deviations for component distributions of Y_{mix_i}

mix_skews — a list of length k_mix with i-th component a vector of skew values for component distributions of Y_{mix_i}

mix_skurts — a list of length k_mix with i-th component a vector of standardized kurtoses for component distributions of Y_{mix_i}

mix_fifths — a list of length k_mix with i-th component a vector of standardized fifth cumulants for component distributions of Y_{mix_i} (not necessary for method = "Fleishman")

mix_sixths — a list of length k_mix with i-th component a vector of standardized sixth cumulants for component distributions of Y_{mix_i} (not necessary for method = "Fleishman")

mix_Six — a list of length k_mix with i-th component a list of vectors of sixth cumulant correction values for component distributions of Y_{mix_i}; use NULL if no correction is desired for a given component or mixture variable; if no correction is desired for any of the Y_{mix} keep as mix_Six = list() (not necessary for method = "Fleishman")

marginal — a list of length equal to k_cat; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; 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, these should be input the same as for ordinal variables with more than 2 categories (i.e. the user-specified probability is the probability of the 1st category, which has the smaller support value)

support — a list of length equal to k_cat; the i-th element is a vector containing the r ordered support values; if not provided (i.e. support = list()), the default is for the i-th element to be the vector 1, ..., r

lam — a vector of lambda (mean > 0) constants for the Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

p_zip — a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see 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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)

size — a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables

prob — a vector of success probability parameters for the NB variables; order the same as in size

mu — a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)

p_zinb — a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)

pois_eps — a vector of length k_pois containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable

nb_eps — a vector of length k_nb containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable

rho — the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that rho is specified in terms of the components of Y_{mix}

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

errorloop — if TRUE, uses corr_error to attempt to correct final pairwise correlations to be within epsilon of target pairwise correlations (default = FALSE)

epsilon — the maximum acceptable error between the final and target pairwise correlations (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

use.nearPD — TRUE to convert the overall intermediate correlation matrix to the nearest positive definite matrix with Matrix::nearPD if necessary; if FALSE the negative eigenvalues are replaced with 0 if necessary

Sigma — an intermediate correlation matrix to use if the user wants to provide one, else it is calculated within by intercorr2

cstart — a list of length equal to k_cont + the total number of mixture components containing initial values for root-solving algorithm used in find_constants. If user specified, each list element must be input as a matrix. For method = "Fleishman", each should have 3 columns for c_1, c_2, c_3; for method = "Polynomial", each should have 5 columns for c_1, c_2, c_3, c_4, c_5. If no starting values are specified for a given component, that list element should be NULL.

Value

A list whose components vary based on the type of simulated variables.

If **ordinal variables** are produced: `Y_cat` the ordinal variables,

If **continuous variables** are produced:

`constants` a data.frame of the constants,

`Y_cont` the continuous non-mixture variables,

`Y_comp` the components of the continuous mixture variables,

`Y_mix` the continuous mixture variables,

`sixth_correction` a list of sixth cumulant correction values,

`valid.pdf` a vector where the i-th element is "TRUE" if the constants for the i-th continuous variable generate a valid PDF, else "FALSE"

If **Poisson variables** are produced: `Y_pois` the regular and zero-inflated Poisson variables,

If **Negative Binomial variables** are produced: `Y_nb` the regular and zero-inflated Negative Binomial variables,

Additionally, the following elements:

`Sigma` the intermediate correlation matrix (after the error loop),

`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 for each variable in the error loop,

Overview of Method 2

The intermediate correlations used in method 2 are less simulation based than those in method 1, and no seed is needed. Their calculations involve greater utilization of correction loops which make iterative adjustments until a maximum error has been reached (if possible). In addition, method 2 differs from method 1 in the following ways:

1) The intermediate correlations involving **count variables** are based on the methods of Barbiero & Ferrari (2012, doi: 10.1080/00273171.2012.692630, 2015, doi: 10.1002/asmb.2072). The Poisson or Negative Binomial support is made finite by removing a small user-specified value (i.e. 1e-06) from the total cumulative probability. This truncation factor may differ for each count variable. The count variables are subsequently treated as ordinal and intermediate correlations are calculated using the correction loop of `ord_norm`.

2) The **continuous - count variable** correlations are based on an extension of the method of Demirtas et al. (2012, doi: 10.1002/sim.5362), and the count variables are treated as ordinal. The correction factor is the product of the power method correlation between the continuous variable and the normal variable used to generate it (see Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) and the point-polyserial correlation between the ordinalized count variable and the normal variable used to generate it (see Olsson et al., 1982, doi: 10.1007/BF02294164). The intermediate correlations are the ratio of the target correlations to the correction factor.

Please see the **Comparison of Correlation Methods 1 and 2** vignette for more information and a step-by-step overview of the simulation process.

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 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. The solutions can be used as starting values (see cstart below). 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.

2) The kurtosis may be outside the region of possible values. There is an associated lower boundary for kurtosis 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.

3) The feasibility of the final correlation matrix rho, given the distribution parameters, should be checked first using validcorr2. This function either checks if a given rho is plausible or returns the lower and upper final correlation limits. It should be noted that even if a target correlation matrix is within the "plausible range," it still may not be possible to achieve the desired matrix. This happens most frequently when generating ordinal variables or using negative correlations. The error loop frequently fixes these problems.

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.

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.

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

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.

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, validpar, validcorr2, intercorr2, corr_error, summary_var

Examples

```
## Not run:

# 2 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable
n <- 10000
seed <- 1234
```

```
# Mixture variables: Normal mixture with 2 components;
# mixture of Logistic(0, 1), Chisq(4), Beta(4, 1.5)
# Find cumulants of components of 2nd mixture variable
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))

skews <- skurts <- fifths <- sixths <- NULL
Six <- list()
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
mix_skews <- list(rep(0, 2), c(L[3], C[3], B[3]))
mix_skurts <- list(rep(0, 2), c(L[4], C[4], B[4]))
mix_fifths <- list(rep(0, 2), c(L[5], C[5], B[5]))
mix_sixths <- list(rep(0, 2), c(L[6], C[6], B[6]))
mix_Six <- list(list(NULL, NULL), list(1.75, NULL, 0.03))
Nstcum <- calc_mixmoments(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]],
  mix_skews[[1]], mix_skurts[[1]], mix_fifths[[1]], mix_sixths[[1]])
Mstcum <- calc_mixmoments(mix_pis[[2]], mix_mus[[2]], mix_sigmas[[2]],
  mix_skews[[2]], mix_skurts[[2]], mix_fifths[[2]], mix_sixths[[2]])
means <- c(Nstcum[1], Mstcum[1])
vars <- c(Nstcum[2]^2, Mstcum[2]^2)

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
pois_eps <- 0.0001
size <- 2
prob <- 0.75
p_zinb <- 0.2
nb_eps <- 0.0001

k_cat <- k_pois <- k_nb <- 1
k_cont <- 0
k_mix <- 2
Rey <- matrix(0.39, 8, 8)
diag(Rey) <- 1
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "M2_1", "M2_2",
  "M2_3", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0
Rey["M2_1", "M2_2"] <- Rey["M2_2", "M2_1"] <- Rey["M2_1", "M2_3"] <- 0
Rey["M2_3", "M2_1"] <- Rey["M2_2", "M2_3"] <- Rey["M2_3", "M2_2"] <- 0

# check parameter inputs
validpar(k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, pois_eps, nb_eps, Rey)

# check to make sure Rey is within the feasible correlation boundaries
validcorr2(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", means,
  vars, skews, skurts, fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas,
```

```
  mix_skews, mix_skurts, mix_fifths, mix_sixths, mix_Six, marginal,
  lam, p_zip, size, prob, mu = NULL, p_zinb, pois_eps, nb_eps, Rey, seed)

# simulate without the error loop
Sim2 <- corrvar2(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, pois_eps, nb_eps, Rey, seed,
  epsilon = 0.01)

names(Sim2)

# simulate with the error loop
Sim2_EL <- corrvar2(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial",
  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 = NULL, p_zinb, pois_eps,
  nb_eps, Rey, seed, errorloop = TRUE, epsilon = 0.01)

names(Sim2_EL)

## End(Not run)
```

`corr_error` *Error Loop to Correct Final Correlation of Simulated Variables*

Description

This function attempts to correct the final pairwise correlations of simulated variables to be within `epsilon` of the target correlations. It updates the intermediate normal correlation iteratively in a loop until either the maximum error is less than epsilon or the number of iterations exceeds `maxit`. This function would not ordinarily be called directly by the user. The function is a modification of Barbiero & Ferrari's `ordcont` function in `GenOrd-package`. The `ordcont` function has been modified in the following ways:

1) It works for continuous, ordinal (r >= 2 categories), and count (regular or zero-inflated, Poisson or Negative Binomial) variables.

2) The initial correlation check has been removed because this intermediate correlation Sigma from `corrvar` or `corrvar2` has already been checked for positive-definiteness and used to generate variables.

3) Eigenvalue decomposition is done on `Sigma` to impose the correct interemdiate correlations on the normal variables. If `Sigma` is not positive-definite, the negative eigen values are replaced with 0.

4) The final positive-definite check has been removed.

5) The intermediate correlation update function was changed to accommodate more situations.

6) Allowing specifications for the sample size and the seed for reproducibility.

The vignette **Error Loop Algorithm** describes the algorithm used in the error loop.

Usage

```
corr_error(n = 10000, k_cat = 0, k_cont = 0, k_pois = 0, k_nb = 0,
  method = c("Fleishman", "Polynomial"), means = NULL, vars = NULL,
  constants = NULL, marginal = list(), support = list(), lam = NULL,
  p_zip = 0, size = NULL, mu = NULL, p_zinb = 0, seed = 1234,
  epsilon = 0.001, maxit = 1000, rho0 = NULL, Sigma = NULL,
  rho_calc = NULL)
```

Arguments

n	the sample size
k_cat	the number of ordinal (r >= 2 categories) variables
k_cont	the number of continuous variables (these may be regular continuous variables or components of continuous mixture variables)
k_pois	the number of Poisson (regular or zero-inflated) variables
k_nb	the number of Negative Binomial (regular or zero-inflated) variables
method	the method used to generate the continuous variables. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
means	a vector of means for the continuous variables
vars	a vector of variances for the continuous variables
constants	a matrix with k_cont rows, each a vector of constants c0, c1, c2, c3 (if method = "Fleishman") or c0, c1, c2, c3, c4, c5 (if method = "Polynomial"), like that returned by find_constants
marginal	a list of length equal k_cat; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1)
support	a list of length equal k_cat; the i-th element is a vector of containing the r ordered support values; if not provided, the default is for the i-th element to be the vector 1, ..., r
lam	a vector of lambda (mean > 0) constants for the Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables
p_zip	a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see dzipois)
size	a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
mu	a vector of mean parameters for the NB variables; order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)
p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see dzinegbin)
seed	the seed value for random number generation
epsilon	the maximum acceptable error between the final and target pairwise correlation; smaller epsilons take more time
maxit	the maximum number of iterations to use to find the intermediate correlation; the correction loop stops when either the iteration number passes maxit or epsilon is reached

rho0	the target correlation matrix
Sigma	the intermediate correlation matrix previously used in corrvar or corrvar2
rho_calc	the final correlation matrix calculated in corrvar or corrvar2 before execution of corr_error

Value

A list with the following components:

Sigma the intermediate MVN correlation matrix resulting from the error loop

rho_calc the calculated final correlation matrix generated from Sigma

Y_cat the ordinal variables

Y the continuous (mean 0, variance 1) variables

Y_cont the continuous variables with desired mean and variance

Y_pois the Poisson variables

Y_nb the Negative Binomial variables

niter a matrix containing the number of iterations required for each variable pair

References

Please see references for SimCorrMix.

See Also

corrvar, corrvar2

---

intercorr *Calculate Intermediate MVN Correlation for Ordinal, Continuous, Poisson, or Negative Binomial Variables: Correlation Method 1*

Description

This function calculates a k x k intermediate matrix of correlations, where k = k_cat + k_cont + k_pois + k_nb, to be used in simulating variables with corrvar. The k_cont includes regular continuous variables and components of continuous mixture variables. The ordering of the variables must be ordinal, continuous non-mixture, components of continuous mixture variables, regular Poisson, zero-inflated Poisson, regular Negative Binomial (NB), and zero-inflated NB (note that it is possible for k_cat, k_cont, k_pois, and/or k_nb to be 0). There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with validpar. There is a message given if the calculated intermediate correlation matrix Sigma is not positive-definite because it may not be possible to find a MVN correlation matrix that will produce the desired marginal distributions. This function is called by the simulation function corrvar, and would only be used separately if the user wants to first find the intermediate correlation matrix. This matrix Sigma can be used as an input to corrvar.

Please see the **Comparison of Correlation Methods 1 and 2** vignette for information about calculations by variable pair type and the differences between this function and intercorr2.

Usage

```
intercorr(k_cat = 0, k_cont = 0, k_pois = 0, k_nb = 0,
  method = c("Fleishman", "Polynomial"), constants = NULL,
  marginal = list(), support = list(), lam = NULL, p_zip = 0,
  size = NULL, prob = NULL, mu = NULL, p_zinb = 0, rho = NULL,
  seed = 1234, epsilon = 0.001, maxit = 1000, nrand = 1e+05)
```

Arguments

k_cat	the number of ordinal (r >= 2 categories) variables (default = 0)
k_cont	the number of continuous non-mixture variables and components of continuous mixture variables (default = 0)
k_pois	the number of regular and zero-inflated Poisson variables (default = 0)
k_nb	the number of regular and zero-inflated Negative Binomial variables (default = 0)
method	the method used to generate the k_cont continuous variables. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
constants	a matrix with k_cont rows, each a vector of constants c0, c1, c2, c3 (if method = "Fleishman") or c0, c1, c2, c3, c4, c5 (if method = "Polynomial") like that returned by find_constants
marginal	a list of length equal to k_cat; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1; default = list())
support	a list of length equal to k_cat; the i-th element is a vector of containing the r ordered support values; if not provided (i.e. support = list()), the default is for the i-th element to be the vector 1, ..., r
lam	a vector of lambda (mean > 0) constants for the regular and zero-inflated Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables
p_zip	a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see 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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)
size	a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
prob	a vector of success probability parameters for the NB variables; order the same as in size
mu	a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)

p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)
rho	the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that rho is specified in terms of the components of Y_{mix}
seed	the seed value for random number generation (default = 1234)
epsilon	the maximum acceptable error between the pairwise correlations (default = 0.001) in the calculation of ordinal intermediate correlations with ord_norm
maxit	the maximum number of iterations to use (default = 1000) in the calculation of ordinal intermediate correlations with ord_norm
nrand	the number of random numbers to generate in calculating intermediate correlations (default = 10000)

Value

the intermediate MVN correlation matrix

References

Please see corrvar for references.

See Also

corrvar

Examples

```
## Not run:

# 1 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable
seed <- 1234

# Mixture of N(-2, 1) and N(2, 1)
constants <- rbind(c(0, 1, 0, 0, 0, 0), c(0, 1, 0, 0, 0, 0))

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
size <- 2
prob <- 0.75
p_zinb <- 0.2

k_cat <- k_pois <- k_nb <- 1
k_cont <- 2
Rey <- matrix(0.35, 5, 5)
diag(Rey) <- 1
```

```
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0

Sigma1 <- intercorr(k_cat, k_cont, k_pois, k_nb, "Polynomial", constants,
  marginal, support, lam, p_zip, size, prob, mu = NULL, p_zinb, Rey, seed)
Sigma1

## End(Not run)
```

`intercorr2` *Calculate Intermediate MVN Correlation for Ordinal, Continuous, Poisson, or Negative Binomial Variables: Correlation Method 2*

Description

This function calculates a `k x k` intermediate matrix of correlations, where `k = k_cat + k_cont + k_pois + k_nb`, to be used in simulating variables with `corrvar2`. The `k_cont` includes regular continuous variables and components of continuous mixture variables. The ordering of the variables must be ordinal, continuous non-mixture, components of continuous mixture variables, regular Poisson, zero-inflated Poisson, regular Negative Binomial (NB), and zero-inflated NB (note that it is possible for `k_cat`, `k_cont`, `k_pois`, and/or `k_nb` to be 0). There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with `validpar`. There is a message given if the calculated intermediate correlation matrix `Sigma` is not positive-definite because it may not be possible to find a MVN correlation matrix that will produce the desired marginal distributions. This function is called by the simulation function `corrvar2`, and would only be used separately if the user wants to first find the intermediate correlation matrix. This matrix `Sigma` can be used as an input to `corrvar2`.

Please see the **Comparison of Correlation Methods 1 and 2** vignette for information about calculations by variable pair type and the differences between this function and `intercorr`.

Usage

```
intercorr2(k_cat = 0, k_cont = 0, k_pois = 0, k_nb = 0,
  method = c("Fleishman", "Polynomial"), constants = NULL,
  marginal = list(), support = list(), lam = NULL, p_zip = 0,
  size = NULL, prob = NULL, mu = NULL, p_zinb = 0, pois_eps = 1e-04,
  nb_eps = 1e-04, rho = NULL, epsilon = 0.001, maxit = 1000)
```

Arguments

`k_cat`	the number of ordinal (r >= 2 categories) variables (default = 0)
`k_cont`	the number of continuous non-mixture variables and components of continuous mixture variables (default = 0)
`k_pois`	the number of regular and zero-inflated Poisson variables (default = 0)
`k_nb`	the number of regular and zero-inflated Negative Binomial variables (default = 0)
`method`	the method used to generate the `k_cont` continuous variables. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.

constants — a matrix with k_cont rows, each a vector of constants c0, c1, c2, c3 (if method = "Fleishman") or c0, c1, c2, c3, c4, c5 (if method = "Polynomial") like that returned by find_constants

marginal — a list of length equal to k_cat; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1; default = list())

support — a list of length equal to k_cat; the i-th element is a vector of containing the r ordered support values; if not provided (i.e. support = list()), the default is for the i-th element to be the vector 1, ..., r

lam — a vector of lambda (mean > 0) constants for the regular and zero-inflated Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

p_zip — a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see 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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)

size — a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables

prob — a vector of success probability parameters for the NB variables; order the same as in size

mu — a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)

p_zinb — a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)

pois_eps — a vector of length k_pois containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable

nb_eps — a vector of length k_nb containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable

rho — the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that rho is specified in terms of the components of Y_{mix}

epsilon — the maximum acceptable error between the pairwise correlations (default = 0.001) in the calculation of ordinal intermediate correlations with ord_norm

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

nrand — the number of random numbers to generate in calculating the bound (default = 10000)

Value

the intermediate MVN correlation matrix

References

Please see corrvar2 for references.

See Also

corrvar2

Examples

```
## Not run:

# 1 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable
# The defaults of pois_eps <- nb_eps <- 0.0001 are used.

# Mixture of N(-2, 1) and N(2, 1)
constants <- rbind(c(0, 1, 0, 0, 0, 0), c(0, 1, 0, 0, 0, 0))

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
size <- 2
prob <- 0.75
p_zinb <- 0.2

k_cat <- k_pois <- k_nb <- 1
k_cont <- 2
Rey <- matrix(0.35, 5, 5)
diag(Rey) <- 1
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0

Sigma2 <- intercorr2(k_cat, k_cont, k_pois, k_nb, "Polynomial", constants,
  marginal, support, lam, p_zip, size, prob, mu = NULL, p_zinb, rho = Rey)
Sigma2

## End(Not run)
```

---

intercorr_cat_nb *Calculate Intermediate MVN Correlation for Ordinal - Negative Binomial Variables: Correlation Method 1*

Description

This function calculates the k_cat x k_nb intermediate matrix of correlations for the k_cat ordinal ($r >= 2$ categories) and k_nb Negative Binomial variables required to produce the target correlations in rho_cat_nb. It extends the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534) to ordinal - Negative Binomial pairs and allows for regular or zero-inflated NB variables. Here, the intermediate correlation between Z1 and Z2 (where Z1 is the standard normal variable discretized to produce an ordinal variable Y1, and Z2 is the standard normal variable used to generate a Negative Binomial variable via the inverse CDF method) is calculated by dividing the target correlation by a correction factor. The correction factor is the product of the upper Frechet-Hoeffding bound on the correlation between a Negative Binomial variable and the normal variable used to generate it and a simulated GSC upper bound on the correlation between an ordinal variable and the normal variable used to generate it (see Demirtas & Hedeker, 2011, doi: 10.1198/tast.2011.10090). The function is used in intercorr and corrvar. This function would not ordinarily be called by the user.

Usage

```
intercorr_cat_nb(rho_cat_nb = NULL, marginal = list(), size = NULL,
  mu = NULL, p_zinb = 0, nrand = 1e+05, seed = 1234)
```

Arguments

rho_cat_nb	a k_cat x k_nb matrix of target correlations among ordinal and Negative Binomial variables; the NB variables should be ordered 1st regular, 2nd zero-inflated
marginal	a list of length equal to k_cat; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1)
size	a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
mu	a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)
p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)
nrand	the number of random numbers to generate in calculating the bound (default = 10000)
seed	the seed used in random number generation (default = 1234)

Value

a k_cat x k_nb matrix whose rows represent the k_cat ordinal variables and columns represent the k_nb Negative Binomial variables

References

Please see references for intercorr_cat_pois

See Also

intercorr, corrvar

---

intercorr_cat_pois *Calculate Intermediate MVN Correlation for Ordinal - Poisson Variables: Correlation Method 1*

Description

This function calculates a k_cat x k_pois intermediate matrix of correlations for the k_cat ordinal ($r >= 2$ categories) and k_pois Poisson variables required to produce the target correlations in rho_cat_pois. It extends the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534) to ordinal - Poisson pairs and allows for regular or zero-inflated Poisson variables. Here, the intermediate correlation between Z1 and Z2 (where Z1 is the standard normal variable discretized to produce an ordinal variable Y1, and Z2 is the standard normal variable used to generate a Poisson variable via the inverse CDF method) is calculated by dividing the target correlation by a correction factor. The correction factor is the product of the upper Frechet-Hoeffding bound on the correlation between a Poisson variable and the normal variable used to generate it and a simulated GSC upper bound on the correlation between an ordinal variable and the normal variable used to generate it (see Demirtas & Hedeker, 2011, doi: 10.1198/tast.2011.10090). The function is used in intercorr and corrvar. This function would not ordinarily be called by the user.

Usage

```
intercorr_cat_pois(rho_cat_pois = NULL, marginal = list(), lam = NULL,
  p_zip = 0, nrand = 1e+05, seed = 1234)
```

Arguments

rho_cat_pois	a k_cat x k_pois matrix of target correlations among ordinal and Poisson variables; the Poisson variables should be ordered 1st regular, 2nd zero-inflated
marginal	a list of length equal to k_cat; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1)
lam	a vector of lambda (mean > 0) constants for the regular and zero-inflated Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables
p_zip	a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see 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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)

nrand	the number of random numbers to generate in calculating the bound (default = 10000)
seed	the seed used in random number generation (default = 1234)

Value

a k_cat x k_pois matrix whose rows represent the k_cat ordinal variables and columns represent the k_pois Poisson variables

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.

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.

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

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.

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.

See Also

intercorr, corrvar

---

intercorr_cont *Calculate Intermediate MVN Correlation for Continuous Variables Generated by Polynomial Transformation Method*

Description

This function finds the intermediate correlation for standard normal random variables which are used in Fleishman's third-order (doi: 10.1007/BF02293811) or Headrick's fifth-order (doi: 10.1016/S01679473(02)000725) polynomial transformation method (PMT) using nleqslv. It is used in intercorr and intercorr2 and would not ordinarily be called by the user. The correlations are found pairwise so that eigen-value or principal components decomposition should be done on the resulting Sigma matrix. The **Comparison of Correlation Methods 1 and 2** vignette contains the equations which were derived by Headrick and Sawilowsky (doi: 10.1007/BF02294317) or Headrick (doi: 10.1016/S01679473(02)000725).

Usage

```
intercorr_cont(method = c("Fleishman", "Polynomial"), constants = NULL,
  rho_cont = NULL)
```

Arguments

method	the method used to generate the continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
constants	a matrix with each row a vector of constants c0, c1, c2, c3 (if method = "Fleishman") or c0, c1, c2, c3, c4, c5 (if method = "Polynomial"), like that returned by find_constants
rho_cont	a matrix of target correlations among continuous variables, does not have to be symmetric

Value

the intermediate matrix of correlations with the same dimensions as rho_cont

References

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

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.

See Also

intercorr, intercorr2, nleqslv

---

intercorr_cont_nb *Calculate Intermediate MVN Correlation for Continuous - Negative Binomial Variables: Correlation Method 1*

Description

This function calculates a k_cont x k_nb intermediate matrix of correlations for the k_cont continuous and k_nb Negative Binomial variables. It extends the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534) to continuous variables generated using Headrick's fifth-order polynomial transformation and regular or zero-inflated NB variables. Here, the intermediate correlation between Z1 and Z2 (where Z1 is the standard normal variable transformed using Headrick's fifth-order or Fleishman's third-order method to produce a continuous variable Y1, and Z2 is the standard normal variable used to generate a Negative Binomial variable via the inverse CDF method) is calculated by dividing the target correlation by a correction factor. The correction factor is the product of the upper Frechet-Hoeffding bound on the correlation between a Negative Binomial variable and the normal variable used to generate it and the power method correlation (described in Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) between Y1 and Z1. The function is used in intercorr and corrvar. This function would not ordinarily be called by the user.

Usage

```
intercorr_cont_nb(method = c("Fleishman", "Polynomial"), constants = NULL,
  rho_cont_nb = NULL, size = NULL, mu = NULL, p_zinb = 0,
  nrand = 1e+05, seed = 1234)
```

Arguments

method	the method used to generate the k_cont continuous variables. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
constants	a matrix with k_cont rows, each a vector of constants c0, c1, c2, c3 (if method = "Fleishman") or c0, c1, c2, c3, c4, c5 (if method = "Polynomial"), like that returned by find_constants
rho_cont_nb	a k_cont x k_nb matrix of target correlations among continuous and Negative Binomial variables; the NB variables should be ordered 1st regular, 2nd zero-inflated
size	a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
mu	a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)
p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)
nrand	the number of random numbers to generate in calculating the bound (default = 10000)
seed	the seed used in random number generation (default = 1234)

Value

a k_cont x k_nb matrix whose rows represent the k_cont continuous variables and columns represent the k_nb Negative Binomial variables

References

Please see references for intercorr_cont_pois.

See Also

find_constants, intercorr, corrvar

---

intercorr_cont_nb2 *Calculate Intermediate MVN Correlation for Continuous - Negative Binomial Variables: Correlation Method 2*

Description

This function calculates a k_cont x k_nb intermediate matrix of correlations for the k_cont continuous and k_nb Negative Binomial variables. It extends the methods of Demirtas et al. (2012, doi: 10.1002/sim.5362) and Barbiero & Ferrari (2015, doi: 10.1002/asmb.2072) by:

1) including non-normal continuous and regular or zero-inflated Negative Binomial variables

2) allowing the continuous variables to be generated via Fleishman's third-order or Headrick's fifth-order transformation, and

3) since the count variables are treated as ordinal, using the point-polyserial and polyserial correlations to calculate the intermediate correlations (similar to findintercorr_cont_cat in SimMultiCorrData).

Here, the intermediate correlation between Z1 and Z2 (where Z1 is the standard normal variable transformed using Headrick's fifth-order or Fleishman's third-order method to produce a continuous variable Y1, and Z2 is the standard normal variable used to generate a Negative Binomial variable via the inverse CDF method) is calculated by dividing the target correlation by a correction factor. The correction factor is the product of the point-polyserial correlation between Y2 and Z2 (described in Olsson et al., 1982, doi: 10.1007/BF02294164) and the power method correlation (described in Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) between Y1 and Z1. After the maximum support value has been found using maxcount_support, the point-polyserial correlation is given by:

$$\rho_{Y2,Z2} = \frac{1}{\sigma_{Y2}} \sum_{j=1}^{r-1} \phi(\tau_j)(y2_{j+1} - y2_j)$$

where

$$\phi(\tau) = (2\pi)^{-1/2} * exp(-0.5\tau^2)$$

Here, y_j is the j-th support value and τ_j is $\Phi^{-1}(\sum_{i=1}^{j} Pr(Y = y_i))$. The power method correlation is given by:

$$\rho_{Y1,Z1} = c_1 + 3c_3 + 15c_5,$$

where $c_5 = 0$ if method = "Fleishman". The function is used in intercorr2 and corrvar2. This function would not ordinarily be called by the user.

Usage

```
intercorr_cont_nb2(method = c("Fleishman", "Polynomial"), constants = NULL,
  rho_cont_nb = NULL, nb_marg = list(), nb_support = list())
```

Arguments

method	the method used to generate the k_cont continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
constants	a matrix with k_cont rows, each a vector of constants c0, c1, c2, c3 (if method = "Fleishman") or c0, c1, c2, c3, c4, c5 (if method = "Polynomial"), like that returned by find_constants
rho_cont_nb	a k_cont x k_nb matrix of target correlations among continuous and Negative Binomial variables; the NB variables should be ordered 1st regular, 2nd zero-inflated
nb_marg	a list of length equal to k_nb ordered 1st regular and 2nd zero-inflated; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1); this is created within intercorr2 and corrvar2
nb_support	a list of length equal to k_nb ordered 1st regular and 2nd zero-inflated; the i-th element is a vector of containing the r ordered support values, with a minimum of 0 and maximum determined via maxcount_support

Value

a k_cont x k_nb matrix whose rows represent the k_cont continuous variables and columns represent the k_nb Negative Binomial variables

References

Please see references in intercorr_cont_pois2.

See Also

find_constants, power_norm_corr, intercorr2, corrvar2

---

intercorr_cont_pois *Calculate Intermediate MVN Correlation for Continuous - Poisson Variables: Correlation Method 1*

Description

This function calculates a k_cont x k_pois intermediate matrix of correlations for the k_cont continuous and k_pois Poisson variables. It extends the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534) to continuous variables generated using Headrick's fifth-order polynomial transformation and zero-inflated Poisson variables. Here, the intermediate correlation between Z1 and Z2 (where Z1 is the standard normal variable transformed using Headrick's fifth-order or Fleishman's third-order method to produce a continuous variable Y1, and Z2 is the

standard normal variable used to generate a Poisson variable via the inverse CDF method) is calculated by dividing the target correlation by a correction factor. The correction factor is the product of the upper Frechet-Hoeffding bound on the correlation between a Poisson variable and the normal variable used to generate it and the power method correlation (described in Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) between Y1 and Z1. The function is used in `intercorr` and `corrvar`. This function would not ordinarily be called by the user.

Usage

```
intercorr_cont_pois(method = c("Fleishman", "Polynomial"), constants = NULL,
  rho_cont_pois = NULL, lam = NULL, p_zip = 0, nrand = 1e+05,
  seed = 1234)
```

Arguments

`method`	the method used to generate the `k_cont` continuous variables. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
`constants`	a matrix with `k_cont` rows, each a vector of constants c0, c1, c2, c3 (if `method` = "Fleishman") or c0, c1, c2, c3, c4, c5 (if `method` = "Polynomial"), like that returned by `find_constants`
`rho_cont_pois`	a `k_cont x k_pois` matrix of target correlations among continuous and Poisson variables; the Poisson variables should be ordered 1st regular, 2nd zero-inflated
`lam`	a vector of lambda (mean > 0) constants for the regular and zero-inflated Poisson variables (see `dpois`); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables
`p_zip`	a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `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 `dpospois`); if `length(p_zip) < length(lam)`, the missing values are set to 0 (and ordered 1st)
`nrand`	the number of random numbers to generate in calculating the bound (default = 10000)
`seed`	the seed used in random number generation (default = 1234)

Value

a `k_cont x k_pois` matrix whose rows represent the `k_cont` continuous variables and columns represent the `k_pois` Poisson variables

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.

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.

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

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.

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.

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.

See Also

power_norm_corr, find_constants, intercorr, corrvar

`intercorr_cont_pois2` *Calculate Intermediate MVN Correlation for Continuous - Poisson Variables: Correlation Method 2*

Description

This function calculates a `k_cont x k_pois` intermediate matrix of correlations for the `k_cont` continuous and `k_pois` Poisson variables. It extends the methods of Demirtas et al. (2012, doi: 10.1002/sim.5362) and Barbiero & Ferrari (2015, doi: 10.1002/asmb.2072) by:

1) including non-normal continuous and regular or zero-inflated Poisson variables

2) allowing the continuous variables to be generated via Fleishman's third-order or Headrick's fifth-order transformation, and

3) since the count variables are treated as ordinal, using the point-polyserial and polyserial correlations to calculate the intermediate correlations (similar to findintercorr_cont_cat) in SimMultiCorrData).

Here, the intermediate correlation between Z1 and Z2 (where Z1 is the standard normal variable transformed using Headrick's fifth-order or Fleishman's third-order method to produce a continuous variable Y1, and Z2 is the standard normal variable used to generate a Poisson variable via the inverse CDF method) is calculated by dividing the target correlation by a correction factor. The correction factor is the product of the point-polyserial correlation between Y2 and Z2 (described in Olsson et al., 1982, doi: 10.1007/BF02294164) and the power method correlation (described in Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) between Y1 and Z1. After the maximum support value has been found using maxcount_support, the point-polyserial correlation is given by:

$$\rho_{Y2,Z2} = \frac{1}{\sigma_{Y2}} \sum_{j=1}^{r-1} \phi(\tau_j)(y2_{j+1} - y2_j)$$

where

$$\phi(\tau) = (2\pi)^{-1/2} * exp(-0.5\tau^2)$$

Here, y_j is the j-th support value and τ_j is $\Phi^{-1}(\sum_{i=1}^{j} Pr(Y = y_i))$. The power method correlation is given by:

$$\rho_{Y1,Z1} = c_1 + 3c_3 + 15c_5,$$

where $c_5 = 0$ if method = "Fleishman". The function is used in intercorr2 and corrvar2. This function would not ordinarily be called by the user.

Usage

```
intercorr_cont_pois2(method = c("Fleishman", "Polynomial"),
  constants = NULL, rho_cont_pois = NULL, pois_marg = list(),
  pois_support = list())
```

Arguments

method	the method used to generate the k_cont continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
constants	a matrix with k_cont rows, each a vector of constants c0, c1, c2, c3 (if method = "Fleishman") or c0, c1, c2, c3, c4, c5 (if method = "Polynomial"), like that returned by find_constants
rho_cont_pois	a k_cont x k_pois matrix of target correlations among continuous and Poisson variables; the Poisson variables should be ordered 1st regular, 2nd zero-inflated
pois_marg	a list of length equal to k_pois ordered 1st regular and 2nd zero-inflated; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1); this is created within intercorr2 and corrvar2
pois_support	a list of length equal to k_pois ordered 1st regular and 2nd zero-inflated; the i-th element is a vector of containing the r ordered support values, with a minimum of 0 and maximum determined via maxcount_support

Value

a k_cont x k_pois matrix whose rows represent the k_cont continuous variables and columns represent the k_pois Poisson variables

References

Please see additional references in intercorr_cont_pois.

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.

See Also

find_constants, power_norm_corr, intercorr2, corrvar2

`intercorr_nb` *Calculate Intermediate MVN Correlation for Negative Binomial Variables: Correlation Method 1*

Description

This function calculates a `k_nb x k_nb` intermediate matrix of correlations for the Negative Binomial variables by extending the method of Yahav & Shmueli (2012, doi: 10.1002/asmb.901). The intermediate correlation between Z1 and Z2 (the standard normal variables used to generate the Negative Binomial variables Y1 and Y2 via the inverse CDF method) is calculated using a logarithmic transformation of the target correlation. First, the upper and lower Frechet-Hoeffding bounds (mincor, maxcor) on $\rho_{Y1,Y2}$ are simulated. Then the intermediate correlation is found as follows:

$$\rho_{Z1,Z2} = \frac{1}{b} * log(\frac{\rho_{Y1,Y2} - c}{a}),$$

where $a = -(maxcor * mincor)/(maxcor + mincor)$, $b = log((maxcor + a)/a)$, and $c = -a$. The function adapts code from Amatya & Demirtas' (2016) package `PoisNor-package` by:

1) allowing specifications for the number of random variates and the seed for reproducibility

2) providing the following checks: if `Sigma_(Z1, Z2) > 1`, `Sigma_(Z1, Z2)` is set to 1; if `Sigma_(Z1, Z2) < -1`, `Sigma_(Z1, Z2)` is set to -1

3) simulating regular and zero-inflated Negative Binomial variables.

The function is used in `intercorr` and `corrvar` and would not ordinarily be called by the user.

Usage

```
intercorr_nb(rho_nb = NULL, size = NULL, mu = NULL, p_zinb = 0,
  nrand = 1e+05, seed = 1234)
```

Arguments

`rho_nb`	a `k_nb x k_nb` matrix of target correlations ordered 1st regular and 2nd zero-inflated
`size`	a vector of size parameters for the Negative Binomial variables (see `dnbinom`); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
`mu`	a vector of mean parameters for the NB variables (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in `size`; for zero-inflated NB this refers to the mean of the NB distribution (see `dzinegbin`)
`p_zinb`	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see `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 `dposnegbin`); if `length(p_zinb) < length(size)`, the missing values are set to 0 (and ordered 1st)
`nrand`	the number of random numbers to generate in calculating the bound (default = 10000)
`seed`	the seed used in random number generation (default = 1234)

Value

the k_nb x k_nb intermediate correlation matrix for the Negative Binomial variables

References

Please see references for intercorr_pois.

See Also

intercorr_pois, intercorr_pois_nb, intercorr, corrvar

---

intercorr_pois *Calculate Intermediate MVN Correlation for Poisson Variables: Correlation Method 1*

Description

This function calculates a k_pois x k_pois intermediate matrix of correlations for the Poisson variables using the method of Yahav & Shmueli (2012, doi: 10.1002/asmb.901). The intermediate correlation between Z1 and Z2 (the standard normal variables used to generate the Poisson variables Y1 and Y2 via the inverse CDF method) is calculated using a logarithmic transformation of the target correlation. First, the upper and lower Frechet-Hoeffding bounds (mincor, maxcor) on $\rho_{Y1,Y2}$ are simulated. Then the intermediate correlation is found as follows:

$$\rho_{Z1,Z2} = \frac{1}{b} * log(\frac{\rho_{Y1,Y2} - c}{a}),$$

where $a = -(maxcor * mincor)/(maxcor + mincor)$, $b = log((maxcor + a)/a)$, and $c = -a$. The function adapts code from Amatya & Demirtas' (2016) package PoisNor-package by:

1) allowing specifications for the number of random variates and the seed for reproducibility

2) providing the following checks: if Sigma_(Z1, Z2) > 1, Sigma_(Z1, Z2) is set to 1; if Sigma_(Z1, Z2) < -1, Sigma_(Z1, Z2) is set to -1

3) simulating regular and zero-inflated Poisson variables.

The function is used in intercorr and corrvar and would not ordinarily be called by the user.

Usage

```
intercorr_pois(rho_pois = NULL, lam = NULL, p_zip = 0, nrand = 1e+05,
  seed = 1234)
```

Arguments

rho_pois	a k_pois x k_pois matrix of target correlations ordered 1st regular and 2nd zero-inflated
lam	a vector of lambda (mean > 0) constants for the regular and zero-inflated Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

p_zip	a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see 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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)
nrand	the number of random numbers to generate in calculating the bound (default = 10000)
seed	the seed used in random number generation (default = 1234)

Value

the k_pois x k_pois intermediate correlation matrix for the Poisson variables

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.

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

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

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.

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.

See Also

intercorr_nb, intercorr_pois_nb, intercorr, corrvar

---

intercorr_pois_nb *Calculate Intermediate MVN Correlation for Poisson - Negative Binomial Variables: Correlation Method 1*

Description

This function calculates a k_pois x k_nb intermediate matrix of correlations for the Poisson and Negative Binomial variables by extending the method of Yahav & Shmueli (2012, doi: 10.1002/asmb.901). The intermediate correlation between Z1 and Z2 (the standard normal variables used to generate the Poisson and Negative Binomial variables Y1 and Y2 via the inverse CDF method) is calculated using a logarithmic transformation of the target correlation. First, the upper and lower

Frechet-Hoeffding bounds (mincor, maxcor) on $\rho_{Y1,Y2}$ are simulated. Then the intermediate correlation is found as follows:

$$\rho_{Z1,Z2} = \frac{1}{b} * log(\frac{\rho_{Y1,Y2} - c}{a}),$$

where $a = -(maxcor * mincor)/(maxcor + mincor)$, $b = log((maxcor + a)/a)$, and $c = -a$. The function adapts code from Amatya & Demirtas' (2016) package `PoisNor-package` by:

1) allowing specifications for the number of random variates and the seed for reproducibility

2) providing the following checks: if `Sigma_(Z1, Z2) > 1`, `Sigma_(Z1, Z2)` is set to 1; if `Sigma_(Z1, Z2) < -1`, `Sigma_(Z1, Z2)` is set to -1

3) simulating regular and zero-inflated Poisson and Negative Binomial variables.

The function is used in `intercorr` and `corrvar` and would not ordinarily be called by the user.

Usage

```
intercorr_pois_nb(rho_pois_nb = NULL, lam = NULL, p_zip = 0,
  size = NULL, mu = NULL, p_zinb = 0, nrand = 1e+05, seed = 1234)
```

Arguments

`rho_pois_nb`	a `k_pois x k_nb` matrix of target correlations; order of each type should be 1st regular, 2nd zero-inflated
`lam`	a vector of lambda (mean > 0) constants for the regular and zero-inflated Poisson variables (see `dpois`); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables
`p_zip`	a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `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 `dpospois`); if `length(p_zip) < length(lam)`, the missing values are set to 0 (and ordered 1st)
`size`	a vector of size parameters for the Negative Binomial variables (see `dnbinom`); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
`mu`	a vector of mean parameters for the NB variables (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in `size`; for zero-inflated NB this refers to the mean of the NB distribution (see `dzinegbin`)
`p_zinb`	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see `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 `dposnegbin`); if `length(p_zinb) < length(size)`, the missing values are set to 0 (and ordered 1st)
`nrand`	the number of random numbers to generate in calculating the bound (default = 10000)
`seed`	the seed used in random number generation (default = 1234)

Value

the k_pois x k_nb intermediate correlation matrix whose rows represent the k_pois Poisson variables and columns represent the k_nb Negative Binomial variables

References

Please see references for intercorr_pois.

See Also

intercorr_pois, intercorr_nb, intercorr, corrvar

---

maxcount_support *Calculate Maximum Support Value for Count Variables: Correlation Method 2*

Description

This function calculates the maximum support value for count variables by extending the method of Barbiero & Ferrari (2015, doi: 10.1002/asmb.2072) to include regular and zero-inflated Poisson and Negative Binomial variables. In order for count variables to be treated as ordinal in the calculation of the intermediate MVN correlation matrix, their infinite support must be truncated (made finite). This is done by setting the total cumulative probability equal to 1 - a small user-specified value (pois_eps or nb_eps). The maximum support value equals the inverse CDF applied to this result. The truncation values may differ for each variable. The function is used in intercorr2 and corrvar2 and would not ordinarily be called by the user.

Usage

```
maxcount_support(k_pois = 0, k_nb = 0, lam = NULL, p_zip = 0,
  size = NULL, prob = NULL, mu = NULL, p_zinb = 0, pois_eps = NULL,
  nb_eps = NULL)
```

Arguments

k_pois	the number of Poisson variables
k_nb	the number of Negative Binomial variables
lam	a vector of lambda (mean > 0) constants for the regular and zero-inflated Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables
p_zip	a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see 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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)
size	a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables

prob	a vector of success probability parameters for the NB variables; order the same as in size
mu	a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)
p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)
pois_eps	a vector of length k_pois containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable
nb_eps	a vector of length k_nb containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable

Value

a data.frame with k_pois + k_nb rows; the column names are:

Distribution Poisson or Negative Binomial

Number the variable index

Max the maximum support value

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.

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

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

See Also

intercorr2, corrvar2

---

norm_ord *Calculate Correlations of Ordinal Variables Obtained from Discretizing Normal Variables*

Description

This function calculates the correlation of ordinal variables (or variables treated as "ordinal"), with given marginal distributions, obtained from discretizing standard normal variables with a specified correlation matrix. The function modifies Barbiero & Ferrari's contord function in GenOrd-package. It uses pmvnorm function from the **mvtnorm** package to calculate multivariate normal cumulative probabilities defined by the normal quantiles obtained at marginal and the supplied correlation matrix Sigma. This function is used within ord_norm and would not ordinarily be called by the user.

Usage

```
norm_ord(marginal = list(), Sigma = NULL, support = list(),
  Spearman = FALSE)
```

Arguments

marginal	a list of length equal to the number of variables; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1)
Sigma	the correlation matrix of the multivariate standard normal variable
support	a list of length equal to the number of variables; the i-th element is a vector of containing the r ordered support values; if not provided (i.e. support = list()), the default is for the i-th element to be the vector 1, ..., r
Spearman	if TRUE, Spearman's correlations are used (and support is not required); if FALSE (default) Pearson's correlations are used

Value

the correlation matrix of the ordinal variables

References

Please see references in ord_norm.

Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T (2017). mvtnorm: Multivariate Normal and t Distributions. R package version 1.0-6. http://CRAN.R-project.org/package=mvtnorm

Genz A, Bretz F (2009), Computation of Multivariate Normal and t Probabilities. Lecture Notes in Statistics, Vol. 195., Springer-Verlag, Heidelberg. ISBN 978-3-642-01688-2

See Also

ord_norm

---

ord_norm *Calculate Intermediate MVN Correlation to Generate Variables Treated as Ordinal*

Description

This function calculates the intermediate MVN correlation needed to generate a variable described by a discrete marginal distribution and associated finite support. This includes ordinal ($r \geq 2$ categories) variables or variables that are treated as ordinal (i.e. count variables in the Barbiero & Ferrari, 2015 method used in corrvar2, doi: 10.1002/asmb.2072). The function is a modification of Barbiero & Ferrari's ordcont function in GenOrd-package. It works by setting the intermediate MVN correlation equal to the target correlation and updating each intermediate pairwise correlation until the final pairwise correlation is within epsilon of the target correlation or the maximum number of iterations has been reached. This function uses norm_ord to calculate the ordinal correlation obtained from discretizing the normal variables generated from the intermediate correlation matrix. The ordcont has been modified in the following ways:

1) the initial correlation check has been removed because this is done within the simulation functions

2) the final positive-definite check has been removed

3) the intermediate correlation update function was changed to accomodate more situations

This function would not ordinarily be called by the user. Note that this will return a matrix that is NOT positive-definite because this is corrected for in the simulation functions `corrvar` and `corrvar2` using the method of Higham (2002) and the `nearPD` function.

Usage

```
ord_norm(marginal = list(), rho = NULL, support = list(),
  epsilon = 0.001, maxit = 1000, Spearman = FALSE)
```

Arguments

`marginal`	a list of length equal to the number of variables; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1)
`rho`	the target correlation matrix
`support`	a list of length equal to the number of variables; the i-th element is a vector of containing the r ordered support values; if not provided (i.e. support = list()), the default is for the i-th element to be the vector 1, ..., r
`epsilon`	the maximum acceptable error between the final and target pairwise correlations (default = 0.001); smaller values take more time
`maxit`	the maximum number of iterations to use (default = 1000) to find the intermediate correlation; the correction loop stops when either the iteration number passes `maxit` or `epsilon` is reached
`Spearman`	if TRUE, Spearman's correlations are used (and support is not required); if FALSE (default) Pearson's correlations are used

Value

A list with the following components:

`SigmaC` the intermediate MVN correlation matrix

`rho0` the calculated final correlation matrix generated from `SigmaC`

`rho` the target final correlation matrix

`niter` a matrix containing the number of iterations required for each variable pair

`maxerr` the maximum final error between the final and target correlation matrices

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

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

See Also

corrvar, corrvar2, norm_ord, intercorr, intercorr2

`plot_simpdf_theory` *Plot Simulated Probability Density Function and Target PDF by Distribution Name or Function for Continuous or Count Variables*

Description

This plots the PDF of simulated continuous or count (regular or zero-inflated, Poisson or Negative Binomial) data and overlays the target PDF (if `overlay` = TRUE), which is specified by distribution name (plus up to 4 parameters) or PDF function `fx` (plus support bounds). If a continuous target distribution is provided (`cont_var = TRUE`), the simulated data y is scaled and then transformed (i.e. $y = sigma * scale(y) + mu$) so that it has the same mean (mu) and variance ($sigma^2$) as the target distribution. The PDF's of continuous variables are shown as lines (using `geom_density` and `geom_line`). It works for valid or invalid power method PDF's. The PMF's of count variables are shown as vertical bar graphs (using `geom_col`). The function returns a `ggplot2-package` object so the user can save it or modify it as necessary. The graph parameters (i.e. `title`, `power_color`, `target_color`, `target_lty`, `legend.position`, `legend.justification`, `legend.text.size`, `title.text.size`, `axis.text.size`, and `axis.title.size`) are inputs to the `ggplot2-package` functions so information about valid inputs can be obtained from that package's documentation.

Usage

```
plot_simpdf_theory(sim_y, title = "Simulated Probability Density Function",
  ylower = NULL, yupper = NULL, power_color = "dark blue",
  overlay = TRUE, cont_var = TRUE, target_color = "dark green",
  target_lty = 2, Dist = c("Benini", "Beta", "Beta-Normal",
  "Birnbaum-Saunders", "Chisq", "Dagum", "Exponential", "Exp-Geometric",
  "Exp-Logarithmic", "Exp-Poisson", "F", "Fisk", "Frechet", "Gamma", "Gaussian",
  "Gompertz", "Gumbel", "Kumaraswamy", "Laplace", "Lindley", "Logistic",
  "Loggamma", "Lognormal", "Lomax", "Makeham", "Maxwell", "Nakagami",
  "Paralogistic", "Pareto", "Perks", "Rayleigh", "Rice", "Singh-Maddala",
  "Skewnormal", "t", "Topp-Leone", "Triangular", "Uniform", "Weibull",
  "Poisson", "Negative_Binomial"), params = NULL, fx = NULL, lower = NULL,
  upper = NULL, legend.position = c(0.975, 0.9),
  legend.justification = c(1, 1), legend.text.size = 10,
  title.text.size = 15, axis.text.size = 10, axis.title.size = 13)
```

Arguments

`sim_y`	a vector of simulated data
`title`	the title for the graph (default = "Simulated Probability Density Function")
`ylower`	the lower y value to use in the plot (default = NULL, uses minimum simulated y value) on the x-axis
`yupper`	the upper y value (default = NULL, uses maximum simulated y value) on the x-axis
`power_color`	the line color for the simulated variable (or column fill color in the case of `Dist` = "Poisson" or "Negative_Binomial")

overlay	if TRUE (default), the target distribution is also plotted given either a distribution name (and parameters) or PDF function fx (with bounds = ylower, yupper)
cont_var	TRUE (default) for continuous variables, FALSE for count variables
target_color	the line color for the target PDF (or column fill color in the case of Dist = "Poisson" or "Negative_Binomial")
target_lty	the line type for the target PDF (default = 2, dashed line)
Dist	name of the distribution. The possible values are: "Benini", "Beta", "Beta-Normal", "Birnbaum-Saunders", "Chisq", "Exponential", "Exp-Geometric", "Exp-Logarithmic", "Exp-Poisson", "F", "Fisk", "Frechet", "Gamma", "Gaussian", "Gompertz", "Gumbel", "Kumaraswamy", "Laplace", "Lindley", "Logistic", "Loggamma", "Lognormal", "Lomax", "Makeham", "Maxwell", "Nakagami", "Paralogistic", "Pareto", "Perks", "Rayleigh", "Rice", "Singh-Maddala", "Skewnormal", "t", "Topp-Leone", "Triangular", "Uniform", "Weibull", "Poisson", and "Negative_Binomial". Please refer to the documentation for each package (either stats-package, VGAM-package, or triangle) for information on appropriate parameter inputs.
params	a vector of parameters (up to 4) for the desired distribution (keep NULL if fx supplied instead); for Poisson variables, must be lambda (mean) and the probability of a structural zero (use 0 for regular Poisson variables); for Negative Binomial variables, must be size, mean and the probability of a structural zero (use 0 for regular NB variables)
fx	a PDF input as a function of x only, i.e. fx = function(x) 0.5 * (x - 1)^2; must return a scalar (keep NULL if Dist supplied instead)
lower	the lower support bound for fx
upper	the upper support bound for fx
legend.position	the position of the legend
legend.justification	the justification of the legend
legend.text.size	the size of the legend labels
title.text.size	the size of the plot title
axis.text.size	the size of the axes text (tick labels)
axis.title.size	the size of the axes titles

Value

A ggplot2-package object.

References

Please see the references for plot_simtheory.

Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.

See Also

calc_theory, ggplot, geom_line, geom_density, geom_col

Examples

```
## Not run:
# Mixture of Beta(6, 3), Beta(4, 1.5), and Beta(10, 20)
Stcum1 <- calc_theory("Beta", c(6, 3))
Stcum2 <- calc_theory("Beta", c(4, 1.5))
Stcum3 <- calc_theory("Beta", c(10, 20))
mix_pis <- c(0.5, 0.2, 0.3)
mix_mus <- c(Stcum1[1], Stcum2[1], Stcum3[1])
mix_sigmas <- c(Stcum1[2], Stcum2[2], Stcum3[2])
mix_skews <- c(Stcum1[3], Stcum2[3], Stcum3[3])
mix_skurts <- c(Stcum1[4], Stcum2[4], Stcum3[4])
mix_fifths <- c(Stcum1[5], Stcum2[5], Stcum3[5])
mix_sixths <- c(Stcum1[6], Stcum2[6], Stcum3[6])
mix_Six <- list(seq(0.01, 10, 0.01), c(0.01, 0.02, 0.03),
  seq(0.01, 10, 0.01))
Bstcum <- calc_mixmoments(mix_pis, mix_mus, mix_sigmas, mix_skews,
  mix_skurts, mix_fifths, mix_sixths)
Bmix <- contmixvar1(n = 10000, "Polynomial", Bstcum[1], Bstcum[2]^2,
  mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts, mix_fifths,
  mix_sixths, mix_Six)
plot_simpdf_theory(Bmix$Y_mix[, 1], title = "Mixture of Beta Distributions",
  fx = function(x) mix_pis[1] * dbeta(x, 6, 3) + mix_pis[2] *
    dbeta(x, 4, 1.5) + mix_pis[3] * dbeta(x, 10, 20), lower = 0, upper = 1)

## End(Not run)
```

`plot_simtheory` *Plot Simulated Data and Target Distribution Data by Name or Function for Continuous or Count Variables*

Description

This plots simulated continuous or count (regular or zero-inflated, Poisson or Negative Binomial) data and overlays data (if `overlay` = TRUE) generated from the target distribution. The target is specified by name (plus up to 4 parameters) or PDF function `fx` (plus support bounds). Due to the integration involved in finding the CDF from the PDF supplied by `fx`, only continuous `fx` may be supplied. Both are plotted as histograms (using `geom_histogram`). If a continuous target distribution is specified (`cont_var = TRUE`), the simulated data y is scaled and then transformed (i.e. $y = sigma * scale(y) + mu$) so that it has the same mean (mu) and variance ($sigma^2$) as the target distribution. It works for valid or invalid power method PDF's. It returns a `ggplot2-package` object so the user can save it or modify it as necessary. The graph parameters (i.e. `title`, `power_color`, `target_color`, `target_lty`, `legend.position`, `legend.justification`, `legend.text.size`, `title.text.size`, `axis.text.size`, and `axis.title.size`) are inputs to the `ggplot2-package` functions so information about valid inputs can be obtained from that package's documentation.

Usage

```
plot_simtheory(sim_y, title = "Simulated Data Values", ylower = NULL,
  yupper = NULL, power_color = "dark blue", overlay = TRUE,
  cont_var = TRUE, target_color = "dark green", nbins = 100,
  Dist = c("Benini", "Beta", "Beta-Normal", "Birnbaum-Saunders", "Chisq",
```

```
"Dagum", "Exponential", "Exp-Geometric", "Exp-Logarithmic", "Exp-Poisson",
"F", "Fisk", "Frechet", "Gamma", "Gaussian", "Gompertz", "Gumbel",
"Kumaraswamy", "Laplace", "Lindley", "Logistic", "Loggamma", "Lognormal",
"Lomax", "Makeham", "Maxwell", "Nakagami", "Paralogistic", "Pareto", "Perks",
"Rayleigh", "Rice", "Singh-Maddala", "Skewnormal", "t", "Topp-Leone",
"Triangular", "Uniform", "Weibull", "Poisson", "Negative_Binomial"),
params = NULL, fx = NULL, lower = NULL, upper = NULL, seed = 1234,
sub = 1000, legend.position = c(0.975, 0.9), legend.justification = c(1,
1), legend.text.size = 10, title.text.size = 15, axis.text.size = 10,
axis.title.size = 13)
```

Arguments

sim_y	a vector of simulated data
title	the title for the graph (default = "Simulated Data Values")
ylower	the lower y value to use in the plot (default = NULL, uses minimum simulated y value) on the y-axis
yupper	the upper y value (default = NULL, uses maximum simulated y value) on the y-axis
power_color	the histogram fill color for the simulated variable (default = "dark blue")
overlay	if TRUE (default), the target distribution is also plotted given either a distribution name (and parameters) or PDF function fx (with support bounds = lower, upper)
cont_var	TRUE (default) for continuous variables, FALSE for count variables
target_color	the histogram fill color for the target distribution (default = "dark green")
nbins	the number of bins to use when creating the histograms (default = 100)
Dist	name of the distribution. The possible values are: "Benini", "Beta", "Beta-Normal", "Birnbaum-Saunders", "Chisq", "Exponential", "Exp-Geometric", "Exp-Logarithmic", "Exp-Poisson", "F", "Fisk", "Frechet", "Gamma", "Gaussian", "Gompertz", "Gumbel", "Kumaraswamy", "Laplace", "Lindley", "Logistic", "Loggamma", "Lognormal", "Lomax", "Makeham", "Maxwell", "Nakagami", "Paralogistic", "Pareto", "Perks", "Rayleigh", "Rice", "Singh-Maddala", "Skewnormal", "t", "Topp-Leone", "Triangular", "Uniform", "Weibull", "Poisson", and "Negative_Binomial". Please refer to the documentation for each package (either stats-package, VGAM-package, or triangle) for information on appropriate parameter inputs.
params	a vector of parameters (up to 4) for the desired distribution (keep NULL if fx supplied instead); for Poisson variables, must be lambda (mean) and the probability of a structural zero (use 0 for regular Poisson variables); for Negative Binomial variables, must be size, mean and the probability of a structural zero (use 0 for regular NB variables)
fx	a PDF input as a function of x only, i.e. fx = function(x) 0.5 * (x - 1)^2; must return a scalar (keep NULL if Dist supplied instead)
lower	the lower support bound for a supplied fx, else keep NULL (note: if an error is thrown from uniroot, try a slightly higher lower bound; i.e., 0.0001 instead of 0)
upper	the upper support bound for a supplied fx, else keep NULL (note: if an error is thrown from uniroot, try a lower upper bound; i.e., 100000 instead of Inf)
seed	the seed value for random number generation (default = 1234)

sub — the number of subdivisions to use in the integration to calculate the CDF from fx; if no result, try increasing sub (requires longer computation time; default = 1000)

legend.position — the position of the legend

legend.justification — the justification of the legend

legend.text.size — the size of the legend labels

title.text.size — the size of the plot title

axis.text.size — the size of the axes text (tick labels)

axis.title.size — the size of the axes titles

Value

A ggplot2-package object.

References

Carnell R (2017). triangle: Provides the Standard Distribution Functions for the Triangle Distribution. R package version 0.11. https://CRAN.R-project.org/package=triangle.

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.

Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.

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

See Also

calc_theory, ggplot, geom_histogram

Examples

```
## Not run:
# Mixture of Beta(6, 3), Beta(4, 1.5), and Beta(10, 20)
Stcum1 <- calc_theory("Beta", c(6, 3))
Stcum2 <- calc_theory("Beta", c(4, 1.5))
Stcum3 <- calc_theory("Beta", c(10, 20))
mix_pis <- c(0.5, 0.2, 0.3)
mix_mus <- c(Stcum1[1], Stcum2[1], Stcum3[1])
mix_sigmas <- c(Stcum1[2], Stcum2[2], Stcum3[2])
mix_skews <- c(Stcum1[3], Stcum2[3], Stcum3[3])
mix_skurts <- c(Stcum1[4], Stcum2[4], Stcum3[4])
mix_fifths <- c(Stcum1[5], Stcum2[5], Stcum3[5])
mix_sixths <- c(Stcum1[6], Stcum2[6], Stcum3[6])
mix_Six <- list(seq(0.01, 10, 0.01), c(0.01, 0.02, 0.03),
  seq(0.01, 10, 0.01))
Bstcum <- calc_mixmoments(mix_pis, mix_mus, mix_sigmas, mix_skews,
  mix_skurts, mix_fifths, mix_sixths)
```

```
Bmix <- contmixvar1(n = 10000, "Polynomial", Bstcum[1], Bstcum[2]^2,
  mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts, mix_fifths,
  mix_sixths, mix_Six)
plot_simtheory(Bmix$Y_mix[, 1], title = "Mixture of Beta Distributions",
  fx = function(x) mix_pis[1] * dbeta(x, 6, 3) + mix_pis[2] *
    dbeta(x, 4, 1.5) + mix_pis[3] * dbeta(x, 10, 20), lower = 0, upper = 1)

## End(Not run)
```

`rho_M1M2` *Approximate Correlation between Two Continuous Mixture Variables M1 and M2*

Description

This function approximates the expected correlation between two continous mixture variables $M1$ and $M2$ based on their mixing proportions, component means, component standard deviations, and correlations between components across variables. The equations can be found in the **Expected Cumulants and Correlations for Continuous Mixture Variables** vignette. This function can be used to see what combination of component correlations gives a desired correlation between $M1$ and $M2$.

Usage

```
rho_M1M2(mix_pis = list(), mix_mus = list(), mix_sigmas = list(),
  p_M1M2 = NULL)
```

Arguments

`mix_pis`	a list of length 2 with 1st component a vector of mixing probabilities that sum to 1 for component distributions of $M1$ and likewise for 2nd component and $M2$
`mix_mus`	a list of length 2 with 1st component a vector of means for component distributions of $M1$ and likewise for 2nd component and $M2$
`mix_sigmas`	a list of length 2 with 1st component a vector of standard deviations for component distributions of $M1$ and likewise for 2nd component and $M2$
`p_M1M2`	a matrix of correlations with rows corresponding to $M1$ and columns corresponding to $M2$; i.e., `p_M1M2[1, 2]` is the correlation between the 1st component of $M1$ and the 2nd component of $M2$

Value

the expected correlation between M1 and M2

References

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

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.

See Also

rho_M1Y

Examples

```
# M1 is mixture of N(-2, 1) and N(2, 1);
# M2 is mixture of Logistic(0, 1), Chisq(4), and Beta(4, 1.5)
# pairwise correlation between components across M1 and M2 set to 0.35
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
p_M11M21 <- p_M11M22 <- p_M11M23 <- 0.35
p_M12M21 <- p_M12M22 <- p_M12M23 <- 0.35
p_M1M2 <- matrix(c(p_M11M21, p_M11M22, p_M11M23, p_M12M21, p_M12M22,
  p_M12M23), 2, 3, byrow = TRUE)
rhoM1M2 <- rho_M1M2(mix_pis, mix_mus, mix_sigmas, p_M1M2)
rhoM1M2
```

---

rho_M1Y *Approximate Correlation between Continuous Mixture Variable M1 and Random Variable Y*

Description

This function approximates the expected correlation between a continous mixture variables $M1$ and another random variable Y based on the mixing proportions, component means, and component standard deviations of $M1$ and correlations between components of $M1$ and Y. The equations can be found in the **Expected Cumulants and Correlations for Continuous Mixture Variables** vignette. This function can be used to see what combination of correlations between components of $M1$ and Y gives a desired correlation between $M1$ and Y.

Usage

```
rho_M1Y(mix_pis = NULL, mix_mus = NULL, mix_sigmas = NULL, p_M1Y = NULL)
```

Arguments

mix_pis	a vector of mixing probabilities that sum to 1 for component distributions of $M1$
mix_mus	a vector of means for component distributions of $M1$
mix_sigmas	a vector of standard deviations for component distributions of $M1$
p_M1Y	a vector of correlations between the components of $M1$ and Y; i.e., p_M1Y[1] is the correlation between the 1st component of $M1$ and Y

Value

the expected correlation between M1 and Y

References

Please see references for rho_M1M2.

See Also

rho_M1Y

Examples

```
# M1 is mixture of N(-2, 1) and N(2, 1); C1 is a continuous non-mixture
# variable (but could also be an ordinal or count variable)
# pairwise correlation between components of M1 and C1 set to 0.35
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
p_M11C1 <- p_M12C1 <- 0.35
p_M1C1 <- c(p_M11C1, p_M12C1)
rho_M1C1 <- rho_M1Y(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]], p_M1C1)
rho_M1C1
```

SimCorrMix *Simulation of Correlated Data of Multiple Variable Types including Continuous and Count Mixture Distributions*

Description

SimCorrMix generates continuous (normal, non-normal, or mixture distributions), binary, ordinal, and count (Poisson or Negative Binomial, regular or zero-inflated) variables with a specified correlation matrix, or one continuous variable with a mixture distribution. This package can be used to simulate 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 methods extend those found in the **SimMultiCorrData** package. 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). Non-mixture distributions require the user to specify mean, variance, skewness, standardized kurtosis, and standardized fifth and sixth cumulants. Mixture distributions require these inputs for the component distributions plus the mixing probabilities. 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. However, the package provides functions to approximate expected correlations with continuous mixture variables given target correlations with the components. Binary and ordinal variables are simulated using a modification of GenOrd-package's ordsample function. Count variables are simulated using the inverse CDF method. There are two simulation pathways which calculate intermediate correlations involving count variables differently. Correlation Method 1 adapts Yahav and Shmueli's 2012 method (doi: 10.1002/asmb.901). Correlation Method 2 adapts Barbiero and Ferrari's 2015 modification of GenOrd-package (doi: 10.1002/asmb.2072). The optional error loop may be used to improve the

accuracy of the final correlation matrix. The package also provides functions to calculate the standardized cumulants of continuous mixture distributions, check parameter inputs, calculate feasible correlation boundaries, and summarize and plot simulated variables.

Vignettes

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

1) **Calculation of Correlation Boundaries** explains how the feasible correlation boundaries are calculated for each of the two simulation pathways.

2) **Comparison of Correlation Methods 1 and 2** describes the two simulation pathways that can be followed for generation of correlated data.

3) **Continuous Mixture Distributions** demonstrates how to simulate one continuous mixture variable using `contmixvar1` and gives a step-by-step guideline for comparing a simulated distribution to the target distribution.

4) **Error Loop Algorithm** details the algorithm involved in the optional error loop that helps to minimize correlation errors.

5) **Expected Cumulants and Correlations for Continuous Mixture Variables** derives the equations used by the function `calc_mixmoments` to find the mean, standard deviation, skew, standardized kurtosis, and standardized fifth and sixth cumulants for a continuous mixture variable. The vignette also explains how the functions `rho_M1M2` and `rho_M1Y` approximate the expected correlations with continuous mixture variables based on the target correlations with the components.

6) **Overall Workflow for Generation of Correlated Data** gives a step-by-step guideline to follow with an example containing continuous non-mixture and mixture, ordinal, zero-inflated Poisson, and zero-inflated Negative Binomial variables. It executes both correlated data simulation functions with and without the error loop.

7) **Variable Types** describes the different types of variables that can be simulated in **SimCorrMix**.

Functions

This package contains 3 *simulation* functions:

`contmixvar1`, `corrvar`, and `corrvar2`

4 data description (*summary*) function:

`calc_mixmoments`, `summary_var`, `rho_M1M2`, `rho_M1Y`

2 *graphing* functions:

`plot_simpdf_theory`, `plot_simtheory`

3 *support* functions:

`validpar`, `validcorr`, `validcorr2`

and 16 *auxiliary* functions (should not normally be called by the user, but are called by other functions):

`corr_error`, `intercorr`, `intercorr2`, `intercorr_cat_nb`, `intercorr_cat_pois`, `intercorr_cont_nb`, `intercorr_cont_nb2`, `intercorr_cont_pois`, `intercorr_cont_pois2`, `intercorr_cont`, `intercorr_nb`, `intercorr_pois`, `intercorr_pois_nb`, `maxcount_support`, `ord_norm`, `norm_ord`

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

Carnell R (2017). triangle: Provides the Standard Distribution Functions for the Triangle Distribution. R package version 0.11. https://CRAN.R-project.org/package=triangle.

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.

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, 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.

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

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.

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

summary_var *Summary of Simulated Variables*

Description

This function summarizes the results of contmixvar1, corrvar, or corrvar2. The inputs are either the simulated variables or inputs for those functions. See their documentation for more information. If summarizing result from contmixvar1, mixture parameters may be entered as vectors instead of lists.

Usage

```
summary_var(Y_cat = NULL, Y_cont = NULL, Y_comp = NULL, Y_mix = NULL,
  Y_pois = NULL, Y_nb = NULL, means = NULL, vars = NULL, skews = NULL,
  skurts = NULL, fifths = NULL, sixths = NULL, mix_pis = list(),
  mix_mus = list(), mix_sigmas = list(), mix_skews = list(),
  mix_skurts = list(), mix_fifths = list(), mix_sixths = list(),
  marginal = list(), lam = NULL, p_zip = 0, size = NULL, prob = NULL,
  mu = NULL, p_zinb = 0, rho = NULL)
```

Arguments

Y_cat	a matrix of ordinal variables
Y_cont	a matrix of continuous non-mixture variables
Y_comp	a matrix of components of continuous mixture variables
Y_mix	a matrix of continuous mixture variables
Y_pois	a matrix of Poisson variables
Y_nb	a matrix of Negative Binomial variables
means	a vector of means for the k_cont non-mixture and k_mix mixture continuous variables (i.e. rep(0, (k_cont + k_mix)))
vars	a vector of variances for the k_cont non-mixture and k_mix mixture continuous variables (i.e. rep(1, (k_cont + k_mix)))
skews	a vector of skewness values for the k_cont non-mixture continuous variables
skurts	a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0) for the k_cont non-mixture continuous variables
fifths	a vector of standardized fifth cumulants for the k_cont non-mixture continuous variables (not necessary for method = "Fleishman")
sixths	a vector of standardized sixth cumulants for the k_cont non-mixture continuous variables (not necessary for method = "Fleishman")
mix_pis	a list of length k_mix with i-th component a vector of mixing probabilities that sum to 1 for component distributions of Y_{mix_i}
mix_mus	a list of length k_mix with i-th component a vector of means for component distributions of Y_{mix_i}
mix_sigmas	a list of length k_mix with i-th component a vector of standard deviations for component distributions of Y_{mix_i}
mix_skews	a list of length k_mix with i-th component a vector of skew values for component distributions of Y_{mix_i}

mix_skurts — a list of length k_mix with i-th component a vector of standardized kurtoses for component distributions of Y_{mix_i}

mix_fifths — a list of length k_mix with i-th component a vector of standardized fifth cumulants for component distributions of Y_{mix_i} (not necessary for method = "Fleishman")

mix_sixths — a list of length k_mix with i-th component a vector of standardized sixth cumulants for component distributions of Y_{mix_i} (not necessary for method = "Fleishman")

marginal — a list of length equal to k_cat; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; 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, these should be input the same as for ordinal variables with more than 2 categories (i.e. the user-specified probability is the probability of the 1st category, which has the smaller support value)

lam — a vector of lambda (mean > 0) constants for the Poisson variables (see dpois); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

p_zip — a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see 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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)

size — a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables

prob — a vector of success probability parameters for the NB variables; order the same as in size

mu — a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)

p_zinb — a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)

rho — the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that rho is specified in terms of the components of Y_{mix}

Value

A list whose components vary based on the type of simulated variables.

If **ordinal variables** are produced:

ord_sum a list, where the i-th element contains a data.frame with target and simulated cumulative probabilities for ordinal variable Y_i

If **continuous variables** are produced:

cont_sum a data.frame summarizing Y_cont and Y_comp,

target_sum a data.frame with the target distributions for Y_cont and Y_comp,

mix_sum a data.frame summarizing Y_mix,

target_mix a data.frame with the target distributions for Y_mix,

If **Poisson variables** are produced:

pois_sum a data.frame summarizing Y_pois

If **Negative Binomial variables** are produced:

nb_sum a data.frame summarizing Y_nb

Additionally, the following elements:

rho_calc the final correlation matrix for Y_cat, Y_cont, Y_comp, Y_pois, and Y_nb

rho_mix the final correlation matrix for Y_cat, Y_cont, Y_mix, Y_pois, and Y_nb

maxerr the maximum final correlation error of rho_calc from the target rho.

References

See references for SimCorrMix.

See Also

contmixvar1, corrvar, corrvar2

Examples

```
## Not run:

# 2 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable
n <- 10000
seed <- 1234

# Mixture variables: Normal mixture with 2 components;
# mixture of Logistic(0, 1), Chisq(4), Beta(4, 1.5)
# Find cumulants of components of 2nd mixture variable
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))

skews <- skurts <- fifths <- sixths <- NULL
Six <- list()
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
mix_skews <- list(rep(0, 2), c(L[3], C[3], B[3]))
mix_skurts <- list(rep(0, 2), c(L[4], C[4], B[4]))
mix_fifths <- list(rep(0, 2), c(L[5], C[5], B[5]))
mix_sixths <- list(rep(0, 2), c(L[6], C[6], B[6]))
mix_Six <- list(list(NULL, NULL), list(1.75, NULL, 0.03))
Nstcum <- calc_mixmoments(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]],
```

```
  mix_skews[[1]], mix_skurts[[1]], mix_fifths[[1]], mix_sixths[[1]])
Mstcum <- calc_mixmoments(mix_pis[[2]], mix_mus[[2]], mix_sigmas[[2]],
  mix_skews[[2]], mix_skurts[[2]], mix_fifths[[2]], mix_sixths[[2]])
means <- c(Nstcum[1], Mstcum[1])
vars <- c(Nstcum[2]^2, Mstcum[2]^2)

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
size <- 2
prob <- 0.75
p_zinb <- 0.2

k_cat <- k_pois <- k_nb <- 1
k_cont <- 0
k_mix <- 2
Rey <- matrix(0.39, 8, 8)
diag(Rey) <- 1
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "M2_1", "M2_2",
  "M2_3", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0
Rey["M2_1", "M2_2"] <- Rey["M2_2", "M2_1"] <- Rey["M2_1", "M2_3"] <- 0
Rey["M2_3", "M2_1"] <- Rey["M2_2", "M2_3"] <- Rey["M2_3", "M2_2"] <- 0

# check parameter inputs
validpar(k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, rho = Rey)

# check to make sure Rey is within the feasible correlation boundaries
validcorr(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", means,
  vars, skews, skurts, fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas,
  mix_skews, mix_skurts, mix_fifths, mix_sixths, mix_Six, marginal,
  lam, p_zip, size, prob, mu = NULL, p_zinb, Rey, seed)

# simulate without the error loop
Sim1 <- corrvar(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, Rey, seed, epsilon = 0.01)

Summ1 <- summary_var(Sim1$Y_cat, Y_cont = NULL, Sim1$Y_comp, Sim1$Y_mix,
  Sim1$Y_pois, Sim1$Y_nb, means, vars, skews, skurts, fifths, sixths,
  mix_pis, mix_mus, mix_sigmas, mix_skews, mix_skurts, mix_fifths,
  mix_sixths, marginal, lam, p_zip, size, prob, mu = NULL, p_zinb, Rey)

Sim1_error <- abs(Rey - Summ1$rho_calc)
summary(as.numeric(Sim1_error))


## End(Not run)
```

validcorr *Determine Correlation Bounds for Ordinal, Continuous, Poisson, and/or Negative Binomial Variables: Correlation Method 1*

Description

This function calculates the lower and upper correlation bounds for the given distributions and checks if a given target correlation matrix rho is within the bounds. It should be used before simulation with corrvar. However, even if all pairwise correlations fall within the bounds, it is still possible that the desired correlation matrix is not feasible. This is particularly true when ordinal variables ($r \geq 2$ categories) are generated or negative correlations are desired. Therefore, this function should be used as a general check to eliminate pairwise correlations that are obviously not reproducible. It will help prevent errors when executing the simulation. The *ordering* of the variables in rho must be 1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixture, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, and 7th zero-inflated NB. Note that it is possible for k_cat, k_cont, k_mix, k_pois, and/or k_nb to be 0. The target correlations are specified with respect to the components of the continuous mixture variables. There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with validpar.

Please see the **Comparison of Correlation Methods 1 and 2** vignette for the differences between the two correlation methods, and the **Calculation of Correlation Boundaries** vignette for a detailed explanation of how the correlation boundaries are calculated.

Usage

```
validcorr(n = 10000, k_cat = 0, k_cont = 0, k_mix = 0, k_pois = 0,
  k_nb = 0, method = c("Fleishman", "Polynomial"), means = NULL,
  vars = NULL, skews = NULL, skurts = NULL, fifths = NULL,
  sixths = NULL, 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(), lam = NULL, p_zip = 0, size = NULL, prob = NULL,
  mu = NULL, p_zinb = 0, rho = NULL, seed = 1234, use.nearPD = TRUE)
```

Arguments

n	the sample size (i.e. the length of each simulated variable; default = 10000)
k_cat	the number of ordinal (r >= 2 categories) variables (default = 0)
k_cont	the number of continuous non-mixture variables (default = 0)
k_mix	the number of continuous mixture variables (default = 0)
k_pois	the number of regular Poisson and zero-inflated Poisson variables (default = 0)
k_nb	the number of regular Negative Binomial and zero-inflated Negative Binomial variables (default = 0)
method	the method used to generate the k_cont non-mixture and k_mix mixture continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
means	a vector of means for the k_cont non-mixture and k_mix mixture continuous variables (i.e. rep(0, (k_cont + k_mix)))

vars — a vector of variances for the k_cont non-mixture and k_mix mixture continuous variables (i.e. `rep(1, (k_cont + k_mix))`)

skews — a vector of skewness values for the `k_cont` non-mixture continuous variables

skurts — a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0) for the `k_cont` non-mixture continuous variables

fifths — a vector of standardized fifth cumulants for the `k_cont` non-mixture continuous variables (not necessary for `method` = "Fleishman")

sixths — a vector of standardized sixth cumulants for the `k_cont` non-mixture continuous variables (not necessary for `method` = "Fleishman")

Six — a list of vectors of sixth cumulant correction values for the `k_cont` non-mixture continuous variables if no valid PDF constants are found,
ex: `Six = list(seq(0.01, 2, 0.01), seq(1, 10, 0.5))`; if no correction is desired for variable Y_{cont_i}, set set the i-th list component equal to `NULL`; if no correction is desired for any of the Y_{cont} keep as `Six = list()` (not necessary for `method` = "Fleishman")

mix_pis — a list of length `k_mix` with i-th component a vector of mixing probabilities that sum to 1 for component distributions of Y_{mix_i}

mix_mus — a list of length `k_mix` with i-th component a vector of means for component distributions of Y_{mix_i}

mix_sigmas — a list of length `k_mix` with i-th component a vector of standard deviations for component distributions of Y_{mix_i}

mix_skews — a list of length `k_mix` with i-th component a vector of skew values for component distributions of Y_{mix_i}

mix_skurts — a list of length `k_mix` with i-th component a vector of standardized kurtoses for component distributions of Y_{mix_i}

mix_fifths — a list of length `k_mix` with i-th component a vector of standardized fifth cumulants for component distributions of Y_{mix_i} (not necessary for `method` = "Fleishman")

mix_sixths — a list of length `k_mix` with i-th component a vector of standardized sixth cumulants for component distributions of Y_{mix_i} (not necessary for `method` = "Fleishman")

mix_Six — a list of length `k_mix` with i-th component a list of vectors of sixth cumulant correction values for component distributions of Y_{mix_i}; use `NULL` if no correction is desired for a given component or mixture variable; if no correction is desired for any of the Y_{mix} keep as `mix_Six = list()` (not necessary for `method` = "Fleishman")

marginal — a list of length equal to `k_cat`; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; 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, these should be input the same as for ordinal variables with more than 2 categories (i.e. the user-specified probability is the probability of the 1st category, which has the smaller support value)

lam — a vector of lambda (> 0) constants for the Poisson variables (see `dpois`); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

p_zip — a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `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 dpospois); if length(p_zip) < length(lam), the missing values are set to 0 (and ordered 1st)
size	a vector of size parameters for the Negative Binomial variables (see dnbinom); the order should be 1st regular NB variables, 2nd zero-inflated NB variables
prob	a vector of success probability parameters for the NB variables; order the same as in size
mu	a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)
p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)
rho	the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that rho is specified in terms of the components of Y_{mix}
seed	the seed value for random number generation (default = 1234)
use.nearPD	TRUE to convert rho to the nearest positive definite matrix with Matrix::nearPD if necessary

Value

A list with components:

rho the target correlation matrix, which will differ from the supplied matrix (if provided) if it was converted to the nearest positive-definite matrix

L_rho the lower correlation bound

U_rho the upper correlation bound

If continuous variables are desired, additional components are:

constants the calculated constants

sixth_correction a vector of the sixth cumulant correction values

valid.pdf a vector with i-th component equal to "TRUE" if variable Y_i has a valid power method PDF, else "FALSE"

If a target correlation matrix rho is provided, each pairwise correlation is checked to see if it is within the lower and upper bounds. If the correlation is outside the bounds, the indices of the variable pair are given.

valid.rho TRUE if all entries of rho are within the bounds, else FALSE

Reasons for Function Errors

1) The most likely cause for function errors is that no solutions to `fleish` or `poly` converged when using `find_constants`. If this happens, the function 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.

2) The kurtosis may be outside the region of possible values. There is an associated lower boundary for kurtosis 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

Please see `corrvar` for additional references.

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.

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

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.

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

See Also

`find_constants`, `corrvar`, `validpar`

Examples

```
## Not run:

# 2 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable
n <- 10000
seed <- 1234

# Mixture variables: Normal mixture with 2 components;
# mixture of Logistic(0, 1), Chisq(4), Beta(4, 1.5)
# Find cumulants of components of 2nd mixture variable
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))

skews <- skurts <- fifths <- sixths <- NULL
Six <- list()
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
```

```
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
mix_skews <- list(rep(0, 2), c(L[3], C[3], B[3]))
mix_skurts <- list(rep(0, 2), c(L[4], C[4], B[4]))
mix_fifths <- list(rep(0, 2), c(L[5], C[5], B[5]))
mix_sixths <- list(rep(0, 2), c(L[6], C[6], B[6]))
mix_Six <- list(list(NULL, NULL), list(1.75, NULL, 0.03))
Nstcum <- calc_mixmoments(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]],
  mix_skews[[1]], mix_skurts[[1]], mix_fifths[[1]], mix_sixths[[1]])
Mstcum <- calc_mixmoments(mix_pis[[2]], mix_mus[[2]], mix_sigmas[[2]],
  mix_skews[[2]], mix_skurts[[2]], mix_fifths[[2]], mix_sixths[[2]])
means <- c(Nstcum[1], Mstcum[1])
vars <- c(Nstcum[2]^2, Mstcum[2]^2)

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
size <- 2
prob <- 0.75
p_zinb <- 0.2

k_cat <- k_pois <- k_nb <- 1
k_cont <- 0
k_mix <- 2
Rey <- matrix(0.39, 8, 8)
diag(Rey) <- 1
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "M2_1", "M2_2",
  "M2_3", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0
Rey["M2_1", "M2_2"] <- Rey["M2_2", "M2_1"] <- Rey["M2_1", "M2_3"] <- 0
Rey["M2_3", "M2_1"] <- Rey["M2_2", "M2_3"] <- Rey["M2_3", "M2_2"] <- 0

# check parameter inputs
validpar(k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, rho = Rey)

# check to make sure Rey is within the feasible correlation boundaries
validcorr(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", means,
  vars, skews, skurts, fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas,
  mix_skews, mix_skurts, mix_fifths, mix_sixths, mix_Six, marginal,
  lam, p_zip, size, prob, mu = NULL, p_zinb, Rey, seed)

## End(Not run)
```

`validcorr2` *Determine Correlation Bounds for Ordinal, Continuous, Poisson, and/or Negative Binomial Variables: Correlation Method 2*

Description

This function calculates the lower and upper correlation bounds for the given distributions and checks if a given target correlation matrix `rho` is within the bounds. It should be used before sim-

ulation with corrvar2. However, even if all pairwise correlations fall within the bounds, it is still possible that the desired correlation matrix is not feasible. This is particularly true when ordinal variables ($r \geq 2$ categories) are generated or negative correlations are desired. Therefore, this function should be used as a general check to eliminate pairwise correlations that are obviously not reproducible. It will help prevent errors when executing the simulation. The *ordering* of the variables in rho must be 1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixture, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, and 7th zero-inflated NB. Note that it is possible for k_cat, k_cont, k_mix, k_pois, and/or k_nb to be 0. The target correlations are specified with respect to the components of the continuous mixture variables. There are no parameter input checks in order to decrease simulation time. All inputs should be checked prior to simulation with validpar.

Please see the **Comparison of Correlation Methods 1 and 2** vignette for the differences between the two correlation methods, and the **Calculation of Correlation Boundaries** vignette for a detailed explanation of how the correlation boundaries are calculated.

Usage

```
validcorr2(n = 10000, k_cat = 0, k_cont = 0, k_mix = 0, k_pois = 0,
  k_nb = 0, method = c("Fleishman", "Polynomial"), means = NULL,
  vars = NULL, skews = NULL, skurts = NULL, fifths = NULL,
  sixths = NULL, 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(), lam = NULL, p_zip = 0, size = NULL, prob = NULL,
  mu = NULL, p_zinb = 0, pois_eps = 1e-04, nb_eps = 1e-04, rho = NULL,
  seed = 1234, use.nearPD = TRUE)
```

Arguments

n	the sample size (i.e. the length of each simulated variable; default = 10000)
k_cat	the number of ordinal (r >= 2 categories) variables (default = 0)
k_cont	the number of continuous non-mixture variables (default = 0)
k_mix	the number of continuous mixture variables (default = 0)
k_pois	the number of regular Poisson and zero-inflated Poisson variables (default = 0)
k_nb	the number of regular Negative Binomial and zero-inflated Negative Binomial variables (default = 0)
method	the method used to generate the k_cont non-mixture and k_mix mixture continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
means	a vector of means for the k_cont non-mixture and k_mix mixture continuous variables (i.e. rep(0, (k_cont + k_mix)))
vars	a vector of variances for the k_cont non-mixture and k_mix mixture continuous variables (i.e. rep(1, (k_cont + k_mix)))
skews	a vector of skewness values for the k_cont non-mixture continuous variables
skurts	a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0) for the k_cont non-mixture continuous variables
fifths	a vector of standardized fifth cumulants for the k_cont non-mixture continuous variables (not necessary for method = "Fleishman")

`sixths` a vector of standardized sixth cumulants for the `k_cont` non-mixture continuous variables (not necessary for `method` = "Fleishman")

`Six` a list of vectors of sixth cumulant correction values for the `k_cont` non-mixture continuous variables if no valid PDF constants are found,
ex: `Six = list(seq(0.01, 2, 0.01), seq(1, 10, 0.5))`; if no correction is desired for variable Y_{cont_i}, set set the i-th list component equal to `NULL`; if no correction is desired for any of the Y_{cont} keep as `Six = list()` (not necessary for `method` = "Fleishman")

`mix_pis` a list of length `k_mix` with i-th component a vector of mixing probabilities that sum to 1 for component distributions of Y_{mix_i}

`mix_mus` a list of length `k_mix` with i-th component a vector of means for component distributions of Y_{mix_i}

`mix_sigmas` a list of length `k_mix` with i-th component a vector of standard deviations for component distributions of Y_{mix_i}

`mix_skews` a list of length `k_mix` with i-th component a vector of skew values for component distributions of Y_{mix_i}

`mix_skurts` a list of length `k_mix` with i-th component a vector of standardized kurtoses for component distributions of Y_{mix_i}

`mix_fifths` a list of length `k_mix` with i-th component a vector of standardized fifth cumulants for component distributions of Y_{mix_i} (not necessary for `method` = "Fleishman")

`mix_sixths` a list of length `k_mix` with i-th component a vector of standardized sixth cumulants for component distributions of Y_{mix_i} (not necessary for `method` = "Fleishman")

`mix_Six` a list of length `k_mix` with i-th component a list of vectors of sixth cumulant correction values for component distributions of Y_{mix_i}; use `NULL` if no correction is desired for a given component or mixture variable; if no correction is desired for any of the Y_{mix} keep as `mix_Six = list()` (not necessary for `method` = "Fleishman")

`marginal` a list of length equal to `k_cat`; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; 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, these should be input the same as for ordinal variables with more than 2 categories (i.e. the user-specified probability is the probability of the 1st category, which has the smaller support value)

`lam` a vector of lambda (> 0) constants for the Poisson variables (see `dpois`); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

`p_zip` a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `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 `dpospois`); if `length(p_zip) < length(lam)`, the missing values are set to 0 (and ordered 1st)

`size` a vector of size parameters for the Negative Binomial variables (see `dnbinom`); the order should be 1st regular NB variables, 2nd zero-inflated NB variables

`prob` a vector of success probability parameters for the NB variables; order the same as in `size`

mu	a vector of mean parameters for the NB variables (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in size; for zero-inflated NB this refers to the mean of the NB distribution (see dzinegbin)
p_zinb	a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see 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 dposnegbin); if length(p_zinb) < length(size), the missing values are set to 0 (and ordered 1st)
pois_eps	a vector of length k_pois containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable
nb_eps	a vector of length k_nb containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable
rho	the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that rho is specified in terms of the components of Y_{mix}
seed	the seed value for random number generation (default = 1234)
use.nearPD	TRUE to convert rho to the nearest positive definite matrix with Matrix::nearPD if necessary

Value

A list with components:

rho the target correlation matrix, which will differ from the supplied matrix (if provided) if it was converted to the nearest positive-definite matrix

L_rho the lower correlation bound

U_rho the upper correlation bound

If continuous variables are desired, additional components are:

constants the calculated constants

sixth_correction a vector of the sixth cumulant correction values

valid.pdf a vector with i-th component equal to "TRUE" if variable Y_i has a valid power method PDF, else "FALSE"

If a target correlation matrix rho is provided, each pairwise correlation is checked to see if it is within the lower and upper bounds. If the correlation is outside the bounds, the indices of the variable pair are given.

valid.rho TRUE if all entries of rho are within the bounds, else FALSE

Reasons for Function Errors

1) The most likely cause for function errors is that no solutions to fleish or poly converged when using find_constants. If this happens, the function 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.

2) The kurtosis may be outside the region of possible values. There is an associated lower boundary for kurtosis 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

Please see corrvar2 and validcorr for references.

See Also

find_constants, corrvar2, validpar

Examples

```
## Not run:

# 2 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable
n <- 10000
seed <- 1234

# Mixture variables: Normal mixture with 2 components;
# mixture of Logistic(0, 1), Chisq(4), Beta(4, 1.5)
# Find cumulants of components of 2nd mixture variable
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))

skews <- skurts <- fifths <- sixths <- NULL
Six <- list()
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
mix_skews <- list(rep(0, 2), c(L[3], C[3], B[3]))
mix_skurts <- list(rep(0, 2), c(L[4], C[4], B[4]))
mix_fifths <- list(rep(0, 2), c(L[5], C[5], B[5]))
mix_sixths <- list(rep(0, 2), c(L[6], C[6], B[6]))
mix_Six <- list(list(NULL, NULL), list(1.75, NULL, 0.03))
Nstcum <- calc_mixmoments(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]],
  mix_skews[[1]], mix_skurts[[1]], mix_fifths[[1]], mix_sixths[[1]])
Mstcum <- calc_mixmoments(mix_pis[[2]], mix_mus[[2]], mix_sigmas[[2]],
  mix_skews[[2]], mix_skurts[[2]], mix_fifths[[2]], mix_sixths[[2]])
means <- c(Nstcum[1], Mstcum[1])
vars <- c(Nstcum[2]^2, Mstcum[2]^2)

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
pois_eps <- 0.0001
size <- 2
prob <- 0.75
p_zinb <- 0.2
nb_eps <- 0.0001

k_cat <- k_pois <- k_nb <- 1
```

```
k_cont <- 0
k_mix <- 2
Rey <- matrix(0.39, 8, 8)
diag(Rey) <- 1
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "M2_1", "M2_2",
  "M2_3", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0
Rey["M2_1", "M2_2"] <- Rey["M2_2", "M2_1"] <- Rey["M2_1", "M2_3"] <- 0
Rey["M2_3", "M2_1"] <- Rey["M2_2", "M2_3"] <- Rey["M2_3", "M2_2"] <- 0

# check parameter inputs
validpar(k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, pois_eps, nb_eps, Rey)

# check to make sure Rey is within the feasible correlation boundaries
validcorr2(n, k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", means,
  vars, skews, skurts, fifths, sixths, Six, mix_pis, mix_mus, mix_sigmas,
  mix_skews, mix_skurts, mix_fifths, mix_sixths, mix_Six, marginal,
  lam, p_zip, size, prob, mu = NULL, p_zinb, pois_eps, nb_eps, Rey, seed)

## End(Not run)
```

---

validpar *Parameter Check for Simulation or Correlation Validation Functions*

Description

This function checks the parameter inputs to the simulation functions contmixvar1, corrvar, and corrvar2 and to the correlation validation functions validcorr and validcorr2. 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.

Usage

```
validpar(k_cat = 0, k_cont = 0, k_mix = 0, k_pois = 0, k_nb = 0,
  method = c("Fleishman", "Polynomial"), means = NULL, vars = NULL,
  skews = NULL, skurts = NULL, fifths = NULL, sixths = NULL,
  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 = NULL, p_zip = 0, size = NULL, prob = NULL,
  mu = NULL, p_zinb = 0, pois_eps = 1e-04, nb_eps = 1e-04, rho = NULL,
  Sigma = NULL, cstart = list())
```

Arguments

k_cat	the number of ordinal (r >= 2 categories) variables (default = 0)
k_cont	the number of continuous non-mixture variables (default = 0)
k_mix	the number of continuous mixture variables (default = 0)
k_pois	the number of regular Poisson and zero-inflated Poisson variables (default = 0)
k_nb	the number of regular Negative Binomial and zero-inflated Negative Binomial variables (default = 0)
method	the method used to generate the k_cont non-mixture and k_mix mixture continuous variables. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
means	a vector of means for the k_cont non-mixture and k_mix mixture continuous variables (i.e. rep(0, (k_cont + k_mix)))
vars	a vector of variances for the k_cont non-mixture and k_mix mixture continuous variables (i.e. rep(1, (k_cont + k_mix)))
skews	a vector of skewness values for the k_cont non-mixture continuous variables
skurts	a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0) for the k_cont non-mixture continuous variables
fifths	a vector of standardized fifth cumulants for the k_cont non-mixture continuous variables (not necessary for method = "Fleishman")
sixths	a vector of standardized sixth cumulants for the k_cont non-mixture continuous variables (not necessary for method = "Fleishman")
Six	a list of vectors of sixth cumulant correction values for the k_cont non-mixture continuous variables if no valid PDF constants are found, ex: Six = list(seq(0.01, 2, 0.01), seq(1, 10, 0.5)); if no correction is desired for variable Y_{cont_i}, set set the i-th list component equal to NULL; if no correction is desired for any of the Y_{cont} keep as Six = list() (not necessary for method = "Fleishman")
mix_pis	a vector if using contmixvar1 or a list of length k_mix with i-th component a vector of mixing probabilities that sum to 1 for component distributions of Y_{mix_i}
mix_mus	a vector if using contmixvar1 or a list of length k_mix with i-th component a vector of means for component distributions of Y_{mix_i}
mix_sigmas	a vector if using contmixvar1 or a list of length k_mix with i-th component a vector of standard deviations for component distributions of Y_{mix_i}
mix_skews	a vector if using contmixvar1 or a list of length k_mix with i-th component a vector of skew values for component distributions of Y_{mix_i}
mix_skurts	a vector if using contmixvar1 or a list of length k_mix with i-th component a vector of standardized kurtoses for component distributions of Y_{mix_i}
mix_fifths	a vector if using contmixvar1 or a list of length k_mix with i-th component a vector of standardized fifth cumulants for component distributions of Y_{mix_i} (not necessary for method = "Fleishman")
mix_sixths	a vector if using contmixvar1 or a list of length k_mix with i-th component a vector of standardized sixth cumulants for component distributions of Y_{mix_i} (not necessary for method = "Fleishman")

`mix_Six` if using `contmixvar1`, a list of vectors of sixth cumulant corrections for the components of the continuous mixture variable; else a list of length `k_mix` with i-th component a list of vectors of sixth cumulant correction values for component distributions of Y_{mix_i}; use `NULL` if no correction is desired for a given component or mixture variable; if no correction is desired for any of the Y_{mix} keep as `mix_Six = list()` (not necessary for `method` = "Fleishman")

`marginal` a list of length equal to `k_cat`; the i-th element is a vector of the cumulative probabilities defining the marginal distribution of the i-th variable; if the variable can take r values, the vector will contain r - 1 probabilities (the r-th is assumed to be 1; default = list()); for binary variables, these should be input the same as for ordinal variables with more than 2 categories (i.e. the user-specified probability is the probability of the 1st category, which has the smaller support value)

`support` a list of length equal to `k_cat`; the i-th element is a vector containing the r ordered support values; if not provided (i.e. `support = list()`), the default is for the i-th element to be the vector 1, ..., r

`lam` a vector of lambda (mean > 0) constants for the Poisson variables (see `dpois`); the order should be 1st regular Poisson variables, 2nd zero-inflated Poisson variables

`p_zip` a vector of probabilities of structural zeros (not including zeros from the Poisson distribution) for the zero-inflated Poisson variables (see `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 `dpospois`); if `length(p_zip) < length(lam)`, the missing values are set to 0 (and ordered 1st)

`size` a vector of size parameters for the Negative Binomial variables (see `dnbinom`); the order should be 1st regular NB variables, 2nd zero-inflated NB variables

`prob` a vector of success probability parameters for the NB variables; order the same as in `size`

`mu` a vector of mean parameters for the NB variables (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL); order the same as in `size`; for zero-inflated NB this refers to the mean of the NB distribution (see `dzinegbin`)

`p_zinb` a vector of probabilities of structural zeros (not including zeros from the NB distribution) for the zero-inflated NB variables (see `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 `dposnegbin`); if `length(p_zinb) < length(size)`, the missing values are set to 0 (and ordered 1st)

`pois_eps` a vector of length `k_pois` containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable

`nb_eps` a vector of length `k_nb` containing total cumulative probability truncation values; if none are provided, the default is 0.0001 for each variable

`rho` the target correlation matrix which must be ordered *1st ordinal, 2nd continuous non-mixture, 3rd components of continuous mixtures, 4th regular Poisson, 5th zero-inflated Poisson, 6th regular NB, 7th zero-inflated NB*; note that `rho` is specified in terms of the components of Y_{mix}

Sigma	an intermediate correlation matrix to use if the user wants to provide one, else it is calculated within by intercorr
cstart	a list of length equal to k_cont + the total number of mixture components containing initial values for root-solving algorithm used in find_constants. If user specified, each list element must be input as a matrix. For method = "Fleishman", each should have 3 columns for $c1, c2, c3$; for method = "Polynomial", each should have 5 columns for $c1, c2, c3, c4, c5$. If no starting values are specified for a given component, that list element should be NULL.

Value

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

See Also

contmixvar1, corrvar, corrvar2, validcorr, validcorr2

Examples

```
## Not run:
# 2 continuous mixture, 1 binary, 1 zero-inflated Poisson, and
# 1 zero-inflated NB variable

# Mixture variables: Normal mixture with 2 components;
# mixture of Logistic(0, 1), Chisq(4), Beta(4, 1.5)
# Find cumulants of components of 2nd mixture variable
L <- calc_theory("Logistic", c(0, 1))
C <- calc_theory("Chisq", 4)
B <- calc_theory("Beta", c(4, 1.5))

skews <- skurts <- fifths <- sixths <- NULL
Six <- list()
mix_pis <- list(c(0.4, 0.6), c(0.3, 0.2, 0.5))
mix_mus <- list(c(-2, 2), c(L[1], C[1], B[1]))
mix_sigmas <- list(c(1, 1), c(L[2], C[2], B[2]))
mix_skews <- list(rep(0, 2), c(L[3], C[3], B[3]))
mix_skurts <- list(rep(0, 2), c(L[4], C[4], B[4]))
mix_fifths <- list(rep(0, 2), c(L[5], C[5], B[5]))
mix_sixths <- list(rep(0, 2), c(L[6], C[6], B[6]))
mix_Six <- list(list(NULL, NULL), list(1.75, NULL, 0.03))
Nstcum <- calc_mixmoments(mix_pis[[1]], mix_mus[[1]], mix_sigmas[[1]],
  mix_skews[[1]], mix_skurts[[1]], mix_fifths[[1]], mix_sixths[[1]])
Mstcum <- calc_mixmoments(mix_pis[[2]], mix_mus[[2]], mix_sigmas[[2]],
  mix_skews[[2]], mix_skurts[[2]], mix_fifths[[2]], mix_sixths[[2]])
means <- c(Nstcum[1], Mstcum[1])
vars <- c(Nstcum[2]^2, Mstcum[2]^2)

marginal <- list(0.3)
support <- list(c(0, 1))
lam <- 0.5
p_zip <- 0.1
size <- 2
prob <- 0.75
p_zinb <- 0.2

k_cat <- k_pois <- k_nb <- 1
```

```
k_cont <- 0
k_mix <- 2
Rey <- matrix(0.39, 8, 8)
diag(Rey) <- 1
rownames(Rey) <- colnames(Rey) <- c("O1", "M1_1", "M1_2", "M2_1", "M2_2",
  "M2_3", "P1", "NB1")

# set correlation between components of the same mixture variable to 0
Rey["M1_1", "M1_2"] <- Rey["M1_2", "M1_1"] <- 0
Rey["M2_1", "M2_2"] <- Rey["M2_2", "M2_1"] <- Rey["M2_1", "M2_3"] <- 0
Rey["M2_3", "M2_1"] <- Rey["M2_2", "M2_3"] <- Rey["M2_3", "M2_2"] <- 0

# use before contmixvar1 with 1st mixture variable:
# change mix_pis to not sum to 1

check1 <- validpar(k_mix = 1, method = "Polynomial", means = Nstcum[1],
  vars = Nstcum[2]^2, mix_pis = C(0.4, 0.5), mix_mus = mix_mus[[1]],
  mix_sigmas = mix_sigmas[[1]], mix_skews = mix_skews[[1]],
  mix_skurts = mix_skurts[[1]], mix_fifths = mix_fifths[[1]],
  mix_sixths = mix_sixths[[1]])

# use before validcorr: should return TRUE

check2 <- validpar(k_cat, k_cont, k_mix, k_pois, k_nb, "Polynomial", 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 = NULL, p_zinb, rho = Rey)


## End(Not run)
```

Index

∗Topic **Fleishman,**
intercorr_cont, 34
∗Topic **Fleishman**
contmixvar1, 4
corrvar, 8
corrvar2, 16
∗Topic **Headrick**
contmixvar1, 4
corrvar, 8
corrvar2, 16
intercorr_cont, 34
∗Topic **NegativeBinomial**
corrvar, 8
corrvar2, 16
intercorr_cat_nb, 31
intercorr_cont_nb, 35
intercorr_cont_nb2, 37
intercorr_nb, 42
intercorr_pois_nb, 44
maxcount_support, 46
∗Topic **ParameterCheck**
validpar, 74
∗Topic **Poisson**
corrvar, 8
corrvar2, 16
intercorr_cat_pois, 33
intercorr_cont_pois, 38
intercorr_cont_pois2, 40
intercorr_pois, 43
intercorr_pois_nb, 44
maxcount_support, 46
∗Topic **bounds**
validcorr, 65
validcorr2, 69
∗Topic **continuous,**
intercorr_cont, 34
∗Topic **continuous**
contmixvar1, 4
corrvar, 8
corrvar2, 16
intercorr_cont_nb, 35
intercorr_cont_nb2, 37
intercorr_cont_pois, 38
intercorr_cont_pois2, 40
norm_ord, 47
∗Topic **correlation,**
intercorr_cont, 34
∗Topic **correlation**
corr_error, 24
intercorr, 26
intercorr2, 29
intercorr_cat_nb, 31
intercorr_cat_pois, 33
intercorr_cont_nb, 35
intercorr_cont_nb2, 37
intercorr_cont_pois, 38
intercorr_cont_pois2, 40
intercorr_nb, 42
intercorr_pois, 43
intercorr_pois_nb, 44
norm_ord, 47
ord_norm, 48
rho_M1M2, 55
rho_M1Y, 56
validcorr, 65
validcorr2, 69
∗Topic **cumulants**
calc_mixmoments, 3
∗Topic **error**
corr_error, 24
∗Topic **method1**
corrvar, 8
intercorr, 26
intercorr_cat_nb, 31
intercorr_cat_pois, 33
intercorr_cont_nb, 35
intercorr_cont_pois, 38
intercorr_nb, 42
intercorr_pois, 43
intercorr_pois_nb, 44
validcorr, 65
∗Topic **method2**
corrvar2, 16
intercorr2, 29
intercorr_cont_nb2, 37
intercorr_cont_pois2, 40

maxcount_support, 46
validcorr2, 69
∗Topic **mixture**
calc_mixmoments, 3
contmixvar1, 4
corrvar, 8
corrvar2, 16
rho_M1M2, 55
rho_M1Y, 56
∗Topic **ordinal**
corrvar, 8
corrvar2, 16
intercorr_cat_nb, 31
intercorr_cat_pois, 33
norm_ord, 47
ord_norm, 48
∗Topic **plot**
plot_simpdf_theory, 50
plot_simtheory, 52
∗Topic **simulation**
contmixvar1, 4
corrvar, 8
corrvar2, 16
∗Topic **summary**
summary_var, 61

calc_lower_skurt, *6, 7*, *12, 13*, *21*, *68*, *73*
calc_mixmoments, 3, *58*
calc_theory, *51*, *54*
contmixvar1, *3*, 4, *58*, *61*, *63*, *74–77*
contord, *47*
corr_error, *11*, *14*, *19*, *22*, 24, *26*, *58*
corrvar, *3*, 8, *24*, *26*, *28*, *32–34*, *36, 37*, *39, 40*, *42–46*, *49, 50*, *58*, *61*, *63*, *65*, *68*, *74*, *77*
corrvar2, *3*, 16, *24*, *26*, *29*, *31*, *37, 38*, *41*, *46–50*, *58*, *61*, *63*, *70*, *73, 74*, *77*

dnbinom, *11*, *19*, *25*, *27*, *30*, *32*, *36*, *42*, *45, 46*, *62*, *67*, *71*, *76*
dpois, *10*, *18*, *25*, *27*, *30*, *33*, *39*, *43*, *45, 46*, *62*, *66*, *71*, *76*
dposnegbin, *11*, *19*, *28*, *30*, *32*, *36*, *42*, *45*, *47*, *62*, *67*, *72*, *76*
dpospois, *10*, *19*, *27*, *30*, *33*, *39*, *44–46*, *62*, *67*, *71*, *76*
dzinegbin, *11*, *19*, *25*, *27, 28*, *30*, *32*, *36*, *42*, *45*, *47*, *62*, *67*, *72*, *76*
dzipois, *10*, *18*, *25*, *27*, *30*, *33*, *39*, *44–46*, *62*, *66*, *71*, *76*

find_constants, *3*, *5–8*, *11*, *13, 14*, *17*, *19*, *21, 22*, *25*, *27*, *30*, *35–41*, *68*, *72, 73*, *77*
findintercorr_cont_cat, *37*, *40*
fleish, *7*, *13*, *21*, *68*, *72*

geom_col, *50, 51*
geom_density, *50, 51*
geom_histogram, *52*, *54*
geom_line, *50, 51*
ggplot, *51*, *54*

intercorr, *8*, *11*, *14*, 26, *29*, *32–37*, *39, 40*, *42–46*, *50*, *58*, *77*
intercorr2, *16*, *19*, *22*, *26*, 29, *34, 35*, *37, 38*, *41*, *46, 47*, *50*, *58*
intercorr_cat_nb, 31, *58*
intercorr_cat_pois, *33*, 33, *58*
intercorr_cont, 34, *58*
intercorr_cont_nb, 35, *58*
intercorr_cont_nb2, 37, *58*
intercorr_cont_pois, *37*, 38, *41*, *58*
intercorr_cont_pois2, *38*, 40, *58*
intercorr_nb, 42, *44*, *46*, *58*
intercorr_pois, *43*, 43, *46*, *58*
intercorr_pois_nb, *43, 44*, 44, *58*

maxcount_support, *37, 38*, *40, 41*, 46, *58*

nearPD, *49*
nleqslv, *34, 35*
norm_ord, 47, *48*, *50*, *58*

ord_norm, *11*, *19, 20*, *28*, *30*, *47, 48*, 48, *58*
ordcont, *24*, *48*
ordsample, *57*

plot_simpdf_theory, 50, *58*
plot_simtheory, *51*, 52, *58*
pmvnorm, *47*
poly, *7*, *13*, *21*, *68*, *72*
power_norm_corr, *38*, *40, 41*

rho_M1M2, 55, *57, 58*
rho_M1Y, *56*, 56, *57, 58*
rmultinom, *6*

SimCorrMix, 57, *63*
SimCorrMix-package *(*SimCorrMix*)*, 57
SimMultiCorrData, *37*, *40*
summary_var, *5*, *7, 8*, *14*, *16*, *22*, *58*, 61

triangle, *51*, *53*

validcorr, *8*, *13, 14*, *58*, 65, *73, 74*, *77*
validcorr2, *16*, *21, 22*, *58*, 69, *74*, *77*
validpar, *5*, *7, 8*, *14*, *16*, *22*, *26*, *29*, *58*, *65*, *68*, *70*, *73*, 74